MULTIPROCESSING WITH MICROPROCESSORS FOR POWER FLOW ANALYSES

by

Ramanathan Ramanathan

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

APPROVED:

L. L. Grigsby

A. G. Phadke

S. Rahman

C. E. Nunnally

L. C. Frair

May, 1982
Blacksburg, Virginia

ACKNOWLEDGEMENTS

The author wishes to express his heartfelt thanks to his advisor Dr. L. L. Grigsby for his wounderful guidance, timely suggestions and constant encouragement throughout the entire period of this work.

The author is greatly thankful to Dr. L. C. Frair, Dr. C. E. Nunnally, Dr. A. G. Phadke and Dr. S. Rahman for serving on his graduate committee. Dr. C. E. Nunnally deserves more than special thanks for guiding the author through his work with the microprocessor.

The financial assistance provided by the Energy Research Group is greately appreciated.

The author would also like to express his appreciation to his wife Neela and his son Shekar for their patience and understanding throughout his graduate work. Special thanks also go to Neela for her help in preparing the thesis.

CONTENTS

ACKNOWLEDGEMENTS ii

Chapter

page

I. INTRODUCTION 1

Purpose of This Investigation 1
Thesis Outline 3

II. APPLICATIONS OF MULTIPROCESSING TO POWER SYSTEM PROBLEMS 4

Areas Conducive to Multiprocessing in Power Systems 4
Load flow 5
Economic dispatch 6
Machine Organization Types 7
Multiprocessor system 7
Parallel processors 7
Pipeline processors 8
Multifunction processors 8
Parallel Software 8
Vectorizer 9
Parallel methods 9
Some examples in Star-100 vector programming 10
Data Base Considerations 11
Constraints on New Algorithms 12
Constraints on Computer Architecture 13
Possible Architecture for Power System Problems 13

III. DECOMPOSITION TECHNIQUE FOR NETWORK PROBLEMS . . . 16

Definitions 17
Decomposition 19
Effect of System Modifications 25

IV. DISTRIBUTED LOAD FLOW 27

Load Flow 27
Gauss-Seidel method 28
Voltage controlled buses 30
Fort/80 Implementation of Gauss-Seidel Load Flow 31
Simulating two dimensional arrays 35

Distributed Load Flow 35
Effective Decomposition Technique 40

V. IMPROVED AND DISTRIBUTED ECONOMIC DISPATCH 44

Optimum Dispatch 45
Nonlinear programming techniques 46
Classical method without transmission losses 49
Improved algorithm 52
Optimum dispatch with transmission losses . 54
Improved optimum dispatch with transmission losses 57
Derivation of incremental transmission loss 58
Distributed Economic Dispatch 59
Distributed dispatch without transmission losses 59
Distributed dispatch with transmission losses 65

VI. MULTI-MICROPROCESSOR ARCHITECTURE 70

Characteristics of Distributed Microcomputer System 70
Dependency 71
Performance 72
Synchronicity 72
Staticity 73
Operating system and network requirements . 74
Cost 74
Bus Architecture 74
Intel 8080 or 8085 Version of Distributed Processing 76
Intel 8086 Version of Distributed Processing . 78

VII. RESULTS 84

Distributed Processing Simulation Results . . . 84
Results for Microprocessor Load Flow 105
Distributed Microprocessor Load Flow 112
Results for Optimal Decomposition 115
Results for Improved Economic Dispatch . . . 121

VIII. CONCLUSIONS 126

Suggestions for Further Work 128

BIBLIOGRAPHY 130

LIST OF TABLES

Table

page

1. Intel 8087 and 8086 comparison 82

2. Load flow result for the 5 bus system 108

3. Line data for the 6 bus system 109

4. Load flow data for the 6 bus system 110

5. Load flow result for the 6 bus system 111

6. Case I result for the IEEE 14 bus system 118

7. Case II results for the IEEE 14 bus system 119

8. Case III results for the IEEE 14 bus system . . . 120

9. Economic dispatch load flow result for the IEEE 14 bus system 122

10. Incremental cost curve data for generators 123

11. Comparison of nonlinear techniques with the improved method 124

LIST OF FIGURES

Figure

page

1. Distributed Computer System. 15

2. Model network 21

3. Area I is split into two parts 22

4. Adding a new subsystem to the existing system . . . 26

5. Microprocessor load flow flow-chart 34

6. Distributed load flow flow-chart 37

7. Flow-chart for optimum dispatch with transmission losses 56

8. Distributed dispatch without transmission losses . . 64

9. Distributed economic dispatch with losses 69

10. Intel 8085 block diagram for distributed processing 77

11. Intel 8087 coprocessor arrangement 81

12. Intel 8086 version of distributed processing 83

13. IEEE 30 bus system diagram 86

14. IEEE 57 bus system diagram 87

15. CPU time versus no. of subsystems for the 30 bus system 88

16. Execution time versus number of subsystems for the 57 bus system 89

17. Number of iterations versus number of subsystems for the 30 bus system 90

18. Number of iterations versus number of subsystems for the 57 bus system 91

19. CPU time versus no. of iterations in a two subsystem 30 bus system 93

20. CPU time versus no. of iterations in a two subsystem 57 bus system 94

21. No. of data transfers for a 2 subsystem 30 bus system 95

22. No. of data transfers for a two subsystem 57 bus system 96

23. CPU time versus no. of iterations done before passing data for a three subsystem 30 bus system 97

24. CPU time versus no. of iterations done before passing data for a 3 subsystem 57 bus system . . 98

25. No. of data transfers versus no. of iterations done before passing data for a 3 subsystem 30 bus system 99

26. No. of data transfers versus no. of iterations done before passing data for a 3 subsytem 57 bus system 100

27. Speedup for the 30 bus system 101

28. Speedup for the 57 bus system 102

29. Efficiency of the 30 bus system 103

30. Efficiency of the 57 bus system 104

31. 5 bus system diagram 106

32. 6 bus system diagram 107

33. CPU time versus no. of iterations before passing data for a two subsystem 6 bus system 113

34. No. of data passes versus no. of iterations before passing data for a two subsystem 6 bus system . 114

35. IEEE 14 bus test system diagram 116

Chapter I

INTRODUCTION

1.1 PURPOSE OF THIS INVESTIGATION

As the cost and demand for energy is going up rapidly it is necessary to generate, transmit, distribute and utilize energy efficiently. So it is important, both during design and operation stages, to identify factors that influence the cost.

In order to improve the system performance as well as to minimize overall cost, it is necessary to consider the whole system as a single entity in the design stage. This presents computational obstacles and requires improved solution methods and sophisticated programming.

Some of the major factors that contribute to the computational burden in power system design and simulation are:

1. Size of the system network
2. Detailed modeling of the equipment
3. Sophistication of applications
4. Time span of dynamic simulation
5. System expansions

When the above items are considered, for even a moderate size utility, the computational requirements for ideal solutions invariably exceed computer capabilities.

On the other hand, as systems grow larger and more complex, computerized control becomes a necessity. Small errors may lead to complicated problems. Hence fast and reliable on-line control is necessary.

Thus it is necessary to develop new multiprocessing algorithms for achieving speed, dimensionality, reliability, efficiency and increased throughput capability.

Little work has been reported in the published literature [1-16] on the use of multiprocessing to solve power system problems. This author has seen no published report of an actual implementation.

The above reasons promted the author to do the research reported in this thesis. The aim was to develop efficient algorithms to solve the power system problems using distributed processing. This entailed the development of an efficient decomposition technique which was used for load flow and economic load dispatch. Such a technique was developed and substantiated by simulation and actual implementation. Existing algorithms for load flow analysis were modified so that they could be implemented on a distributed processing system.

In order to do economic dispatch, an improved algorithm was developed. This algorithm is faster and also works very efficiently on a distributed processing system.

Studies were conducted on the use of distributed processing for the solution of power system problems using both simulation techniques and actual implementation on microprocessors. The various characteristics of the distributed processing were investigated. The results indicate that distributed processing is a very attractive scheme for power system analysis.

1.2 THESIS OUTLINE

In chapter II, the need for multiprocessing in the power industries are explored. A suitable computer architecture for power system problems is presented. In chapter III, an efficient decomposition technique is developed to solve the problem in parallel. In chapter IV, simulation of distributed processing using IBM 370/158 computer is considered. A microprocessor version of load flow program is developed and distributed load flow for on-line and off-line applications are studied. In chapter V, a novel algorithm for economic dispatch with and without transmission losses is presented. The improved algorithm is extended to distributed processing. Chapter VI deals with the computer architecture for multiple processor system. Intel 8080, 8085 and 8086 versions of distributed and multiprocessing methods are discussed. In chapter VII, results are presented. The final chapter summarizes the results and presents conclusions.

Chapter II

APPLICATIONS OF MULTIPROCESSING TO POWER SYSTEM PROBLEMS

This chapter explores the areas that are conducive to the application of multiprocessors in power system problems. Both software and data base are considered in this work. Different machine organizations are explained. Necessary constraints on new algorithms for power system problems as well as computer architectures are presented. A possible computer architecture for power system problems is also presented.

2.1 AREAS CONDUCIVE TO MULTIPROCESSING IN POWER SYSTEMS

Areas where multiprocessing can be applied in power system can be classified into nine categories as follows:

1. Load flow
2. Economic load dispatch
3. Stability
4. State estimation
5. System control
   a) remote operation
   b) real time control
6. Data acquisition

7. System security
8. Energy management
9. Load management
   a) power factor control
   b) demand control
   c) start up cost

2.1.1 Load flow

A load flow study consists of the calculation of the unknown bus or node voltages, the unknown power generations and the power flow of a given power system for specified loads and some set of specified bus voltages and generations.

