Iterative Solution of Large-Scale Systems by Hybrid Techniques

WEI CHEN AND LAWRENCE P. MCNAMEE, MEMBER, IEEE

Abstract—A hybrid computing technique is presented for improving the resolution of the discrete-space discrete-time (DSDT) resistive network method and the treatment of large-scale systems encountered in the solution of partial differential equations.

Index Terms—Hybrid computation, iterative solutions, resistive grid, structure.

In the solution process, the large-scale system is first partitioned into a number of smaller and more easily manageable sections. The partial solutions obtained from the sections are then connected together to yield the overall solution by means of iterative computations. In the hybrid approach, the repetitive computations are carried out in a parallel mode by the analog equipment, such that the per iterative cycle time can be substantially reduced. The convergence of the hybrid iterative schemes is inferred by the positive definiteness of the network equations. Thus the convergence of the hybrid iterative method requires that the equation system be identified with a physically realizable passive resistive network. A digital simulation example is also included to illustrate the concepts involved.

I. INTRODUCTION

In practice, closed form solutions to partial differential equations (PDE) are difficult to formulate. Most often numerical techniques are employed, although other computer methods are available [1]–[4].

The numerical procedure usually starts with the replacement of the original PDE by a corresponding set of finite-difference equations. The finite-difference approximations can be established either in an explicit form or by a system of simultaneous algebraic equations. The solution to the explicit finite-difference approximations can be computed straightforwardly, but it is subject to numerical instability. The simultaneous solution of the explicit finite-difference approximations is unconditionally stable regardless of discretization sizes [15]–[19]. However, the computational task of solving a large system of simultaneous algebraic equations is by no means trivial. Run times can be excessive because of the serial mode of computation; likewise, roundoff errors can be significant because of the large number of calculations required.

In 1953, it appeared that Gabriel Kron overcame many of these computational difficulties [20]. He presented a procedure for finding the solution to large systems of equations without resorting to the inversion of large matrices. Kron’s method, which is based upon network analogs, consists of three steps: tearing a large network problem into a smaller number of manageable problems, then solving each network separately, and finally recombing the individual solutions, under boundary constraints, to obtain the solution of the original network problem. The validity of Kron’s method was proved by Roth [21] in 1955.

Kron’s method, however, did not achieve the high degree of success that the scheme seemed to offer. It was evident that significant loss could become dominant when the solutions of the subnetworks were recombined to form the overall solution. Nevertheless, Kron’s method, because of its electrical analog approach, did provide a strong impetus to other partitioning methods, particularly those associated with hybrid computing.

Resistive network analogs have also been employed to solve partial differential equations. The analogy is obtained by approximating the continuous system by an array of lumped elements. The discretization of the continuous system leads to an accurate approach to the solution of field problems provided that the grid increments formed by the lumped elements are sufficiently fine. Mathematically the discretized system is described by a set of simultaneous algebraic equations. Both the “physical approach” that replaces differential elements of the continuous system by the corresponding lumped elements and the mathematical derivation that yields the algebraic equation system were described by Karplus [22].

The resistive network solution retains the numerical stability of the implicit formulation, and the network relaxes to the solution without the burden of actually solving simultaneous equations. The general approach that makes use of finite-difference approximations to represent the variables along the space and time coordinates is known as the discrete-space discrete-time (DSDT) method [23]–[26]. The DSDT method based on resistive network analogies has been employed successfully in the treatment of various types of partial differential equations with arbitrary boundaries.

In order to achieve the required accuracy and resolution, a large number of resistive elements must be used. It is important to note, however, that the size of the resistive network is physically bounded by the voltage resolution limits.
and component accuracy, as well as the amount of equipment and space available.

Hybrid computing techniques have proved to be most effective in overcoming many of the computational limits to purely digital computation and equipment requirements of analog computation. The special DSDT hybrid computer system designed by Karplus to solve nonlinear partial differential equations is representative of such approaches [24]. In his system, analog elements operating under digital computer control are called upon at each step of the computation to provide nonlinear function generation and matrix inversion whenever needed. The resolution is controlled by selecting as many of the analog elements as required and then iterating between the analog outputs and the digital computer until a convergence criterion is satisfied.

A continuous-time discrete-space (CTDS) hybrid computing technique has also been proposed and simulated by Hsu and Howe [27], who demonstrated their method by solving a number of boundary value problems. The CTDS technique requires that the problem be partitioned according to the available equipment and that successive runs be made between each analog station until a tolerance limit is satisfied at each analog node. No criteria were presented to guarantee that convergence could be obtained.

In this paper, a DSDT hybrid computing method for handling large-scale linear algebraic problems is presented. The basic approach calls for the partitioning of the original large system into a number of smaller sections whose size is dictated by the available analog equipment. The analog network is then used to solve each section one at a time starting with some convenient boundary conditions between sections. The solutions for the sections so obtained are combined iteratively to yield the overall solution by satisfying boundary conditions between sections. A convergence criterion based on a priori physically realizability is also provided.

