A SPECIAL-PURPOSE DATA DRIVEN MULTIPROCESSOR FOR ASYNCHRONOUS PARALLEL NEWTON'S ALGORITHMS

Shohei Fujita
Hiroshi Kasai
Toshinobu Suzuki
Takeshi Fukao

Department of Computer Science
Tokyo Institute of Technology
O-okayama 2-12-1, Meguro-ku, Tokyo 152

ABSTRACT: A class of asynchronous parallel Newton's algorithms: Purely-Asynchronous (PAN), Asynchronous (AN), and Semi-Asynchronous (SAN) algorithms are introduced for the semantics of data driven approach. A sufficient condition is presented to guarantee local convergence of these asynchronous algorithms. Furthermore, it is shown that the PAN algorithm is linearly convergent, the AN algorithm is superlinearly convergent, and the SAN algorithm is quadratically convergent. The order of convergence depends on the degree of interaction between two processing elements. It is possible to produce an algorithm which has an advantage of quadratic convergence and a factor of speed-up by asynchronous iteration. This is realized on special-purpose data driven multiprocessor. An asynchronous parallel Newton's program implementation based on the concept of activity templates is presented; and a basic architecture used in our experimental construction of the special-purpose processor is discussed.

KEY WORDS: MIMD, data driven, system of nonlinear equations, asynchronous parallel Newton's algorithms, degree of interaction, linear convergence, superlinear convergence, quadratic convergence, activity template, special-purpose processor.

1. INTRODUCTION

With advances in the VLSI technology, projecting these development on computer architecture and algorithm has a very interesting impact on the large-scale scientific computing problems (Fig. 1.1). It would be quite practicable to focus just on special-purpose machine for concurrent scientific numerical computations and to think of the machine as being an attachment to a general-purpose computing systems.

This paper presents a special-purpose data driven multiprocessor for solving system of nonlinear algebraic equations:

"Given a mapping $f: R^n \to R^n$, find a point $\xi \in R^n$ such that

$$f(\xi) = 0." \qquad (1.1)$$

Often f is smooth, i.e., $f \in C^1(R^n)$, as we shall henceforth assume.

In Section 2 we introduce a class of asynchronous parallel Newton's Algorithms: Purely-Asynchronous (PAN), Asynchronous (AN), and Semi-Asynchronous (SAN) algorithms, which are implemented on data driven multiprocessor system.* Section 3 gives a convergence condition for three asynchronous algorithms (Theorem 1). Furthermore, we show that the PAN method is linearly convergent, the AN method is superlinearly convergent, and the SAN method is quadratically convergent (Theorem 2). The order of convergence depends on the degree of interaction between two processing elements. It is proved that each processing element should keep iteration by using the most recent data and transfer immediately its computed results to other processing elements (Theorem 3). Hence, it is possible to produce an asynchronous algorithm which has an advantage of quadratic convergence and is superior to the synchronized counterpart with respect to overall computation time. This is realized on data driven multiprocessor system. Data driven program implementation using activity templates is shown in Section 4. Construction of the special-purpose data driven multiprocessor is discussed in Section 5.

* In Traub [15] an algorithm was introduced for parallel implementation of Newton's method on multiprocessor system; and a methodology was developed to analyze speed-up for parallel algorithm to nonlinear algebraic equation. In Kung [10] an asynchronous algorithm consisting of parallel processes which are not synchronized at any time was discussed; and a possibilty of the superiority in speed of some asynchronous iterative algorithm over the synchronized counterpart was shown. A specially designated asynchronous parallel Newton's method is analyzed in Baudet [2]. All of them, however, are based on the "shared memory" multiprocessors such as C.mmp, which have problems with physical address conflict during memory access.

2. NEW ARCHITECTURE - NEW ALGORITHM: ASYNCHRONOUS PARALLEL NEWTON'S ALGORITHMS

In 1966 Karp and Miller [9] first proposed a concept of data flow. Since then several data flow architectures have been presented. The fundamental performance bounds imposed by the von Neuman model can be achieved only with significant change to the "clock driven" von Neuman architecture. In attempting to overcome these fundamental difficulties, there are two approaches:

- "demand driven" approach,

and

- "data driven" approach

which have some promise (e.g., [3]).

The approach espoused here is of the data driven due to the feeling that the data driven approach is well suitable for both

- "horizontal" concurrency (parallel processing)

and

- "vertical" concurrency (pipeline processing)

which exist in the Newton's method.