A study of load flow is required for the following reasons:

1. To find the most economic operation of generators.
2. To keep voltages and power flow within tolerances.
3. To study upsets or outages.
4. Planning studies for additions or expansions.
5. Continuous performance evaluation.

Some difficulties encountered in achieving a solution to load flow problems are:

1. The equations that are used for load flow model are nonlinear and must be solved iteratively.

2. The system may be large thereby requiring many equations in the model.
3. Equations involve complex coefficients with complex unknowns.
4. Systems are often interconnected.

2.1.2 Economic dispatch

Economic dispatch is the determination of the amount of power that each generator in a system should produce so that the total cost of generation is minimized and the customer demands are met while satisfying all other system constraints.

Load flow and economic dispatch are very important studies in power system analysis. They are naturals for multiprocessing because industries spend large amounts of money and time on them.

A few papers have been published in the area of power system stability using multiprocessors. There has been very little work done in the other areas mentioned.

2.2 MACHINE ORGANIZATION TYPES

There are two types of machine organization [17,18]. They are uniprocessor and multioperation machines. The uniprocessors are classified into overlap and non-overlap type systems. The multioperation machines are classified into multiprocessor systems, parallel processors, pipeline processors and multifunction processors.

2.2.1 Multiprocessor system

This system has P processors each with its own control unit. Each control unit can execute an independent program using its own processor and sharing the memory hierarchy. The processors communicate via memory or direct paths. Some examples for multiprocessing systems are Burroughs D825, B6700, B7700, Univac 1108, 1110, IBM 370/158, Honeywell multics and Bell Labs CLC.

2.2.2 Parallel processors

Parallel processor computers have an array of simultaneously acting identical processors driven by a common control unit. The system can execute only one program at a time. Parallel architecture machines are CDC 7600, CDC Star-100, Illiac IV and Texas Instruments ACS.

2.2.3 Pipeline processors

The basic idea in a pipeline processor is to introduce some simultaneity of processing by breaking a complex, time consuming function into a series of simpler, faster, operations.

2.2.4 Multifunction processors

In this arrangement there is only one control unit and one processor. However, the processor can carry out more than one function simultaneously.

2.3 PARALLEL SOFTWARE

When parallel processors are used, sequential processing software is inadequate. So changes are needed in operating systems, programming languages and compilers.

Parallel processing can be obtained in the following ways:

1. Detect parallel segments in a serial program by using a vectorizer.
2. Write subroutines in machine language for parallel operation.
3. Develop general purpose languages.

2.3.1 Vectorizer

Parallel processing can be easily obtained by using the vectorizer. The major functions performed are:

1. Syntax check
2. Vector analysis and diagnostics
3. Allocation of temporaries
4. Generation of vector sequences
5. Handling effective use of machines such as input/output devices, memory, etc.

The advantages of vectorizers are:

1. The programmer can use standard Fortran
2. Recognition of possible vector operations
3. Transportability

The vectorizer has two phases. One is the analysis phase, and the second is the code generation phase. In the analysis phase, Fortran DO loops are analyzed and individual groups of statements are analyzed to find possible vector operations. The code generation phase depends upon the machine used.

2.3.2 Parallel methods

The following are very important points to be noted when parallel methods are considered:

1. The different algorithms existing for serial processing may not be efficient for parallel processing. So new methods are necessary. For example, a change from manual to computer solution of load flow problems requires a change from loop to nodal equations. Solving the problems in parallel processors may require entirely new parallel processor algorithms.
2. The branching operation has to be considered properly.
3. Instruction loops may not be of the same length.
4. Real work can be done only by persons familiar with power systems and computers.
5. The field is still new.

2.3.3 Some examples in Star-100 vector programming

In this section examples are presented on how to vectorize a given program locally [19].

The conventional DO loop shown below

```
      DO 1 I =1,500
    1 C(I)    =A(I) + B(I)
```

can be written in Star 100 Fortran as follows:

```
      C(1;500) = A(1;500) + B(1;500)
```

In the conventional DO loop if there is a conditional statement as shown below,

```
      DO 1 I =1,500
```

```
      IF (A(I).LT.0.0) GO TO 1
      C(I)   = A(I) + B(I)
    1 CONTINUE
```

The loop can be vectorized by using a bit vector as follows:

```
      BX(1;500) = A(1;500).GE.0.0
      CALL Q8ADDNV (.,A(1;500).,B(1;500).,BX(1;500).,
                    C(1;500)))
```

In this approach the addition is performed only if BX is 1. Otherwise it is not.

By local restructuring, nested DO loops can be vectorized. For example:

```
      DO 7 I = 1,M
      DO 7 J = 2,N
    7 X(I,J) = X(I,J-1) + A(I,J)
```

In vector notation, this becomes

```
      DO 7 J   = 2,N
    7 X(I,J;M) = X(1,J-1;M) + A(1,J;M)
```

2.4 DATA BASE CONSIDERATIONS

The data base plays an important role in designing the computer architecture. The following are important points to be noted during data base considerations:

1. Data should be retrieved from bulk storage on a single fetch instruction.

2. Data should be aligned so that a single instruction stream can be used by the processors.
3. Dummy data should be inserted to overcome the irregularities.
4. Consideration should be given to recomputing needed information rather than fetching it form bulk storage.
5. Proper arrangement and structure should be designed to minimize the transfer of information between the modules.

2.5 CONSTRAINTS ON NEW ALGORITHMS

For a power system problem, the following are necessary constraints to be considered when developing an algorithm:

1. Expansion must be easy in order to include new stations.
2. Processors could be installed at dispatch centers generating stations or distribution stations.
3. Cost must be minimized.
4. Each station must be able to decide its own algorithms.
5. If there is a fault in one of the station computers, it must not affect the entire system.
6. The method should aid on-line control.

2.6 CONSTRAINTS ON COMPUTER ARCHITECTURE

In developing computer architecture for power system problems, the following are important points:

1. Computers must be able to act independently.
2. Only nearby processors must be connected when the computers are miles apart.
3. Data transfer between computers must be minimum.
4. Computers must not spend most of the time waiting for data.
5. Future expansion of the computer system must be easy to include new processors, I/O devices, memory, etc.

2.7 POSSIBLE ARCHITECTURE FOR POWER SYSTEM PROBLEMS

Taking all the above considerations into account, it seems that distributed computing systems are very attractive for power systems. A distributed computing system is one that uses multiple interconnected computers possibly separated by some distance. Figure (1) shows the possible computer architecture. In this figure, regional computers are connected only to the nearby processors and to the central computers by communication links. The regional computers can be any type of computers or microprocessors depending upon the application. In this arrangement, all computers act on various pieces of the problem in a parallel fashion. This

saves time and as they are close to the controlled element, it is possible to use them for online control. The computers may be physically close to each other, thus forming a cluster of computers or they may be miles apart. Distributed processing requires that the system be decomposed into subsystems and requires some data transfer between these subsystems. Hence, an efficient decomposition technique is necessary.

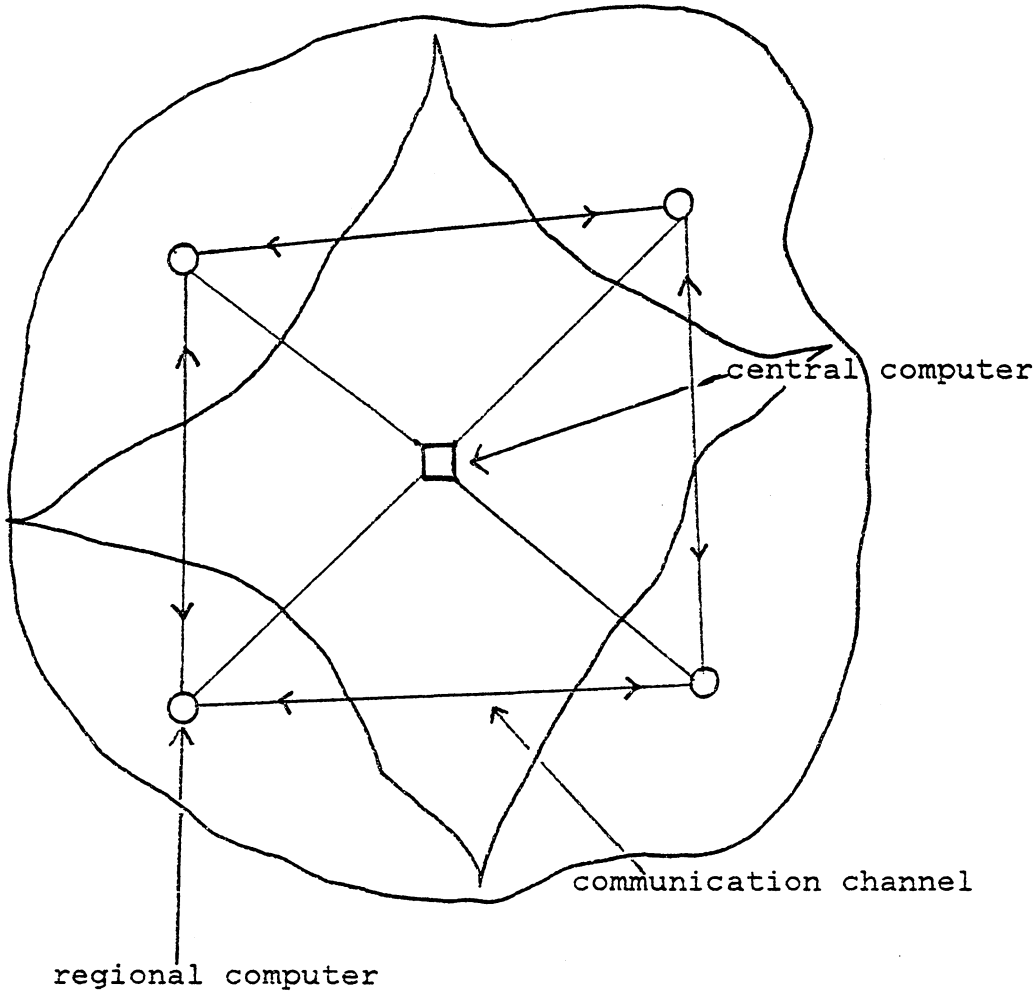

Figure 1: Distributed Computer System.