The development starts with the derivation of the positive definite properties associated with resistive networks. The role of the iterative method in combining the partial solutions is then presented. Finally, the hybrid computing approach, including a numerical example, is described.

II. RESISTIVE NETWORKS

In this section, the linear algebraic property of resistive networks is called upon to establish the convergence criteria for the proposed iterative scheme. It is shown that for equation systems with positive definite coefficient matrices, a general class of iterative schemes converge.

A. The Network Equations

The voltage–current relationship in a resistive network can be expressed by a system of linear equations [28]. The equations are obtained by applying one of Kirchhoff’s laws; either summing the currents at each node to zero, or by summing the voltages across the elements that form a closed loop to zero.

In theory, it is equally meaningful whether the physical unknown variable at given coordinates points is represented by currents or voltages. However, the voltage analog is favored in practice due to the relative ease with which the voltages across any two points can be measured without disturbing the network configuration. The use of the voltage analog implies that the voltages at each node in the network are the unknowns in the network equations.

The network equations based on the node analysis can be written in matrix form

\[
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2 \\
\vdots \\
v_n
\end{bmatrix}
= 
\begin{bmatrix}
i_1 \\
i_2 \\
\vdots \\
i_n
\end{bmatrix}
\]

or

\[
AV = J
\]

where \( A \) is the admittance matrix, \( J \) is the current vector, and \( V \) is the voltage vector, the solution vector.

B. The Positive Definiteness of the \( A \) Matrix

Closer examination of the coefficient matrix \( A \) reveals that the \( A \) matrix associated with a resistive network is symmetric. It can also be shown that the \( A \) matrix is positive definite. The symmetric form of the \( A \) matrix is the direct consequence of the bilateral characteristic of all passive network elements. The positive definiteness of the \( A \) matrix can be inferred by considering the power dissipation of the excited network [28].

As an example, consider the network below.

\[
\begin{bmatrix}
(y_{12} + y_{13}) & -y_{12} & -y_{13} \\
-y_{12} & (y_{12} + y_{23}) & -y_{23} \\
-y_{13} & -y_{23} & (y_{13} + y_{23})
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2 \\
v_3
\end{bmatrix}
= 
\begin{bmatrix}
i_1 \\
i_2 \\
i_3
\end{bmatrix}
\]

The quadratic form is

\[
V^TAV = y_{12}(v_1 - v_2)^2 + y_{13}(v_1 - v_3)^2 + y_{23}(v_2 - v_3)^2
\]

which represents the power dissipated by the resistance \( R_{12}, R_{13}, \) and \( R_{23} \) under the respective voltage differences. The power dissipation is necessarily positive at all times since the network is passive. Therefore, the quadratic form

\[
V^TAV > 0, \text{ for all } V \neq 0
\]

and thus the coefficient matrix \( A \) is positive definite.

For the general case, consider the following network
where there are \( n \) nodes, all interconnected to each other by \( y_{ij} \). Any open circuits in the network are denoted by \( y_{ij} = 0 \).

The network equation can be written as

\[
\begin{align*}
\text{node 1} & \quad (v_1 - v_2)y_{12} + (v_1 - v_3)y_{13} + \cdots + (v_1 - v_n)y_{1n} = i_1 \\
\text{node 2} & \quad (v_2 - v_1)y_{21} + (v_2 - v_3)y_{23} + \cdots + (v_2 - v_n)y_{2n} = i_2 \\
& \quad \vdots \\
\text{node n} & \quad (v_n - v_1)y_{n1} + (v_n - v_2)y_{n2} + \cdots + (v_n - v_{n-1})y_{nn-1} = i_n.
\end{align*}
\]

These equations can be rewritten as

\[
\sum_{j=1}^{n} a_{ij} y_j = i_i, \quad i = 1, 2, \ldots, n
\]

or in matrix form \( AV = \mathcal{J} \). The coefficient matrix \( A = (a_{ij}) \) is given by

\[
a_{ij} = -y_{ij} = a_{ji}, \quad \text{for} \quad i \neq j
\]

and

\[
a_{ii} = \sum_{j=1}^{n} y_{ij} = \sum_{j=1, j \neq i}^{n} y_{ji}.
\]  \( \text{(1)} \)

Equation (1) expresses \( a_{ii} \) as sums of positive terms. Therefore the matrix \( A \) has all positive diagonal entries.