The principle of data driven approach are (e.g., [8], [16]):

(1) Asynchrony (Activation by availability): An operation is executed when and only when all required input data values become available.

(2) Decentralization (Functionality): An operation is functional and has no side-effect. When an operation is executed, the input data values are used up, i.e., they are no longer available as input to any other operations. A complete set of results is always produced if the operation terminate. The computation with the operation has no effect on other computation (except to compete for resource).

Performance of concurrent system is, in general, influenced intensively how to decompose a program into subtasks. General procedure that identify program structures which is implementable concurrently would be difficult problem. Even if this is not so difficult (e.g, [1], [6], [7]), we cannot directly apply our knowledge of the conventional serial algorithms to concurrent system because efficient serial algorithms do not necessarily lead to efficient parallel algorithms. Moreover, the suitability of an algorithm for SIMD system is essentially uncorrelated or perhaps negatively correlated to its suitability for MIMD system. Similarly, it is worthwhile to study algorithms for MIMD systems without necessarily commiting to one particular style system.

The purpose of this section is to present a new class of asynchronous parallel algorithms to Eq. (1.1), which are implemented on MIMD systems, especially data driven processor with suitability for the asynchronous and functional semantics of data driven architecture. The main characteristics of the asynchronous algorithms presented here is that its processing elements never idle for input data at any time.

The Class of Asynchronous Newton's Algorithms:

The algorithm of Newton's method to find a solution of (1.1) is given by

$$x_{k+1} = x_k - J(x_k)^{-1} f(x_k) \qquad k = 0,1,\ldots \quad (2.1.a)$$

$$x_0 \in D_\varepsilon = \{ x \mid \| x - \xi \| < \varepsilon \}, \quad (2.1.b)$$

where $J(x_k)$ denotes the Jacobian matrix of f at x_k.

To speed up the computation of the iterative algorithm(2.1), we assign two processing elements P_1 and P_2 such that

- A processing element P_1 executes the evaluation of f and computation of x_{k+1}, i.e.,

$$J \Delta x_k = -f, \; x_{k+1} = x_k + \Delta x_k,$$

- A processing element P_2 executes the evaluation of J,

which are implemented concurrently.

Remark 2.1: The cost of evaluation of the Jacobian matrix J will be more expensive than that of f. Moreover, the LU factorization of the Jacobian matrix in solving the linear system is one of the major cost at each iteration. The effect of changing the Jacobian matrix of nonlinear algebraic equations arising in solving stiff ODEs problems and/or NLP problems is a crucial issue for the efficiency of the codes.

Remark 2.2: It is important to exploit concurrency both at the operation level and at the procedure level. In this paper we concentrate on the latter. In Jacobian code many independent tasks can be performed concurrently. There exists concurrency in solution code for the linear system. These are discussed in more detail elsewhere.

A reasonable asynchronous parallel iterative algorithm consisting of two processing elements P_1 and P_2 can be defined as follows:

$$x_{k+1} = x_k - J(y_k)^{-1} f(x_k), \quad k = 0,1,2,\ldots \quad (2.2.a)$$

$$x_0 \in D_\varepsilon. \quad (2.2.b)$$

Hence the asynchronous parallel algorithm (2.2) is classified by the relative properties between the two processing elements P_1 and P_2.

$$P_1 \longmapsto x_y$$

$$P_2 \longmapsto y_k$$

(1) Purely-Asynchronous Parallel Newton's (PAN) Method:
The choice of y_k is completly arbitrary as long as the following constraint is satisfied:

$$y_k \in D_\varepsilon, \quad k = 0,1,2,\ldots \quad (2.3)$$

(2) Asynchronous Parallel Newton's (AN) Method:
We demand the following condition on y_k:

$$y_k \in E_k \text{ and } \lim_{k\to\infty} \eta_k = 0, \qquad (2.4.a)$$

where

$$E_k \triangleq \{y_k \in D_\varepsilon \mid \| y_k - x_k \| + \| Y_k - \xi \| \leq \eta_k\}, \qquad (2.4.b)$$

that is, E_k is ellipse with foci x_k and ξ, and radius η_k. This procedure is a natural one. In fact, the procedure (2.4) is satisfied provided the sequence $\{y_k\}$ will be chosen from among the prior iterates.