Chapter III

DECOMPOSITION TECHNIQUE FOR NETWORK PROBLEMS

Since their advent, microprocessor systems have been solving an increasing number of problems from logic replacement to every increasing complex real time mathematical applications. Advances in semiconductor technology continue to improve the performance and expand the capabilities of today's highly integerated and complex microprocessors while decreasing the cost per function. This has opened up the applications which where once done by large main frame computers. The exponential decrease of cost per performance levels in microprocessor and improved performance provide the system designer with new tools to solve today's complex problems. Hence it makes sense to distribute the tasks over multiple processors to perform efficiently and increase throughput capacity. Today's low cost of hardware and high performance makes distributed processing very attractive for power industries.

To perform distributed processing it is necessary to decompose a large complex system, representing many nodes and branches joining one point to another, into a number of subsystems of lower dimension. However, it is clear that each subsystem cannot be handled completely independent of the

other subsystem because in a meaningful problem the subsystem has some interaction through branches. Hence the effectiveness of any decomposition method depends on the following factors:

1. What technique to choose to decompose a large scale system.
2. How to adjust the interdependency among subsystems.
3. How to minimize the interdependency between subsystems.
4. What method can be applied to solve the subsystem.
5. How to recombine the separate subsystem solutions to obtain the overall solution.

3.1 DEFINITIONS

A network G is defined as a collection of points called nodes $N = (n_1, n_2, \ldots . n_n)$ and line segments called branches B $= (b_{ij}; i,j\ N)$. In a power system nodes are represented by the buses and the branches by the transmission lines. The branch b_{ij} is incident at the nodes n_i and n_j. It can be denoted as $b_{ij} = (n_i, n_j)$.

A subnetwork $G_i = (N_i, B_i)$ is a subset of the $B_i \subset B$ branches and $N_i \subset N$ nodes of a network $G = (N,B)$. The subgraph is said to be proper if it consists of strictly less

than all the branches and nodes of the graph.

A network is connected if there exists at least one path between any two nodes.

A cut-set is a set of branches of a connected graph G = (B,N) whose removal causes the graph to become unconnected into exactly two connected subnetworks, with the further stipulation that the removal of any proper subset of this set leaves the graph connected.

For any subnetwork, $G_k = (N_k, B_k)$ has nodes $N_k \subseteq N$ and branches $B_k \subseteq B$. For any two subnetworks, k and l, $N_k \cap N_l = 0$ and $B_k \cap B_l = 0$. The subnetwork is a disjoint network.

A set of cutset branches B_c with $B_c \cap B_k = 0$ for any subnetwork k

and $B_c \cup B_1 \cup B_2 \cup \quad \cup B_m = B$

and $N_1 \cup N_2 \cup N_3 \cup \quad \cup N_m = N$

and $B_c \cap B_1 \cap B_2 \cap \quad \cap B_m = 0$

and $N_1 \cap N_2 \cap N_3 \cap \quad \cap N_m = 0$

is called cut branches.

3.2 DECOMPOSITION

In this section, the decomposition technique is presented. The network equation of a system can be represented as:

$$I = Y\ V$$

or

$$V = Z\ I$$

or in general

$$A\ x = b$$

where

I = node current vector

V = node voltage vector

Y = bus admittance matrix

Z = bus impedance matrix

The method to solve this type of problem using sparsity techniques are presented in literatures [20,21]. The method is very useful in solving large complex problems. But this technique may not be suitable for power system applications where the topology of the system is changing; also the method is not suitable for on-line control. The above decomposed solution may not very well adjust to closing and opening of circuit breakers or expansion in the system. Hence to overcome these difficulties, this work develops a method to decompose a system using the physical structure of the system.

Let us consider a system in figure 2. The system is divided into three areas. In figure 2 the area I subnetwork consist of local currents and the cutset currents from other subnetworks. The cutset currents are currents that are flowing from other subnetworks. In this case the currents flow from area II and area III. The excitation in the subsystem I can be split into two parts as shown in figure 3. One is due to cutset currents and other is due to local currents. The voltage in the system can be obtained by the principle of superposition.

Let the node voltages due to local currents IL be EL and due to cutset currents IC be EC. Then

$$\begin{bmatrix} IL_1 \\ IL_2 \\ IL_3 \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{bmatrix} \begin{bmatrix} EL_1 \\ EL_2 \\ EL_3 \end{bmatrix} \qquad (3.1)$$

$$\begin{bmatrix} 0 \\ IC_{42} \\ IC_{102} \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{bmatrix} \begin{bmatrix} EC_1 \\ EC_2 \\ EC_3 \end{bmatrix} \qquad (3.2)$$