In order to facilitate the detailed expansion of the quadratic form, the vector \( U = [u_1, u_2, \ldots, u_n]^T \) is introduced, such that

\[
V^TAV = UV, \quad \text{or} \quad U = V^TA.
\]

The component of \( U \) can be expressed as

\[
u_i = \sum_{j=1}^{n} v_j a_{ij}.
\]  \( \text{(2)} \)

Substituting (1) into (2) yields

\[
u_i = -\sum_{j=1}^{n} v_j y_{ij} + \sum_{k \neq i} v_k y_{ik}.
\]

After substituting the expression for \( u_i \), and taking note of the fact that \( y_{ij} = y_{ji} \), the quadratic form can be written as

\[
V^TAV = \sum_{i=1}^{n} u_i v_i
= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (v_i - v_j)^2 y_{ij}.
\]

This equation represents the sum total of the power dissipated by all resistors in the entire network. Therefore \( V^TAV > 0 \), for all \( V \neq 0 \), which proves that the coefficient matrix \( A \) is positive definite for network equations describing a resistive network. The very fact that a finite-difference approximation is derived by the physical approach insures the positive definiteness of the coefficient matrix. The a priori knowledge on the coefficient matrix has significant influence in establishing the convergence of the prospective iterative computing schemes.

In practice, current sources are seldom used. The networks are most frequently energized by voltage sources across feed-in resistors at the independent node points. When voltage sources are used to supply the independent values, the coefficient matrix assumes a form different from the previous cases where current sources are used. Subscript \( b \) is used here to distinguish the feed-in nodes and feed-in resistors from the interior nodes and resistors. From the simple example below, \( v_1, v_2, \) and \( v_3 \) can be identified as the interior node voltages, and \( y_{12}, y_{13}, \) and \( y_{23} \) as the conductances of the interior resistors. The feed-in node voltages and the corresponding conductances are \( v_{b1}, v_{b2}, \) and \( v_{b3} \) and \( y_{1b1}, y_{2b2}, \) and \( y_{3b3} \). It is obvious that a feed-in resistor is always associated with one interior node. Thus the labeling of the boundary voltage points and feed-in resistors is always unique.

The corresponding network equations describing the network with three interior points can be described by

\[
\begin{bmatrix}
(y_{1b1} + y_{12} + y_{13}), & -y_{12}, & -y_{13} \\
-y_{12}, & (y_{2b2} + y_{12} + y_{23}), & -y_{23} \\
-y_{13}, & -y_{23}, & (y_{3b3} + y_{13} + y_{23})
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2 \\
v_3
\end{bmatrix} =
\begin{bmatrix}
y_{1b1}v_{b1} \\
y_{2b2}v_{b2} \\
y_{3b3}v_{b3}
\end{bmatrix}.
\]

The quadratic form can be expressed as

\[
V^TAV = (v_1 - v_2)^2 y_{12} + (v_1 - v_3)^2 y_{13} + (v_2 - v_3)^2 y_{23}
+ y_{1b1}v_1^2 + y_{2b2}v_2^2 + y_{3b3}v_3^2.
\]

Using an approach similar to that for the previous case where boundaries are specified in terms of feed-in electrical currents, the quadratic form for the voltage boundaries can be written in general as the sum total of the power dissipated by all the resistors plus the products of boundary conductances by the squares of the voltages at the corresponding interior nodes.
TABLE I
SOME COMMONLY EMPLOYED ITERATION METHODS

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$H$</th>
<th>Iteration Schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Matrix Expanded Forms</td>
</tr>
<tr>
<td>Jacobi</td>
<td>$H = D^{-1}$</td>
<td>$X^{(k)} = D^{-1} (E + F) X^{(k-1)} + D^{-1} K$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_{ii} x_i^{(k)} = - \sum_{j \neq i} a_{ij} x_j^{(k-1)} + k_i, 1 \leq i \leq n, k \geq 0$</td>
</tr>
<tr>
<td>Gauss-Seidel</td>
<td>$H = (D - E)^{-1}$</td>
<td>$X^{(k)} = (D - E)^{-1} F X^{(k-1)} + (D - E)^{-1} K$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_{ii} x_i^{(k)} = - \sum_{j = 1}^{i-1} a_{ij} x_j^{(k-1)} - \sum_{j = i+1}^{n} a_{ij} x_j^{(k-1)} + k_i, 1 \leq i \leq n, k \geq 0$</td>
</tr>
<tr>
<td>Overrelaxation</td>
<td>$H = \omega (I - \omega L)^{-1} D^{-1}$</td>
<td>$X^{(k)} = (I - \omega L)^{-1} [(I - \omega I) + \omega U] X^{(k-1)} + \omega (I - \omega L)^{-1} D^{-1} K$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$L = D^{-1} E, U = D^{-1} F, 1 \leq \omega \leq 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_{ii} x_i^{(k)} = a_{ii} x_i^{(k-1)} + \omega \left[ - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} + k_i - a_{ii} x_i^{(k-1)} \right]$</td>
</tr>
</tbody>
</table>

That is, the quadratic form is

$$V^T A V = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (v_i - v_j)^2 y_{ij} + \sum_{i=1}^{n} v_i^2 y_{ii}$$

where $n$ is the number of interior nodes.