(3) Semi-Asynchronous Parallel Newton's (SAN) Method:
We may impose the condition:

$$y_k = \lambda_k x_k + (1-\lambda_k)\xi, \quad 0 \leq \lambda_k \leq 1, \qquad (2.5)$$

which will be useful provided we have a priori knowledge of the line joining x_k to ξ. In case with no information concerning ξ at all, we are forced to choose $\lambda_k = 1$; and the algorithm (2.2) will be reduced to a synchronized parallel Newton's Method.

Remark 2.3: With the advent of new computer architecture such as data driven computing, we put emphasis on not saving labor of computation in the Newton's method (2.1) but speed-up in the Newton's method (2.1) through asynchronous iterative algorithm (2.2). We are no longer interested in the conventional computational complexity of an algorithm. It may be preferable to perform more operations to obtain a faster and more reliable solution.

Since the asynchronous iterative algorithms implemented on MIMD systems is different from the conventional Newton's method (2.1) implemented on SISD systems, it is of interest to investigate the performance of the asynchronous algorithms:

- to find (local) convergence condition for the asynchronous algorithms,
- to select the optimal value for y_k.

3. CONVERGENCE AND PERFORMANCE

In this section, we show a sufficient condition for local convergence of the asynchronous parallel Newton's methods, and how the orders of convergence of three asynchronous algorithms change according to the relative relationship between the two processing elements P_1 and P_2.

Theorem 1: Suppose there exist constants $\varepsilon > 0$ and $M > 0$ such that

(H-1) $x_0 \in D_\varepsilon \triangleq \{x \mid \| x - \xi \| < \varepsilon\}$

(H-2) $\| J(x) - J(y) \| \leq M \| x - y \|$, for all $x, y \in D_\varepsilon$

(H-3) $M \| J(\xi)^{-1} \| \varepsilon < 2/5$.

Then, the sequence $\{x_k\}$ defined by the Purely-Asynchronous Parallel Newton's (PAN) Method is well-defined, remains in D_ε and converges to the solution ξ of (1.1).

Proof of Theorem 1: By the hypotheses (H-1) and (H-2), it follows that

$$\| J(y_k) - J(\xi) \| < M\varepsilon, \quad \text{for } y_k \in D_\varepsilon. \qquad (3.1)$$

Hence, in view of the hypothesis (H-3) and by virture of Banach Lemma (e.g., Ortega et al.[12], $J(y_k)$ is invertible for all $y_k \in D_\varepsilon$; and we have

$$\| J(y_k)^{-1} \| \leq \frac{\| J(\xi)^{-1} \|}{1 - M \| J(\xi)^{-1} \| \varepsilon} < \infty \qquad (3.2)$$

which means that the sequence $\{x_k\}$ defined by (3.2) is well-defined for all $y_k \in D_\varepsilon$.

From (2.2), (H-2), and (3.2), it follows that

$$\| x_{k+1} - \xi \| \leq \frac{M\delta}{2(1-M\delta\varepsilon)} (\| y_k - x_k \| + \| y_k - \xi \|) \| x_k - \xi \| \qquad (3.3)$$

where

$$\delta \triangleq \| J(\xi)^{-1} \|. \qquad (3.4)$$

For $x_0 \in D_\varepsilon$, we have

$$\| y_0 - x_0 \| + \| y_0 - \xi \| < 3\varepsilon \qquad (3.5)$$

so that together with (3.3)

$$\| x_1 - \xi \| \leq \alpha \| x_0 - \xi \| \qquad (3.6)$$

where

$$\alpha \triangleq 3M\delta\varepsilon \,/\, 2(1 - M\delta\varepsilon) < 1 \qquad (3.7)$$

by the hypothesis (H-3). Thus, $x_1 \in D_\varepsilon$. Assuming $x_k \in D_\varepsilon$ we have

$$\| y_k - x_k \| + \| y_k - \xi \| < 3\varepsilon, \quad \text{for } x_k, y_k \in D_\varepsilon, \qquad (3.8)$$

so that (3.3) and (3.8) gives

$$\| x_{k+1} - \xi \| \leq \alpha \| x_k - \xi \|. \qquad (3.9)$$

Since $\alpha < 1$, it is proved by mathematical induction that $x_k \in D_\varepsilon$ for all k. Therefore, the sequence $\{x_k\}$ generated by the PAN method remains in D_ε and converges the solution ξ of (1.1). This completes the proof of Theorem 1.