where

Y_{ij} = mutual admittance between nodes i and j

Y_{ii} = self admittance of node i

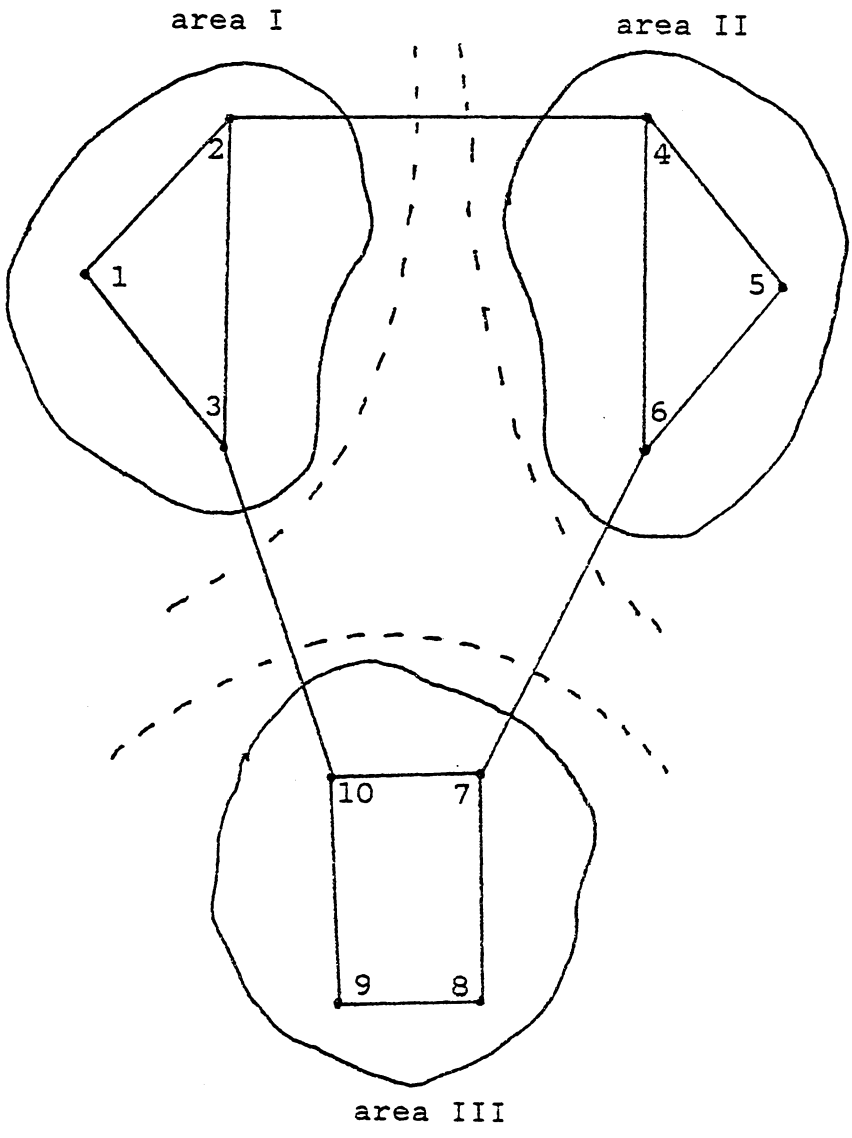

Figure 2: Model network

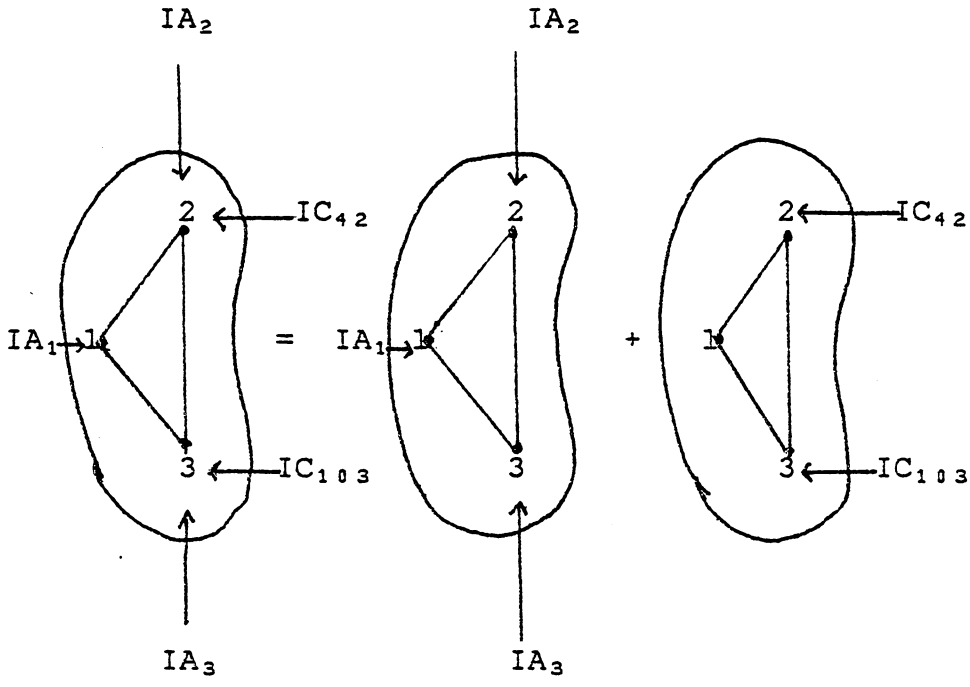

Figure 3: Area I is split into two parts

The bus admittance matrix is only for the local subsystem. The node voltage V at any instant is sum of the voltage EL due to local current plus the voltage EC due to cutset current.

$$V = EC + EL$$

from equations (3.1) and (3.2)

$$\begin{bmatrix} IA_1 \\ IA_2 \\ IA_3 \end{bmatrix} + \begin{bmatrix} 0 \\ IC_{42} \\ IC_{103} \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \qquad (3.3)$$

or in general for any area

$$[IL] + [IC] = [Y]\,[V] \qquad (3.4)$$

the cutset currents in area I can be obtained as follows

$$IC_{42} = (V_4 - V_2)\, Y_{42} \qquad (3.5)$$

$$IC_{103} = (V_{10} - V_3)\, Y_{103}$$

from (3.3) and (3.5)

$$\begin{bmatrix} IA_1 \\ IA_2 \\ IA_3 \end{bmatrix} + \begin{bmatrix} 0 \\ V_4 Y_{42} \\ V_{10} Y_{103} \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22}' & Y_{23} \\ Y_{31} & Y_{32} & Y_{33}' \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \qquad (3.6)$$

where

$$Y_{22}' = Y_{22} + Y_{42}$$

and

$$Y_{33}' = Y_{33} + Y_{103}$$

Now it is important to note that the Y bus matrix includes the cutset elements also. Similarly for area II

$$\begin{bmatrix} IA_4 \\ IA_5 \\ IA_6 \end{bmatrix} + \begin{bmatrix} V_2Y_{24} \\ 0 \\ V_7Y_{76} \end{bmatrix} = \begin{bmatrix} Y_{44}' & Y_{45} & Y_{46} \\ Y_{54} & Y_{55} & Y_{56} \\ Y_{64} & Y_{65} & Y_{66}' \end{bmatrix} \begin{bmatrix} V_4 \\ V_5 \\ V_6 \end{bmatrix} \quad (3.7)$$

and for area III

$$\begin{bmatrix} IA_7 \\ IA_8 \\ IA_9 \\ IA_{10} \end{bmatrix} + \begin{bmatrix} V_6Y_{67} \\ 0 \\ 0 \\ V_3Y_{310} \end{bmatrix} = \begin{bmatrix} Y_{77}' & Y_{78} & Y_{79} & Y_{710} \\ Y_{87} & Y_{88} & Y_{89} & Y_{810} \\ Y_{97} & Y_{98} & Y_{99} & Y_{910} \\ Y_{107} & Y_{108} & Y_{109} & Y_{1010}' \end{bmatrix} \begin{bmatrix} V_7 \\ V_8 \\ V_9 \\ V_{10} \end{bmatrix} \quad (3.8)$$

Another very important factor to note here is that, as the size of the Y bus matrix for an area increases, the ratio of the number of cutsets to the number of nodes for that area will decrease.

The above equations (3.6), (3.7) and (3.8) can be solved in parallel by using any iterative technique.

3.3 EFFECT OF SYSTEM MODIFICATIONS

The decomposition technique presented in the last section is very versatile. The system expansion is very easy to implement in this technique with only slight modifications. For example when a new subsystem IV is added to this model as shown in figure 4, changes are needed only to area III equations. Other subsystem equations are not affected by this.

For example, in the figure 2, when the line connected between nodes 7 to 8 is disconnected the subsystem equations for system III only will change. Hence the decomposition technique presented is very effective as well as suitable for online applications.

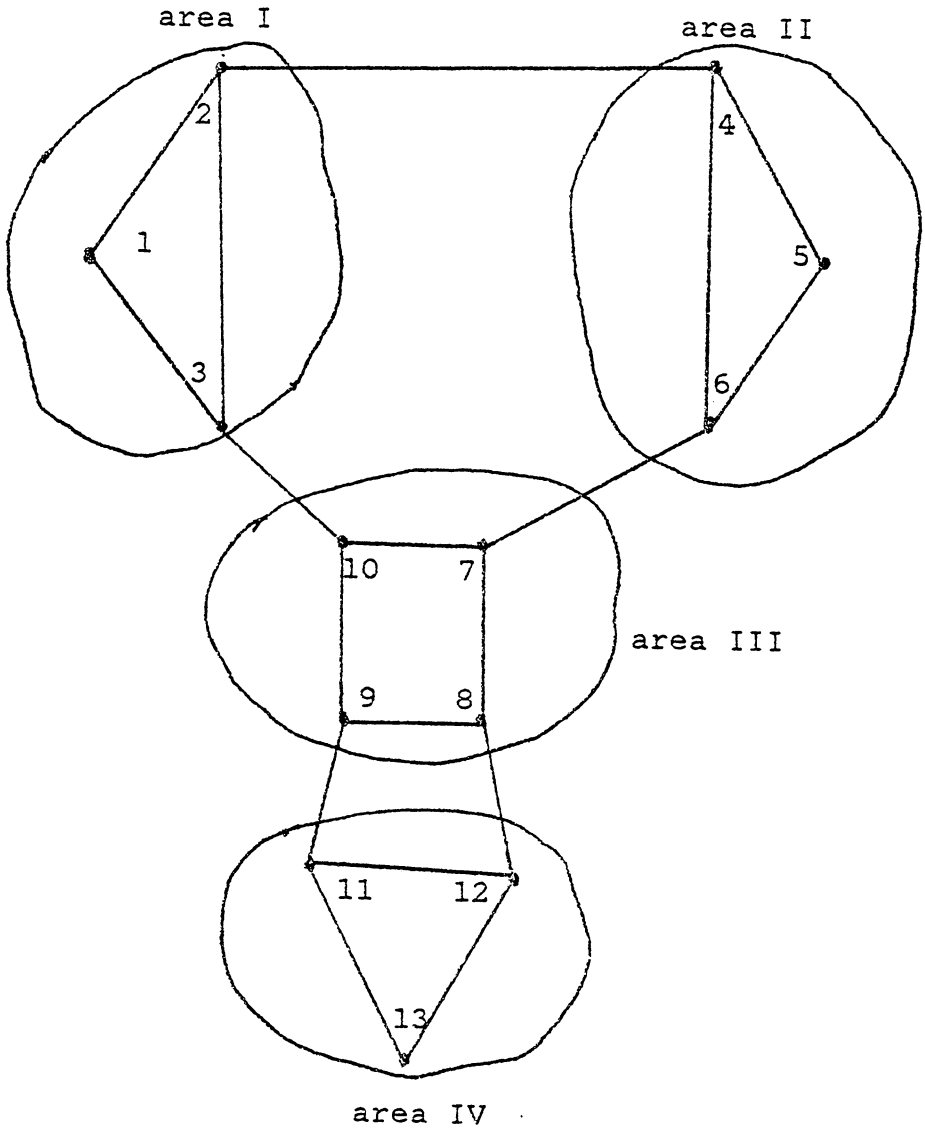

Figure 4: Adding a new subsystem to the existing system

Chapter IV

DISTRIBUTED LOAD FLOW

Recent development of inexpensive and powerful microprocessors have opened up the idea of parallel and distributed processing in the power industries for on-line and off-line control. Hence this chapter presents a technique for distributed processing in the load flow problem. A microprocessor version of load flow algorithm is developed for this purpose. Simulation of distributed processing using IBM 370/158 computer is presented next. Later the microprocessor load flow algorithm is extended to take into account distributed processing.

4.1 LOAD FLOW

The load flow study gives the magnitude and phase angle of the voltage at each bus and the real and reactive power flowing in the lines. The voltage and power flows are needed for future expansions as well as to find the best operating conditions.