III. ITERATIVE METHODS [29]

Iterative methods provide a means for connecting the partial solutions into the overall solution and thus permit the treatments of large-scale systems with modest size computing equipment. A review of some of the basic iteration concepts is included here to make the paper self-contained.

A. The Iterative Concept

In solving a system of linear equations by iterative methods, the solution vector evolves as the limit of a sequence of vectors which is constructed by a recursive process. An iterative scheme can thus be viewed as a process by which a sequence of successively improved approximations to the solution is generated. To visualize the basic concept involved in the construction of the vector sequence, let the matrix equation $AX = K$ represent the system of equations whose solution $X$ is sought. A sequence of vectors $X^{(1)}, X^{(2)}, \ldots, X^{(k)}$ is to be defined as the successive approximation which, in principle, converges to the exact solution. If $X^{(k-1)}$ differs from the true solution $X$, then the residue $K - AX^{(k-1)}$ should be used as a "correction" to $X^{(k-1)}$ in order to arrive at a better approximation $X^{(k)}$. The sequence of successive approximations is constructed by the recurrence formulas

$$X^{(k)} = X^{(k-1)} + H \left[ K - AX^{(k-1)} \right]$$

(3) *(improved estimate) = (previous estimate) + (correction)*

where $X^{(0)}$ is the arbitrary initial approximation. The choice of the matrix $H$ defines the iterative process to be specific when $H = I$; the iterative process becomes that of the successive approximation scheme

$$X^{(k)} = BX^{(k-1)} + K$$

where $B = I - A$.

Table I depicts the most commonly used iterative schemes as members of a family of recursive approximations defined by the correction matrix $H$. As a notational convenience, matrix $A$ is expressed as the sum [29]

$$A = D - E - F$$

where $D = \text{diag} (a_{11}, a_{22}, \ldots, a_{nn})$, and $E$ and $F$ are strictly lower and upper triangular $n \times n$ matrices, respectively, such that
$$A = \begin{bmatrix} -F & 0 & 0 \\ D & 0 & 0 \\ -E & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ D & 0 & 0 \\ 0 & E & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

B. Convergence of Iterative Methods

The use of iterative methods presumes that the successive approximate solutions $X^{(k)}$ will approach the true solution $X$ of the equation system. When the approximations approach the true solution within tolerable limits, the iterative process is said to have converged. With arbitrary splitting of the matrix $A$ into $A = N - P$, the iterative schemes for solving $AX = K$ can be written in the form

$$X^{(k)} = MX^{(k-1)} + G$$

where $M = N^{-1}P$ and $G = N^{-1}K$. It can be shown that under suitable conditions the iterative approximations will converge to the true solution [29], namely,

$$\lim_{k \to \infty} [X^{(k)} - X] = \lim_{k \to \infty} e^{(k)} = 0.$$

It is shown in Appendix I that a necessary and sufficient condition for the convergence of $A, (A^n \to 0)$, is that the absolute values of all the eigenvalues $\lambda_i$ of the matrix $A$ be less than unity.

In other words, a matrix $A$ is convergent if $\rho(A) < 1$, where $\rho(A) = \max_i |\lambda_i|$ is the spectral radius of the matrix $A$.

C. Positive Definite Systems

A linear equation system is said to be positive definite if the coefficient matrix $A$ is positive definite. Very often, the large-scale linear systems that arise in practice have real symmetric $A$ matrices which are positive definite. For example, it was shown previously that a linear system describing a resistive network is always positive definite. The finite-difference equation systems approximating the partial differential equations can also be derived to yield positive definite systems.

In the cases where the linear systems are positive definite, it can be shown that a general class of iterative methods converge. Consider again the splitting of the $A$ matrix into $A = N - P$, and define the iteration matrix $M = N^{-1}P$.

With the iteration matrix defined, the convergence of such a general class of iterative methods for positive definite systems can be determined by applying the following theorem. (See Appendix II for proof.)

Theorem: Let $A$ be an $n \times n$ Hermitian (or symmetric) matrix, and $N$ be an $n \times n$ nonsingular matrix such that together with $N^*$, the conjugate transpose of $N$, $Q = N + N^* - A$ is positive definite. Then the iteration matrix $M$ is convergent if, and only if, the matrix $A$ is positive definite.

As an example to the application of the above theorem, it can be shown that for the case of symmetric $A$ matrix with positive diagonal elements, the Gauss–Seidel method is convergent, if and only if $A$ is positive definite. To verify the above statement, write as before, $A = D - E - F = D - E - E^*$, since $A$ is Hermitian. Now recall that the splitting of the $A$ matrix corresponding to the Gauss–Seidel method is $N = D - E$ and $P = E^*$, and thus,

$$Q = N + N^* - A = D$$

which is positive definite.