Remark 3.1: The hypothesis (H-1) means that the initial approximation x_0 must be close to the solution ξ of (1.1). If the hypothesis (H-2) is satisfied, the Jacobian J of f is said to be Lipschitz continuous and M is called the Lipschitz constant for J. In order to the hypothesis (H-3) hold, the solution ξ of (1.1) must be simple.

Theorem 2: Under the hypotheses of Theorem 1.
(1) Purely-Asynchronous Parallel Newton's (PAN) Method is linearly convergent,

(2) Asynchronous Parallel Newton's (AN) Method is superlinearly convergent, and

(3) Semi-Asynchronous Parallel Newton's(SAN) Method is quadratically convergent.

Proof of Theorem 2:

(1) Linear convergence of the PAN method follows from (3.9) with (3.7).

(2) In view of (3.3), we have

$$\| x_{k+1} - \xi \| \leq \beta \eta_k \| x_k - \xi \|, \quad \text{for all } y_k \in E_k, \tag{3.10}$$

where

$$\beta \triangleq M\delta/2(1 - M\delta\varepsilon). \tag{3.11}$$

Since $\lim_{k\to\infty} \eta_k = 0$, (3.10) proves superlinear convergence of the AN method.

(3) If we choose the sequence $\{y_k\}$ satisfying

$$y_k = \lambda_k x_k + (1 - \lambda_k)\xi, \quad 0 \leq \lambda_k \leq 1, \tag{3.12}$$

it follows from (3.3) that

$$\| x_{k+1} - \xi \| \leq \beta \| x_k - \xi \|^2. \tag{3.13}$$

This shows quadratic convergence of the SAN method.

Remark 3.2: Theorem 2 states that the order of convergence depends on the degree of interaction between the two processing elements. The PAN method has an obvious advantage for implementing in asynchronous manner on MIMD system; but it is linearly convergent. On the other hand, the SAN method converges quadratically but it requires a considerable a priori knowledge on ξ. In practice, the AN method will be most recommended; and we have to construct high-performance MIMD system so that the AN method exceeds the synchronized parallel Newton's method in overall computation time.

To summerize: in view of (3.3) we have

Theorem 3:

- In processing element P_2 it is most desirable to choose the most recent values of the processing element P_1 for y_k.

- If the sequence $\{y_k\}$ in the processing element P_2 is sufficiently close to the sequence $\{x_k\}$ in the processing element P_1, then the order of convergence will be nearly equal to quadratic.

4. DATA DRIVEN PROGRAM IMPLEMENTATION

To illustrate the basic instruction handling mechanism of data driven computing for the asynchronous parallel Newton's methods (the procedures in Theorem 3), it is useful to follow Dennis[4]. In his scheme, a data driven program is a collection of activity templates. An activity template corresponding to the procedure JACOBIAN is shown in Fig. 4.1. There are three kinds of fields for

(1) an operation code specifying the operation to be performed,
(2) n receivers which are places waiting to be filled in with operand values,
(3) destinations which specify what is to be done with the results of performing the operation on the operands.

Fig. 4.2 shows how activity templates are connected to represent the asynchronous parallel Newton's program implementation.

Remark 4.1: To faithfully represent the pipelining operation in data driven program implementation in Fig. 4.2, the linear instruction, for example, must not reactivated until its previous result has been used by the add instruction. This mechanism is realized by using acknowledge signals. This is omitted for clarity in Fig. 4.2.

5. SPECIAL-PURPOSE DATA DRIVEN MULTIPROCESSOR

The basic architectures used in our experimental construction of the data driven multiprocessor are similar to those proposed by Dennis et al. [5]. Our experimental machine has two processing elements because the degree of concurrency in which we are interested in the asynchronous parallel Newton's program is two. The data flow program describing the asynchronous parallel Newton's algorithms is held as a collection of activity templates in the Activity Store. Each activity template has a unique address which is entered in the Instruction Queue unit (FIFO buffer store) when the instruction is ready for execution. The Instruction Queue (FIFO) and 2×2 router can be realized by using the technique of self-timed system (e.g., Seitz [13]). Since the self-timed system has a very different disciplines in the conventional synchronous system, careful design techniques are required to ensure that they support high performance of the data driven multiprocessor. This will be discussed in more details elsewhere.