So far many methods have been presented in literature [22,23] for load flow analysis. Some of the important ones are

1. Gauss load flow

2. Gauss-Seidel
3. Newton-Raphson
4. Fast decoupled
5. Optimal load flow

The load flow study is very important in the power industry because a large amount of time as well as money is spent on it.

So far very little work has been done to implement this algorithm at the microprocessor level. In this work, mainly Gauss-Seidel method is considered. But the technique can be used for other methods of load flow analysis.

4.1.1 Gauss-Seidel method

The power entering the bus k can be written as

$$P_k + jQ_k = V_k I_k^*$$

or

$$I_k = (P_k - jQ_k)/V_k^* \qquad (4.1)$$

where

P_k = real power entering the bus k

Q_k = reactive power entering the bus k

I_k = current entering the bus k

V_k = voltage at bus k

$*$ = complex conjugate value

The current into a node can be written as

$$I_k = \sum_{j=1}^{n} Y_{kj} V_j \qquad (4.2)$$

where

Y_{kj} = mutual admittance of the node k and j

Y_{kk} = self admittance of the node k

n = number of buses in the system

From equations (4.1) and (4.2)

$$V_k = (1/Y_{kk}) \left[(P_k - jQ_k)/V_k^{*} - \sum_{\substack{j=1 \\ j \neq k}}^{n} Y_{kj} V_j \right] \qquad (4.3)$$

where

k = 2,3,4.....n

The equation (4.3) has complex numbers, is nonlinear and is lengthy. Each bus in the system has an equation like this. The above set of equations has to be solved to find the operating conditions of the system.

Let $(P_k - jQ_k)/Y_{kk} = A_k$

$k = 2,3,.....n$

$Y_{kj}/Y_{kk} = B_{kj}$

$k = 2,3,.....n$

$j = 1,2,.....n$

$j = k$

By applying the Gauss-Seidel technique, the equation(4.3) becomes

$$V_k^{i+1} = A_k/V_k^{*i} - \sum_{j=1}^{k-1} B_{kj}V_j^{i+1} - \sum_{j=k+1}^{n} B_{kj}V_j^{i} \qquad (4.4)$$

The above equation is solved by asssuming some initial values for the bus voltages (i=0). The iteration process is continued until the value of voltages during successive itterations are within some specified tolerance for all k= 2,3,.....n. This can be expressed in equation form as follows:

$$\| V_k^{i+1} - V_k^{i} \| \leq \text{some specified maximum error tolerance}$$

4.1.2 Voltage controlled buses

Generally the system contains some voltage controlled buses. At each of the buses, the real power and the voltage magnitude are specified but the reactive power is an unknown variable. The Gauss-seidel method requires knowledge of the injected reactive power at each bus in order to evaluate the right side of the equation (4.4). For a voltage controlled bus the value of the reactive power must be recomputed for each iteration based on the latest voltage estimates at all the buses as follows:

$$Q_k = -\mathrm{Im}\left[V_k^* \sum_{j=1}^{n} Y_{kj} V_j \right]$$

In reality, the reactive power at the voltage controlled buses will be limited. Therefore the solution must also satisfy

$$Q_k(\min) \leq Q_k \leq Q_k(\max)$$

Whenever the value of the reactive power exceeds the limit, it is set to the limiting value and the bus voltage is set to its previous unadjusted value.

4.2 FORT/80 IMPLEMENTATION OF GAUSS-SEIDEL LOAD FLOW

The method presented in the previous section cannot be easily implemented in assembly language. A higher level language is essential. A Fortran version was developed using FORT/80 [40]. Fort/80 cannot accomodate complex arithmetic, two dimensional arrays, trigonometric functions etc. Algorithms were used to implement trigonometric functions while the programming technique had to be modified to take care of complex numbers and two dimensional arrays.

The equations are separated into real and imaginary parts.

$$A_k = (P_k - jQ_k)/Y_{kk} = E_k + jF_k \qquad (4.5)$$

$$B_{ki} = Y_{ki}/Y_{kk} = S_{ki} + jT_{ki} \qquad (4.6)$$

$$Y_{ki} = g_{ki} + jb_{ki} \qquad (4.7)$$

From equation (4.6) and (4.7)

$$S_{ki} = (G_{ki}*G_{kk} + B_{ki}*B_{kk})/Z$$

$$T_{ki} = (B_{ki}*G_{kk} - G_{ki}*B_{kk})/Z$$

where

$$Z = G_{kk}*G_{kk} + B_{kk}*B_{kk}$$

From equation (4.5) anf (4.7),

$$E_k = (P_k G_{kk} - Q_k B_{kk})/Z$$

$$F_k = -(P_k B_{kk} + Q_k G_{kk})/Z$$

From equation (4.1) and (4.2),

$$P_k - jQ_k = V_k^* \sum_{j=1}^{n} Y_{kj} V_j$$

Let

$$V_k = C_k + jD_k$$

so,

$$P_k = \sum_{i=1}^{n} g_{ki}(C_k C_i + D_k D_i) + b_{ki}(D_k C_i - C_k D_i)$$

and

$$Q_k = -\sum_{i=1}^{n} g_{ki}(C_k D_i - D_k C_i) + b_{ki}(C_k C_i + D_k D_i)$$

From equations (4.3),(4.5) and (4.6),

$$C_k = CC - \sum_{\substack{i=1 \\ i\neq k}}^{n} S_{ki} C_i - T_{ki} D_i$$

$$D_k = DD - \sum_{\substack{i=1 \\ i\neq k}}^{n} S_{ki} D_i + T_{ki} C_i$$

$$CC = (E_k C_k - F_k D_k)/(C_k C_k + D_k D_k)$$

$$DD = (F_k C_k + D_k E_k)/(C_k C_k + D_k D_k)$$

Where

$$|V_k| = \sqrt{C_k^2 + D_k^2}$$

and the phase angle is

$$\angle V_k = \tan^{-1}(D_k/C_k)$$

Now the above equations can be used to get the load flow solutions as shown in the figure(5).

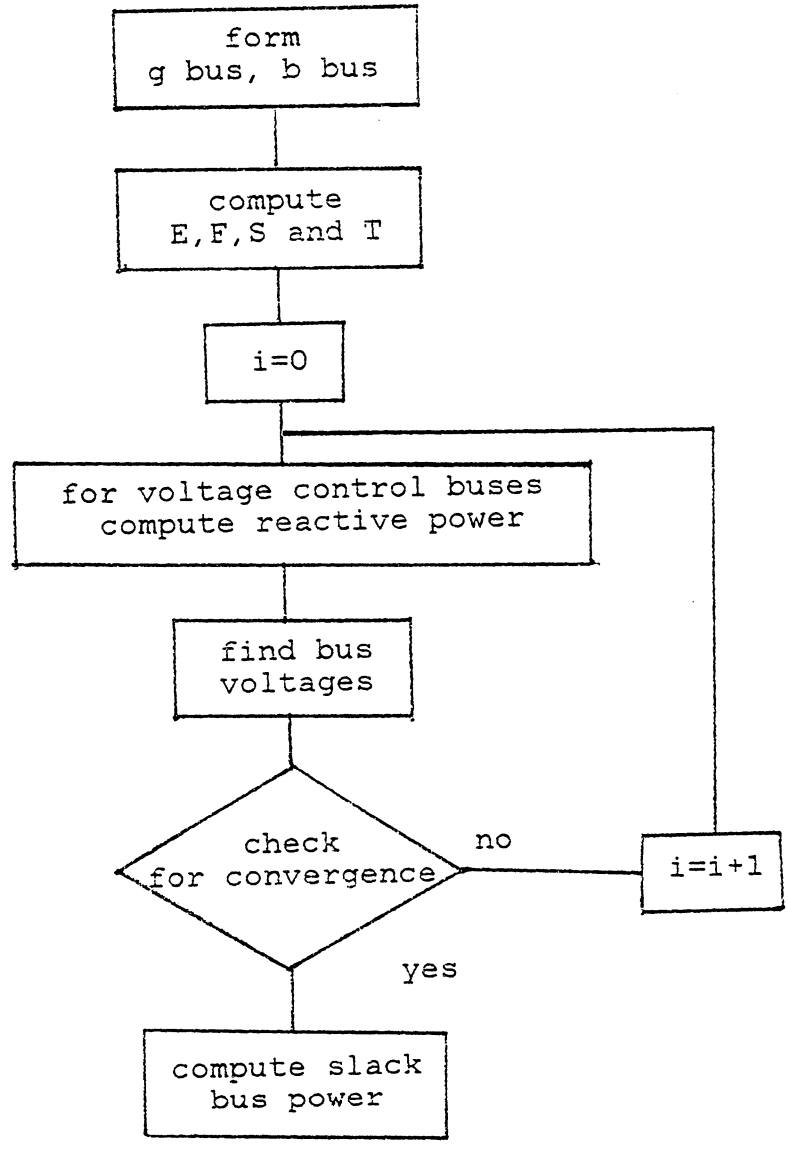

Figure 5: Microprocessor load flow ~~flow~~ chart

4.2.1 Simulating two dimensional arrays

In a two-dimensional array, let N be the row and column dimensions. Let I be the column index and J be the row index. Any element IA(I,J) can be represented as A(K), where

$$K = N(J-1) + I$$

The above equation can be used to simulate a two dimensional array using a single dimensional array.