Therefore, if $A$ is positive definite, the iteration matrix $M = (D - E)^{-1}F$ is convergent, and the Gauss–Seidel method converges.

IV. HYBRID COMPUTING TECHNIQUES

Hybrid computing techniques are particularly suited for iterative computations. The analog devices perform the repetitive computations that must be carried out for every iterative cycle, while the digital computer provides the necessary data storage and automatic control on the course of computation through the use of stored programs.

The concept of seeking the solution to a large system of equations through partitioning and iterative computations can be best described by example. In the following figure, a large physical system is represented by the heavy lines and the external excitations by the arrows.

The dotted lines indicate the boundaries of the partitioned systems and the coupling between sections is indicated by the solid lines. In the above system, it is assumed that at most one fourth of the problem can be handled at a time. The digital computer would set the analog equipment to simulate Section I. The I–II and I–III boundaries can be initially set to zero or some other convenient values. After the equations describing Section I are solved, control is transferred to set up the analog equipment to simulate Section II. The boundary conditions are applied but with the newly computed Section I values. The procedure is repeated for Sections III and IV, each time using the currently computed boundary values.

The entire process is repeated until all boundary conditions are within a predetermined tolerance. Through this iterative model method, large systems of equations can be solved on relatively small computing facilities.

A. Implementation of the Iterative Network Method

In order to take full advantage of the resistive network analogy, rules of implementing the iterative scheme are de-
developed to drastically reduce all arithmetic operations and thus substantially simplify the programming tasks. The iterative method can then be carried out by storing the voltage values at the partition during the course of the iterative cycles.

Consider a simple resistive network with six interior nodes 1 through 6 and known voltages \( v_0 \) and \( v_7 \) as shown below.

![Network Diagram]

The network equations can be written as

\[
\begin{bmatrix}
2 & -1 \\
-1 & 2 & -1 \\
& -1 & 2 & -1 \\
& & -1 & 2 & -1 \\
& & & -1 & 2
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2 \\
v_3 \\
v_4 \\
v_5 \\
v_6 \\
v_7
\end{bmatrix}
= \begin{bmatrix}
v_0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

(4a)

The second quadrant is represented by the solution to the matrix equation,

\[
\begin{bmatrix}
2 & -1 & 0 \\
-1 & 2 & -1 \\
0 & -1 & 2
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2 \\
v_3 \\
v_4 \\
v_5 \\
v_6 \\
v_7
\end{bmatrix}
= \begin{bmatrix}
v_0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

(4b)

which is the equation describing the network below.

![Network Diagram]

Similarly, consider that the submatrix in the fourth quadrant leads to the equation

\[
\begin{bmatrix}
2 & -1 & 0 \\
-1 & 2 & -1 \\
0 & -1 & 2
\end{bmatrix}
\begin{bmatrix}
v_4 \\
v_5 \\
v_6 \\
v_7
\end{bmatrix}
= \begin{bmatrix}
v_3 \\
0 \\
0 \\
0
\end{bmatrix}
\]

(4c)

and that the corresponding network consists of nodes 3 through 7 as shown.

![Network Diagram]
The solution of (4b) and (4c) actually carries out the recursive relationship for one complete iterative cycle.

The resistive network yields solutions to (4b) and (4c) by merely reapplying the adjacent boundary voltages \( v_4 \) and \( v_3 \) to the respective partitions. In this simple example, the two partitions are represented by the nodes 1, 2, 3, and 4, 5, 6.

The preceding illustration can readily be extended to cover the two-dimensional network grids. Consider the rectangular network grid shown in Fig. 1. The corresponding network equations are also included in the same figure.

If the network is divided into two partitions, as shown by the dotted lines, the network partitions as well as their respective network equations are given in Fig. 2.

It can be observed that the right-hand side or the forcing functions of the partitioned matrix equations are the boundary voltages of the adjacent network partition.

The partitioning of the preceding example is by no means unique. The matrix equation can be partitioned into more than two sections, and furthermore, the partitioned matrices do not have to be of the same order. The only requirement is that the partitioning must produce square submatrices. Another permissible way of partitioning is illustrated in Fig. 3.

However, partitioning that produces submatrices of the same order facilitates the programming efforts. The use of a smaller number of partitions can also improve the overall solution rates.

The flow chart for implementing the iterative scheme is shown in Fig. 4. The implementation example was simulated on a digital computer using elimination in place of the resistive network, and the solution vector \( X \) at the \( k \)th iteration is shown in Table II. Fig. 5 shows the sum of squared residue

\[
r_s = (AX - K)^T(AX - K)
\]

at the \( k \)th iterative cycles for different partition sizes.