6. CONCLUSIONS

The purpose of this paper is not to show that the special-purpose data driven multiprocessor is faster in any absolute sense, but rather to present an answer to a basic question about synchronous vs. asynchronous operations which is a basic principle of data driven architecture. New results obtained in this paper are:

(1) We have introduced the class of asynchronous parallel Newton's algorithms: PAN, AN, and SAN algorithms which are implemented on data driven multiprocessor,

(2) Convergence for these asynchronous algorithms has been proved (Theorem 1). Furthermore, we have shown that how the order of convergence depends on the degree of interaction between two processing elements (Theorem 2). It is concluded that in processing element P_2 it is most desirable to choose the most recent value of the processing element P_1. As the sequence $\{y_k\}$ in the processing element P_2 becomes sufficiently close to the sequence $\{x_k\}$ in the processing element P_1, the order of convergence will be nearly equal to quadratic (Theorem 3). Hence, it is possible to produce an algorithm which has characteristics of quadratic convergence and speed-up by asynchronous implementation.

(3) An asynchronous parallel Newton's program implementation based on the concept of activity templates has been presented; and the basic architectures used in our experimental construction of the special-purpose data driven

multiprocessor have been discussed.

We believe that the asynchronous algorithm and the special-purpose data driven processor will be very useful as well in solving stiff ODEs problems and NLP problems.

Remark 6.1: Since data flow environment is fundamentally different in almost every way from the existing methods for solving problems, much remain to be done, in which we are interested

- to find new concurrent algorithms more suitable for data flow environment,
- to write highly concurrent programs* for solving problems.

REFERENCES

[1] S.J. Allan and A.E. Oldehoeft, "A flow analysis procedure for translation of high level languages to a data flow language", in Proc. 1979 Int. Conf. Parallel Processing, pp.26-34, August 1979.

[2] G.M. Baudet, "Asynchronous iterative methods for multiprocessors", J. ACM, vol.25, No.2, pp.226-244, April 1978.

[3] A.L. Davis, "A data flow evaluation system based on the concept of recursive locality", in AFIPS Conf. Proc. vol.48, pp.1079-1086, June 1979.

[4] J.B. Dennis, "The varieties of data flow computers", in Proc. 1st Int. Conf. Distributed Computing Systems, pp.430-439, October 1979.

[5] J.B. Dennis, G.A. Boughton and C.K.C. Leung, "Building blocks for data flow prototypes", in Proc. 7th Symp. Computer Architecture, pp.1-8, May 1980.

[6] D. Johnson et al., "Automatic partitioning of program in multiprocessor systems", in Proc. COMPCON 80 Spring, pp.175-178, February 1980.

[7] D. Johnson, "Data-flow machines threaten the program counter", Electronic Design, 22, pp.255-258, November 1980.

[8] K.P. Gostelow and R.E. Thomas, "Performance of a simulated data-flow computer", IEEE Trans. Computer, vol.C-29, No.10, pp.905-919, October 1980.

[9] R.M. Karp and R.E. Miller, "Properties of a model for parallel computations: determinancy, termination, queueing", SIAM J. Appl. Math., vol.14, No.4, pp.1390-1411, November 1966.

[10] H.T. Kung, "Synchronized and asynchronous parallel algorithms for multiprocessors", in Algorithms and Complexity: New Directions and Recent Results, (J.F. Traub, ed.), pp. 153-200, Academic Press, New York, 1976.

[11] J.R. McGraw, "Data flow computing: Software development", in Proc. 1st Int. Conf. Distributed Computing Systems, pp.242-251, October 1979.

[12] J.M. Ortega and W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.

[13] C.L. Seitz, "System timing", in Introduction to VLSI Systems (C. Mead and L. Conway), Addison-Wesley Pub. Com., pp.218-262, 1980.

[14] H.S. Stone, "Problems of prallel computation", in Complexity of Sequential and Parallel Numerical Algorithms, (J.F. Traub, ed.) pp.1-16, Academic Press, New York, 1973.

[15] J.F. Traub, "Parallel Algorithms and parallel computational complexity", in Proc. IFIP Congress 74, pp.685-687, North-Holland, Amsterdam, 1974.

[16] P.C. Treleaven, "Exploiting program concurrency in computing systems", Computer, pp.42-50, January 1979.