4.3 DISTRIBUTED LOAD FLOW

Before actual implementation of distributed load flow in the microprocessors, simulation of the above algorithm is carried out in IBM 370/158 computer. The voltages in a subsystem can be obtained from the equation 3.4 once the cutset currents and the local currents are known. The cutset currents can be obtained once the node voltage across the cutset elements and the admittances of the elements are known. In the case of load flow the local currents can be witten as

$$IA_i = (P_i - jQ_i)/V_i^* \qquad (4.8)$$

Therefore for the subsystem I, presented in chapter 3, the load load flow eqations are:

$$\begin{bmatrix} (P_1-jQ_1)/Vc_1 \\ (P_2-jQ_2)/Vc_2 \\ (P_3-jQ_3)/Vc_3 \end{bmatrix} + \begin{bmatrix} 0 \\ V_4Y_{42} \\ V_{10}Y_{103} \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22}' & Y_{23} \\ Y_{31} & Y_{32} & Y_{33}' \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \qquad (4.9)$$

where

Vc = complex conjugate value of V

The equation (4.9) is a nonlinear complex equation. It can be solved only by iterative techniques. The voltages V_4 and V_{10} that are needed to solve the above equation are obtained from the other subsystem via some type of communication links.

In this technique, each subsystem computer first forms the subsystem bus admittance matrix. Then the bus admittance matrix is modified to take into account cutset elements or interconnections between them.

In the load flow studies some set of bus voltages, loads and generations are known. The aim is to find the remaining unknowns. In this study the unknown bus voltages are assumed to be one per unit during the start of the iterative process. Once a few iterations are done on the local subsystem, the data needed between the subsystems are passed. When the updated value is obtained from the other subsystems, the iteration is continued with the most recent values. This process is continued till convergence is obtained in all the subsystems. The technique is presented in figure 6.

It is important to find out 'speedup' and 'efficiency' for an algorithm when using multiprocessing. The speedup of a m-processor computer system over a uniprocessor system is denoted as

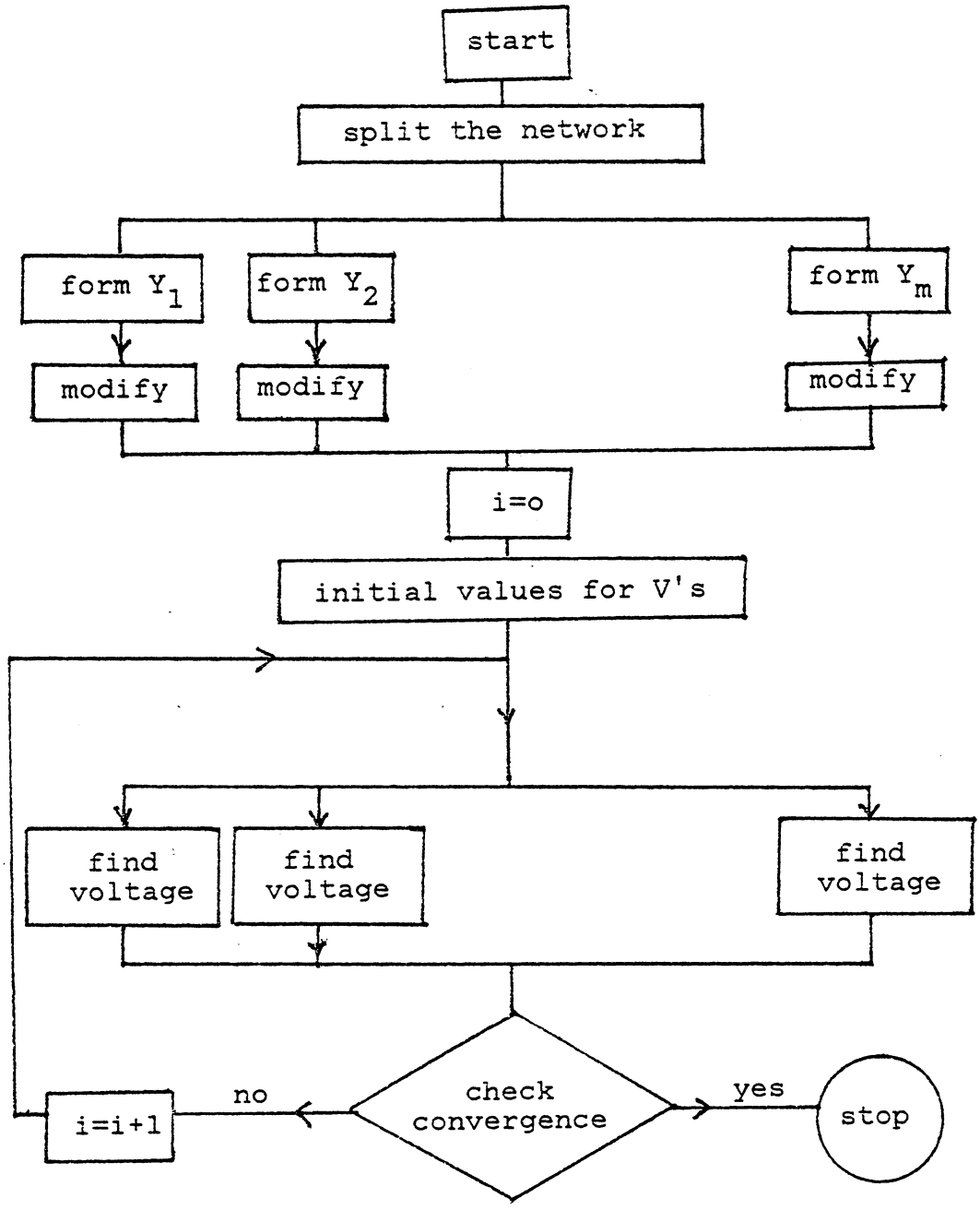

Figure 6: Distributed load flow flow chart

$$Sm[A] = T1[A]/Tm[A] \geq 1$$

where

T1[A] = time taken by a uniprocessor system to complete a task.

Tm[A] = time taken by a m-processor computer system to complete the same task.

The efficiency of this computation is defined as

$$Em[A] = Sm[A]/m \leq 1$$

In the case of serial processing for iterative algorithm when there are m subsystems and the algorithm takes n iteration to achieve the solution then the total time taken is

$$Ts = Tso + \sum_{j=1}^{m} \sum_{i=1}^{n} T_{ij}$$

where

Ts = total time to obtain final solution by serial processing

Tso = overhead time for serial processing

T_{ij} = time taken by ith iteration in the jth subsystem

But when all the subsystems are calculated in parallel by using m processors, the total time is

$$Tm = Tmo + \sum_{j=1}^{m} \max(T_{1j}, T_{2j}, \ldots\ldots\ldots\ldots T_{mj})$$

$$= Tmo + \sum_{j=1}^{m} Tmax_j$$

where

$$Tmax_j = \max(T_{1j}, T_{2j}, \ldots\ldots\ldots\ldots T_{mj})$$

Tmo = overhead time by using m processor

The aim is to maximize the efficiency of the m processor system when the processors are identical. To avoid the wait states, all processors must finish the calculations at the same time during every iteration. Then

$$T_{i1} = T_{i2} = \ldots\ldots\ldots\ldots = T_{im}$$

This indicates that each subsystem must be identical and must have the same number of nodes.

Further reduction in time can be obtained by reducing the overhead time and the number of iterations. The overhead time is the function of number of data passes between the subsystems, type of communication links between the subsystems etc. In a typical system there will be some interaction between the subsystems. So there will be some data passes.

As the number of iterations increases, the total time will also increase. It will be shown later that if the data

is not passed after every iteration, the number of iterations will increase. So there is an optimum number of iterations that can be done for a system before passing the data.

It is also important to minimize the interaction between the subsystems so that convergence will be fast. The next section describes the technique to do that.

4.4 EFFECTIVE DECOMPOSITION TECHNIQUE

This section explains the effective decomposition technique to minimize the interaction between the subnetworks for power flow analysis. The subnetworks are interconnected to other subnetworks through cutset branches. The decomposition of the network must allow for the interactions between subnetworks. Hence to obtain optimal decomposition, the interactions through cutset elements must be minimized.

$$I_{ij} = (V_i - V_j)/(R + jX) \qquad (4.10)$$

where

V_i = bus voltage at ith bus

R = resistance of the line

X = reactance of the line

Z = $R + jX$

I_{ij} = current flowing from bus i to bus j

Then the power flow from bus i to bus j is

$$P_{ij} + jQ_{ij} = V_i \, I_{ij}^{*} \qquad (4.11)$$

From equations 4.10 and 4.11

$$P_{ij} = (R|V_i|^2 - R|V_i||V_j| \cos\theta + X|V_i||V_j| \sin\theta)/(R^2 + X^2)$$

$$Q_{ij} = (X|V_i|^2 - X|V_i||V_j| \cos\theta - R|V_i||V_j| \sin\theta)/(R^2 + X^2)$$

where

$\theta = \angle V_i - \angle V_j$

P_{ij} = real power flow from bus i to bus j

Q_{ij} = reactive power flow from bus i to bus j

In general, for a small change in the magnitude of the bus voltage, the real power at the bus does not change appreciably.