The hybrid mechanization of the iterative method is shown in Fig. 6. The accuracy of the analog device requires the digital computer word length to be no more than 16 bits. Together with the general lack of arithmetic operations in the iterative method, the digital computer requirements can be satisfied by a modest controller type computer with either \( 4K \) or \( 8K \) of 16-bit words memory. The capacity of the analog equipment is largely constrained by the hardware costs and reliability considerations. Resistive networks with 250 to 500 nodes can be implemented with acceptable cost and reliability level while maintaining the advantage of...
TABLE II
RESISTIVE NETWORK SOLUTION VECTOR

<table>
<thead>
<tr>
<th>i/k</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>7</th>
<th>11</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.45604</td>
<td>0.52874</td>
<td>0.57730</td>
<td>0.59795</td>
<td>0.60177</td>
<td>0.60253</td>
<td>0.60255</td>
</tr>
<tr>
<td>2</td>
<td>0.39500</td>
<td>0.48332</td>
<td>0.56300</td>
<td>0.60000</td>
<td>0.60689</td>
<td>0.60826</td>
<td>0.60829</td>
</tr>
<tr>
<td>3</td>
<td>0.36813</td>
<td>0.45321</td>
<td>0.54632</td>
<td>0.59226</td>
<td>0.60085</td>
<td>0.60256</td>
<td>0.60260</td>
</tr>
<tr>
<td>4</td>
<td>0.42857</td>
<td>0.63164</td>
<td>0.74621</td>
<td>0.79181</td>
<td>0.80022</td>
<td>0.80189</td>
<td>0.80193</td>
</tr>
<tr>
<td>5</td>
<td>0.25824</td>
<td>0.45132</td>
<td>0.62837</td>
<td>0.70978</td>
<td>0.72493</td>
<td>0.72794</td>
<td>0.72802</td>
</tr>
<tr>
<td>6</td>
<td>0.20879</td>
<td>0.37632</td>
<td>0.57958</td>
<td>0.67679</td>
<td>0.69566</td>
<td>0.69942</td>
<td>0.69452</td>
</tr>
<tr>
<td>7</td>
<td>0.54649</td>
<td>0.70385</td>
<td>0.82141</td>
<td>0.86734</td>
<td>0.87554</td>
<td>0.87717</td>
<td>0.87721</td>
</tr>
<tr>
<td>8</td>
<td>0.31403</td>
<td>0.50676</td>
<td>0.70223</td>
<td>0.78462</td>
<td>0.79940</td>
<td>0.80233</td>
<td>0.80240</td>
</tr>
<tr>
<td>9</td>
<td>0.22442</td>
<td>0.41290</td>
<td>0.64348</td>
<td>0.78586</td>
<td>0.76429</td>
<td>0.76794</td>
<td>0.76803</td>
</tr>
<tr>
<td>10</td>
<td>0.44336</td>
<td>0.67701</td>
<td>0.83721</td>
<td>0.89294</td>
<td>0.90257</td>
<td>0.90446</td>
<td>0.90451</td>
</tr>
<tr>
<td>11</td>
<td>0.22696</td>
<td>0.45897</td>
<td>0.71566</td>
<td>0.81550</td>
<td>0.83284</td>
<td>0.83626</td>
<td>0.83634</td>
</tr>
<tr>
<td>12</td>
<td>0.15046</td>
<td>0.35561</td>
<td>0.62223</td>
<td>0.77618</td>
<td>0.79780</td>
<td>0.80206</td>
<td>0.80216</td>
</tr>
<tr>
<td>13</td>
<td>0.54522</td>
<td>0.73044</td>
<td>0.85365</td>
<td>0.89588</td>
<td>0.90307</td>
<td>0.90445</td>
<td>0.90452</td>
</tr>
<tr>
<td>14</td>
<td>0.29650</td>
<td>0.53624</td>
<td>0.74493</td>
<td>0.82079</td>
<td>0.83376</td>
<td>0.83631</td>
<td>0.83637</td>
</tr>
<tr>
<td>15</td>
<td>0.19497</td>
<td>0.43902</td>
<td>0.68839</td>
<td>0.78277</td>
<td>0.79894</td>
<td>0.80212</td>
<td>0.80220</td>
</tr>
<tr>
<td>16</td>
<td>0.44102</td>
<td>0.70852</td>
<td>0.83248</td>
<td>0.86978</td>
<td>0.87598</td>
<td>0.87719</td>
<td>0.87722</td>
</tr>
<tr>
<td>17</td>
<td>0.21886</td>
<td>0.51652</td>
<td>0.72202</td>
<td>0.78902</td>
<td>0.80018</td>
<td>0.80237</td>
<td>0.80242</td>
</tr>
<tr>
<td>18</td>
<td>0.13794</td>
<td>0.42520</td>
<td>0.66801</td>
<td>0.75134</td>
<td>0.76526</td>
<td>0.76799</td>
<td>0.76806</td>
</tr>
<tr>
<td>19</td>
<td>0.58713</td>
<td>0.70856</td>
<td>0.77634</td>
<td>0.79770</td>
<td>0.80125</td>
<td>0.80195</td>
<td>0.80196</td>
</tr>
<tr>
<td>20</td>
<td>0.39612</td>
<td>0.56501</td>
<td>0.68197</td>
<td>0.72038</td>
<td>0.72679</td>
<td>0.72804</td>
<td>0.72807</td>
</tr>
<tr>
<td>21</td>
<td>0.32008</td>
<td>0.50620</td>
<td>0.64216</td>
<td>0.69000</td>
<td>0.69798</td>
<td>0.69955</td>
<td>0.69959</td>
</tr>
<tr>
<td>22</td>
<td>0.51138</td>
<td>0.56073</td>
<td>0.59092</td>
<td>0.60063</td>
<td>0.60224</td>
<td>0.60256</td>
<td>0.60257</td>
</tr>
<tr>
<td>23</td>
<td>0.45842</td>
<td>0.53435</td>
<td>0.58735</td>
<td>0.60482</td>
<td>0.60773</td>
<td>0.60830</td>
<td>0.60832</td>
</tr>
<tr>
<td>24</td>
<td>0.42616</td>
<td>0.51160</td>
<td>0.57650</td>
<td>0.59827</td>
<td>0.60190</td>
<td>0.60262</td>
<td>0.60263</td>
</tr>
</tbody>
</table>