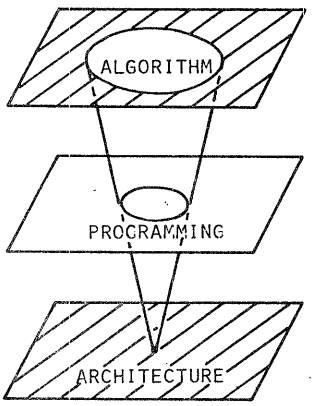

FIG. I.I NEW APPROACH

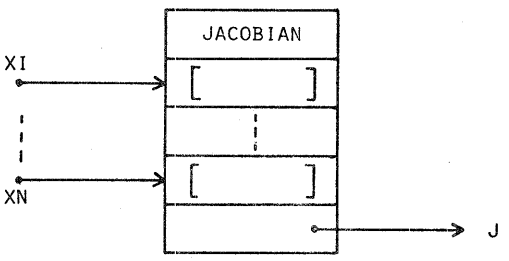

FIG. 4.I ACTIVITY TEMPLATE FOR JACOBIAN

* Efforts in progress are Id, LAU, and VAL languages etc.. Optimizing compiler for data flow language is completely open (McGraw[11]).

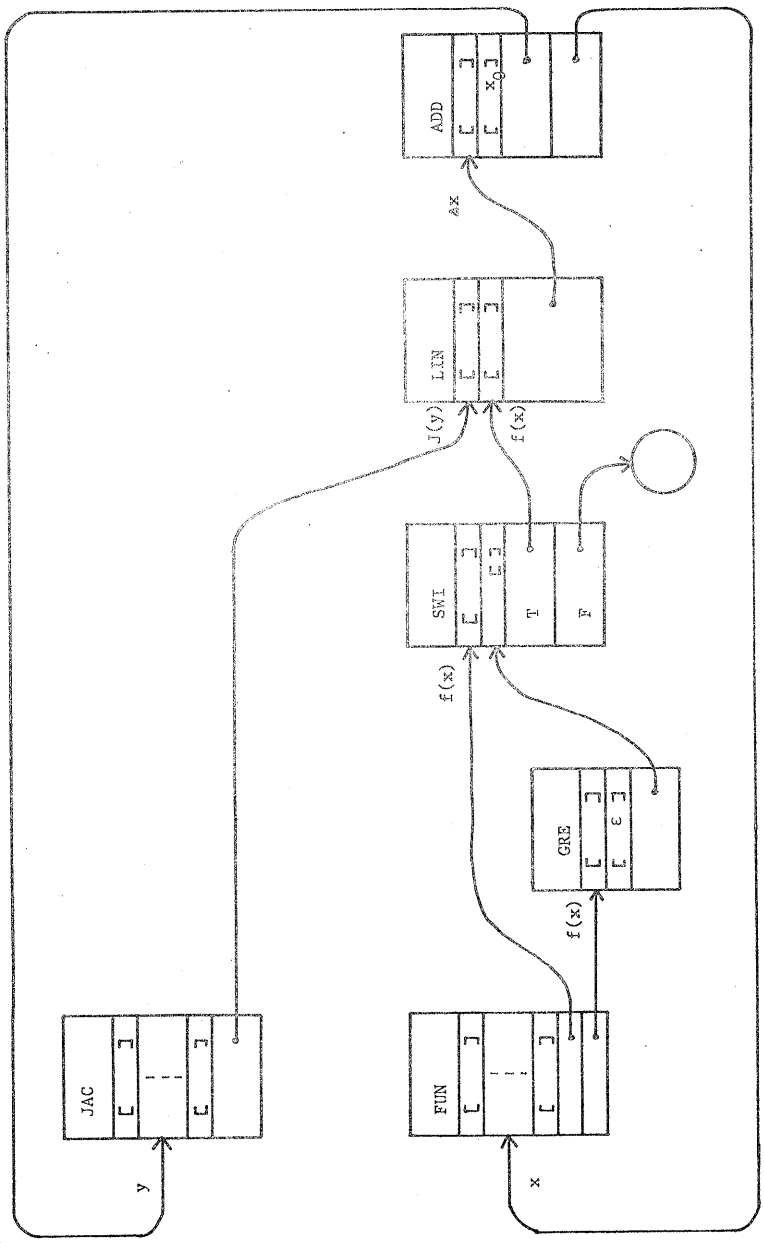

FIG. 4.2 DATA DRIVEN PROGRAM IMPLEMENTATION OF ASYNCHRONOUS PARALLEL NEWTON'S ALGORITHM