Then

$$(\Delta P_{ij}) = (\partial P_{ij}/\partial\theta)\partial\theta + (\partial P_{ij}/\partial V_i)\partial V_i + (\partial P_{ij}/\partial V_j)\partial V_j$$

where the sensitivity factors are

$$(\partial P_{ij}/\partial\theta) = (R|V_i||V_j| \sin\theta + X|V_i||V_j| \cos\theta)/|Z|^2$$

$$(\partial P_{ij}/\partial V_i) = (2R|V_i| - R|V_j| \cos\theta + X|V_j| \sin\theta)/|Z|^2$$

$$(\partial P_{ij}/\partial V_j) = (-R|V_i| \cos\theta + X|V_i| \sin\theta)/|Z|^2$$

when $V_i \simeq V_j \simeq 1$, θ is small and $R << X$

$$(\partial P_{ij}/\partial\theta) \simeq 1/|Z|$$

$$(\partial P_{ij}/\partial V_i) \simeq R/|Z|^2$$

$$(\partial P_{ij}/\partial V_j) \simeq -R/|Z|^2$$

The change in the reactive power

$$(\Delta Q_{ij}) = (\partial Q_{ij}/\partial\theta)\partial\theta + (\partial Q_{ij}/\partial V_i)\partial V_i + (\partial Q_{ij}/\partial V_j)\partial V_j$$

where

$$(\partial Q_{ij}/\partial\theta) = (X|V_i||V_j| \sin\theta - R|V_i||V_j| \cos\theta)/|Z|^2$$

$$(\partial Q_{ij}/\partial V_i) = (2X|V_j| - X|V_j| \cos\theta - R|V_j| \sin\theta)/|Z|^2$$

$$(\partial Q_{ij}/\partial V_j) = (-X|V_i| \cos\theta - R|V_i| \sin\theta)/|Z|^2$$

Also, for a small change in the phase angle of the bus voltage the reactive power does not change appreciably.

Then

$$(\partial Q_{ij}/\partial\theta) \simeq -R/|Z|^2$$

$$(\partial Q_{ij}/\partial V_i) \simeq 1/|Z|$$

$$(\partial Q_{ij}/\partial V_j) \simeq -1/|Z|$$

From these equations it is clear that to have minimum interaction between the subnetworks it is better to cut the branches that have the highest impedances.

Chapter V

IMPROVED AND DISTRIBUTED ECONOMIC DISPATCH

The reserves for coal, oil, natural gas and nuclear energy are depleting while the demand and the cost of energy as well as size and complexity of the system are rapidly increasing. As a result, there is a great interest in the economic operation of the system. To achieve the above goals, economic dispatch is essential. As it is well known, economic dispatch is the determination of the amount of power that each generator in a system should produce so that the total cost of generation is minimized and the customer demands are met while satisfying system constraints. The important factors for economic operation of the system are operating efficiencies of generating plants, fuel and operating cost and transmission line losses. It is also obvious that the most efficient generator in the system does not necessarily yield the lowest cost because it may be far off from the load or the fuel cost may be high in that region.

Since the tariffs are regulated, the reduction in operating expenses is one of the important ways of increasing profits. A reduction of ten dollars per hour indicates an annual savings of about 90,000 dollars.

To solve the above economic dispatch problem, several types of computer algorithms are available[24-31]. As economic dispatch is generally done every few minutes, it is essential to reduce the total computer run time. Depending upon the system size and complexity, it may take appreciable time to converge.

To help to overcome these difficulties, this work presents algorithms with and without transmission losses. A method is developed to find the value of the Lagrange multiplier or penalty factor in closed form. As no iterative process is employed to find the Lagrange multiplier, the algorithm is fast and appears to have no convergence problems. The above algorithm with transmission losses is implemented and substantiated by the results on a standard test system. The improved algorithm is extended to distributed processing.

5.1 OPTIMUM DISPATCH

In a system, for a given load demand, there is an infinite number of possible combinations to generate the required power. But all the combinations may not be optimum. The main object of the economic dispatch is to find the optimum combination such as to minimize the scalar cost function while simultaneously satisfying the necessary system constraints.

5.1.1 Nonlinear programming techniques

In general, the optimal power flow is obtained by optimizing an objective function subject to equality and inequality constraints on some or all of the control and functional parameters. Mathematically the above problem can be stated as follows:

optimize $e(x,u)$

subject to

$$f(x,u) = 0$$

and

$$h(x,u) \leq 0$$

where x and u are the state and control vectors respectively.

The above equations are nonlinear and they can be solved by various penalty function techniques. These techniques transform the constrained optimization problem to an unconstrained problem.

The augmented objective function is

$$e^*(x,u,\lambda,\mu) = e(x,u) + \lambda^T f(x,u) + \mu^T h(x,u)$$

The optimum solution for the above equation can be obtained by applying the Kuhn-Tucker theorem as follows:

$$(\partial e^*/\partial x) = (\partial e/\partial x) + \lambda^T(\partial f/\partial x) + \mu^T(\partial h/\partial x) = 0$$

$$(\partial e^*/\partial u) = (\partial e/\partial u) + \lambda^T(\partial f/\partial u) + \mu^T(\partial h/\partial u) = 0$$

$$(\partial e^*/\partial \lambda) = f(x,u) = 0$$

$$\mu^T h(x,u) = 0 \ , \ \mu > 0$$

The above equations are nonlinear and the solution can be obtained by search techniques.

Some proposed search techniques are: Steepest-Descent (optimal gradient), Fletcher-Powell and the Hessian-Matrix approach. For the above methods initial values for the control vectors are assumed. The feasible solution is computed after successive adjustment of the control parameters as follows:

$$u^{i+1} = u^i + \Delta u^i$$

The main difference between the different techniques is only in the method of computing the new value of u.

In the Steepest-Descent method, the change of u during the ith and i+1 iteration is obtained as follows:

$$\Delta u^i = \theta^i \nabla e(x,u^i) \qquad (5.1)$$

and theta at ith iteration is obtained by minimizing $e(x,u^{i+1})$ with respect to theta. In the Fletcher Powell algorithm,

$$\Delta u^i = -\theta^i \, [\nabla^2 e(x,u^i)]^{-1} \, [\nabla e(x,u^i)] \quad (5.2)$$

and theta at the ith iteration is obtained as in the Steepest-Descent method. However, in Hessian-Matrix the change in u between i and i+1 is obtained by solving the following equation

$$[\nabla^2 e(x,u^i)]\Delta u^i = -\nabla e(x,u^i) \quad (5.3)$$

In on-line applications, the system state can be obtained directly from field measurements. However, for off-line applications, the system state is obtained from load-flow results. Therefore it is necessary to repeat the load flow computation at every iteration step in order to solve the Kuhn-Tucker equations.

From equations (5.1 to 5.3) it is clear that some type of iteration process is needed to obtain the change in u between the ith and ith+1st iteration. Depending upon the technique used, the iteration process can take appreciable computer time and may require some initial conditions to start the iteration.

5.1.2 Classical method without transmission losses

This section presents a technique[23] to do economic dispatch without transmission losses. A system serving an urban area of high load density will have relatively small transmission losses. The overall production cost, CT, to generate the required load demand can be expressed as follows:

$$CT = \sum_{i=1}^{m} C_i$$

where

C_i = the cost expressed in dollars/hour per ith unit

m = number of generators in the system.

The cost functions in this analysis are assumed to be monotonicaly increasing funtions. Hence optimum dispatch with inequality constraints can be solved using the relaxation principle. In this technique first the problem is solved without the inequality constraints and optimal solution is obtained. If the optimum solution is within the permissible limits the optimum feasible solution is obtained. When the constraints are violated they are set to the limiting values and the problem is solved once again. Usually generators capacities are limited. Therefore it is necessary to carry out this procedure when the limits are exceeded and the solution is suboptimal.

In a system the total generation must be equal to the total load when there are no transmission losses. Therefore, the augmented production cost of electrical energy is

$$CC = \sum_{i=1}^{m} C_i \quad -\lambda[\sum_{i=1}^{m} PG_i \quad - PD \ ] \qquad (5.4)$$

where

PG_i = power generated at ith unit

PD = total demand

λ = Lagrange multiplier

From equation (5.4), the minimum total cost is obtained when the partial derivatives of CC with respect to PG_i are zero. That is:

$$(\partial C_1/\partial PG_1) = (\partial C_2/\partial PG_2) = ... \ = \lambda$$

Or in other words, at optimum dispatch all the generators should operate at equal incremental production costs. That is

$$(\partial C_i/\partial PG_i) = ICC_i = \lambda$$

where

ICC_i = incremental production cost of ith unit.

Assume that the power generation at any unit can be expressed as a function of incremental cost as follows:

$$PG_i = a_i + b_i\ (ICC_i) + c_i\ (ICC_i)^2 + \ldots\ldots \quad (5.5)$$

The coefficients a_i, b_i, c_i, etc, are generally obtained by curve-fitting. The number of terms required for the power generated at unit i depends upon the required accuracy. It is also important to note that there may be discontinuities or jumps in the incremental cost curves. Hence it is necessary to take the proper section of the incremental cost curves to find out the power generated at different units and may have to switch back and forth to the different section of the incremental cost curves.

To find the optimum dispatch, the following steps are followed.

Step 1: Make an initial guess for the Lagrange multiplier

Step 2: Calculate power generated in all units

Step 3: Check if total generation minus total load is within specified value. If yes, quit, otherwise continue.

Step 4: Make a new guess for the Lagrange multiplier.

Step 5: Go to step 2.

The above algorithm is easy to program, but it has some disadvantages. It requires some initial guess and may re-

quire a lengthy time to converge depending upon the search technique used to update or to estimate the Lagrange multiplier.