Fig. 4. Flow chart for iterative network solution.
supplying the engineer with intuitive feel for the problem on hand.

V. CONCLUSIONS

The DSDT approach represents only one of the useful techniques for solving problems described by partial differential equations.

The resistive network, as a mechanization of the DSDT method, is conceptually simple and inexpensive to implement. Furthermore, with suitable arrangements of the network topology, the resistive network solutions can provide the engineer with added intuitive feel that is especially useful in problems involving variable boundaries, or where boundaries are to be determined for either optimization or model building purposes.

The study of the iterative methods here is motivated by an attempt to improve the accuracy and resolution of the DSDT resistive networks and to treat large-scale systems.

The iterative schemes involve very simple algorithms and as such the hybrid computing techniques are not indispensable for the working of the iterative methods. However, the hybrid computations will substantially improve the solution rates and reduce operator errors especially in an environment where the operations are largely repetitive.

APPENDIX I

It was stated that a matrix $A$ is convergent if and only if the absolute values of all the eigenvalues of $A$ are less than unity.

This criterion can be proved by first bringing the matrix $A$ into a diagonal form by a similarity transformation $C$, obtaining

$$A = CAC^{-1}$$

where

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ & \lambda_2 \\ & & \ddots \\ & & & \lambda_n \end{bmatrix}$$

and $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of the matrix $A$. Then, by the associative property of matrix multiplication

$$A^n = (CAC^{-1})(CAC^{-1}) \cdots (CAC^{-1}) = CAC^{n-1}$$

and the $m$th power of the diagonal matrix is

$$\Lambda^m = \begin{bmatrix} \lambda_1^m \\ & \lambda_2^m \\ & & \ddots \\ & & & \lambda_n^m \end{bmatrix}$$

In order to have $\Lambda^m \to 0$, it is necessary and sufficient that $\Lambda^m \to 0$, which requires that all eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ must have modulus less than unity, or $\rho(A) < 1$.

If the matrix $A$ cannot be brought into diagonal form, the proof can be similarly carried out by bringing the $A$ matrix into the Jordan canonical form [30].

APPENDIX II

It will be shown that if matrix $A$ is Hermitian and $N$ is a nonsingular matrix of the same order as $A$ such that $Q = N + N^* - A$ is positive definite, then the iteration matrix $M = N^{-1}P$ is convergent if, and only if, $A$ is positive definite. In order to show that the iterative method is convergent, or that the $M$ matrix is convergent, it is both necessary and sufficient that $\rho(M) < 1$, where $\rho(M)$ is the largest eigenvalue of the matrix $M$, i.e.,

$$\rho(M) = \max_{i} |\lambda_i|, \quad 1 \leq i \leq n.$$ 

Since $A = N - P$ and $M = N^{-1}P$, the matrix $A$ can be expressed in terms of $N$ and $M$ by eliminating $P$ such that

$$A = N - P = N - NM = N(I - M). \quad (5)$$
Let \( \lambda \) be any eigenvalue of \( M \), and let \( u \) be the corresponding eigenvector. The eigenvalue and the eigenvector satisfy the relationship

\[
Mu = \lambda u. \tag{6}
\]

Combining (5) and (6) and forming the product \( Au \), there results

\[
Au = (1 - \lambda)Nu. \tag{7}
\]

The conjugate transpose of the above is

\[
u^*A = (1 - \lambda)u^*N\tag{8}
\]

Premultiplying (7) by \( u^* \) and postmultiplying (8) by \( u \), the quadratic forms

\[
u^*Au = (1 - \lambda)u^*Nu
\]

\[
u^*Au = (1 - \lambda)u^*N*u \tag{9}
\]

result since \( A \) is Hermitian.

Let \( \lambda \) be represented by its real and imaginary components \( \alpha \) and \( i\beta \), such that

\[
\lambda = \alpha + i\beta
\]

\[
\bar{\lambda} = \alpha - i\beta. \tag{10}
\]

After substituting \( \lambda \) and \( \bar{\lambda} \) by (10), then (9) can be rearranged

\[
\frac{1}{(1 - \alpha) - i\beta} = \frac{u^*Nu}{u^*Au}
\]

\[
\frac{1}{(1 - \alpha) + i\beta} = \frac{u^*N*u}{u^*Au}.
\]

Adding these two equations produces

\[
\frac{2(1 - \alpha)}{(1 - \alpha)^2 + \beta^2} = \frac{u^*(N + N^*)u}{u^*Au} = 1 + \frac{u^*Qu}{u^*Au}. \tag{11}
\]

Since \( Q \) is positive definite, it implies that

\[
\frac{u^*Qu}{u^*Au} > 0
\]

only if \( A \) is positive definite. Let \( A \) be positive definite, then

\[
1 + \frac{u^*Qu}{u^*Au} > 1
\]

and the left side of (11) becomes

\[
2(1 - \alpha) > (1 - \alpha)^2 + \beta^2 = 1 - 2\alpha + \alpha^2 + \beta^2.
\]

After cancelling out the \( \alpha \) terms from the above equation, the magnitude squared term can be written as

\[
|\lambda|^2 = \alpha^2 + \beta^2 < 1
\]

or

\[
|\lambda| < 1.
\]

Since \( \lambda \) is any eigenvalue of \( M \), then \( \rho(M) < 1 \), the matrix \( M \), is therefore convergent.

---

**References**


[26] ——, “Hybrid computer simulation of ground water basins,” 1967
Detection and Parallel Execution of Independent Instructions

GAROLD S. TJADEN AND MICHAEL J. FLYNN, MEMBER, IEEE

Abstract—For a single instruction stream—single data stream organization the problem of simultaneously issuing several instructions is studied.

Within a block of $N$ instructions, there exist dependencies which must be detected and ordered so that the maximum number of instructions can be issued for execution. These dependencies include memory accessing, data registers (sink-source), operand availability, and procedural dependency (conditional branch). A simple decoding mechanism is proposed to analyze these dependencies and provide maximum instruction issuance.

A simulator has been written to provide an evaluation of the aforementioned schemes using actual programs and formats of a representative computer (IBM 7094).

The results of this simulation indicate that an 86 percent performance improvement could be realized for the appropriately modified 7094, without use of compiler techniques or programmer assistance.

Index Terms—Executable independent instructions, instruction decoding, predecode stack, stack simulator, weakly independent instructions.

INTRODUCTION

A key problem in the design of high-speed, single instruction stream—single data stream computers is the limitation of serial issuing of instructions.

While instructions may be executed simultaneously or out of sequence, depending on the availability of resources, they are decoded serially in strict sequence (see Anderson [11], Tomasulo [12], and Flynn [8]). Indeed, while several computers (IBM System/360 Models 85, 91, 195, and CDC 6600 and 7600) decode at a maximum serial rate, i.e., one instruction decoded per machine cycle, no system attempts a multiple simultaneous decoding of blocks of instructions. The problem has been to organize the instruction format and the decoder to readily ascertain the instructions which qualify for simultaneous execution. The conditions for this "qualification" must be simply implemented as the cost to simultaneously decode $n$ instructions will rise exponentially with $n$. Coupled with this is a degeneration in decode time directly related to $n$.

This paper will present a simple algorithm for simultaneously decoding instructions and will evaluate the algorithm on representative existing code.

Thus, out of a block of $n$ instructions, the $m$ independent instructions are immediately executed (as resources are available). Upon their execution, the remaining instructions, together with the next $m$ instructions, are decoded for independency and those qualified proceed to execution.

In actual machine environment, the detection and issuing of instructions will overlap execution. The algorithm presented here will first discuss the nonoverlap case and then the relatively straightforward extension to the overlapped case.

INSTRUCTIONS

An instruction defines an ordered pair $(x_1, x_2)$ of source operands, a uniform binary procedure OP to be performed on the source operands whose result range is a single operand $f(x_1, x_2)=x_3OPx_2$, a specification of the location of the result (sink), and a specification of the location of the next instruction.

We take care here not to make the definition too broad. If we allow $n$-ary operands or nonuniform procedures, we will be dealing with programs or program segments rather than individual instructions. We add an additional restriction. The specification of the source operands, the sink result, and the next instruction must be made by defining its location in storage. This location is called the effective address. Only the following procedures are considered allowable in generating an effective address.