5.1.3 Improved algorithm

In the previous algorithm, it is clear that most of the time is spent in finding a value of the Lagrange multiplier. To overcome these diffulties, the following method is presented. For simplicity in notation let

$$ICC_i = k \tag{5.6}$$

From equations (5.5) and (5.6)

$$PG_i = a_i + b_i\, k + c_i\, k^2 + \ldots\ldots$$

$$i = 1, 2, \ldots\ldots\ldots m$$

$$\sum_{i=1}^{m} PG_i = \sum_{i=1}^{m} a_i + k \sum_{i=1}^{m} b_i + k^2 \sum_{i=1}^{m} c_i \ldots\ldots \tag{5.7}$$

$$\sum_{i=1}^{m} PG_i = PD \tag{5.8}$$

From equation (5.7) and (5.8)

$$\sum_{i=1}^{m} a_i + k \sum_{i=1}^{m} b_i + k^2 \sum_{i=1}^{m} c_i + ... = PD \qquad (5.9)$$

Let $$A = \sum_{i=1}^{m} a_i - PD$$

$$B = \sum_{i=1}^{m} b_i$$

$$C = \sum_{i=1}^{m} c_i$$

Therfore equation (5.9) becomes

$$A + B\,k + C\,k^2 + ...= 0 \qquad (5.10)$$

When the order of the above equation is less than or equal to four, roots can be obtained using available formulae. When the order is greater than four, roots can be obtained by some other well known root-finding techniques. In general, for most of the applications, equation (5.5) is assumed to be of second order. Therefore the order of equation (5.10) also will be two. So the solution for the above equation becomes

$$k = (-B \pm \sqrt{B*B - 4AC})/2C$$

Now for a given system, the a's, b's and c's are constant. So when the load changes, only the parameter A in equation (5.10) will change. Therefore, k, the incremental cost for optimum operating point, can be obtained. Once k is known, generations are easily obtained by using equation (5.5).

In the above method there is no iteration process for obtaining the Lagrange multiplier. The above algorithm is fast and easy to implement for online application and there is no need for an initial guess for the Lagrange multiplier.

5.1.4 Optimum dispatch with transmission losses

When the energy is transmitted over large distance, the transmission losses may become an important factor. Taking the transmission losses into account, the augmented production cost is

$$CC = \sum_{i=1}^{m} C_i - \lambda \left[\sum_{i=1}^{m} PG_i - PD - PL \right]$$

where

PL = total transmission loss.

The scalar cost function is minimum when

$$(\partial CC/\partial PG_i) = ICC_i - \lambda + \lambda\,(\partial PL/\partial PG_i) = 0$$

for i = 1,2, ...m, and

$$\sum_{i=1}^{m} PG_i - PD - PL = 0 \qquad (5.11)$$

or

$$ICC_i = \lambda\,[1 - (\partial PL/\partial PG_i)]$$

where

$(\partial PL/\partial PG_i)$ = incremental transmission loss due to ith unit.

$= ITL_i$

From the above equations, the loading of each generator can be obtained using figure 7.

In this method, the convergence results are a function of the system and the generator cost characteristics. For on-line implementation, the Lagrange multiplier computation loop [29] may be a big burden. To eliminate the above burden, the next section gives an algorithm to find the Lagrange multiplier in closed form.

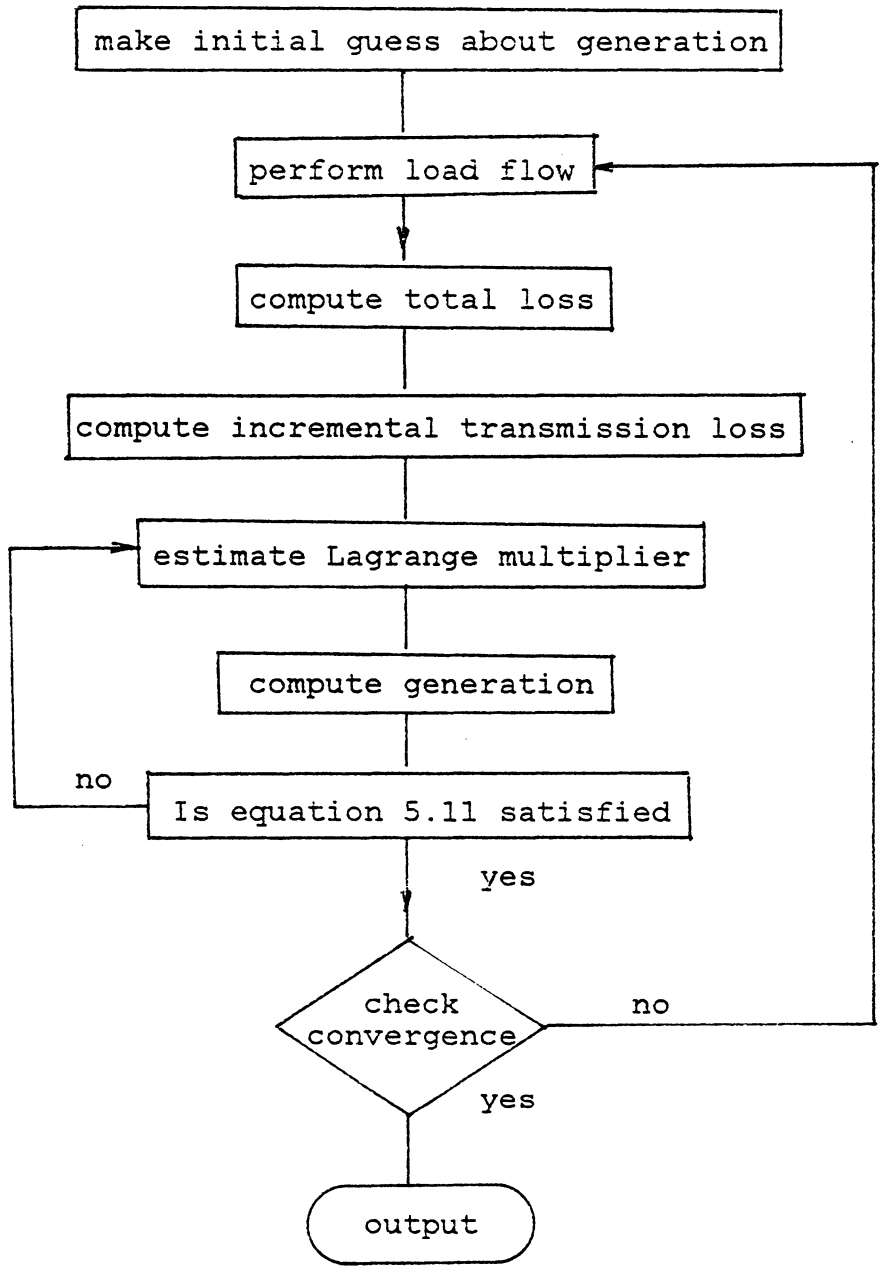

Figure 7: Flow-chart for optimum dispatch with transmission losses

5.1.5 Improved optimum dispatch with transmission losses

The cost function is minimum when

$$ICC_i = \lambda \ (\ 1 \ - ITL_i \ ) \qquad (5.12)$$

for i = 1, 2, ... m.

From equations (5.5) and (5.12)

$$PG_i = a_i + b_i \ \lambda(1-ITL_i) + c_i \ \lambda^2(1-ITL_i)^2$$

$$\sum_{i=1}^{m} PG_i = \sum_{i=1}^{m} a_i + \lambda \sum_{i=1}^{m} b_i(1-ITL_i) + \lambda^2 \sum_{i=1}^{m} c_i(1- ITL_i)^2 \qquad (5.13)$$

From equation (5.11) and (5.13)

$$D + E\lambda + F\lambda^2 = 0 \qquad (5.14)$$

where

$$D = \sum_{i=1}^{m} a_i - PD - PL$$

$$E = \sum_{i=1}^{m} b_i(1-ITL_i)$$

and

$$F = \sum_{i=1}^{m} c_i(1-ITL_i)^2$$

From expression (5.14)

$$\lambda = (-E \pm \sqrt{E*E - 4\ F\ D})/2F$$

Once, the Lagrange multiplier is known, generations can be calculated. In this method, there is no iteration process for the Lagrange multiplier, and hence the convergence speed is again increased.

5.1.6 Derivation of incremental transmission loss

The incremental transmission loss can be obtained by taking the partial derivative of PL with respect to generation at ith unit and from reference [23];

$$ITL_i = \partial PL/\partial PG_i = \sum_{\substack{j=1\\k=1}}^{n} (\partial\ /\partial P_i)[a_{jk}(P_jP_k + Q_jQ_k) + b_{jk}(Q_jP_k - P_jQ_k)] \qquad (5.15)$$

where

$$a_{jk} = (r_{jk}/\ V_j\ V_k\ )\ \cos(\theta_j-\theta_k)$$

$$b_{jk} = (r_{jk}/\ V_j\ V_k\ )\ \sin(\theta_j-\theta_k)$$

P_j = real bus power at jth bus

Q_j = reactive bus power at jth bus

V_j = bus voltage at jth bus

r_{jk} = bus resistance between bus j and k

θ_j = the phase angle of V_j with respect to the slack bus voltage

n = total number of buses in the system

From the equation (5.15) approximate formula for incremental transmission loss is

$$ITL_i = 2 \sum_{k=1}^{n} (P_k a_{ik} - Q_k b_{ik})$$

5.2 DISTRIBUTED ECONOMIC DISPATCH

The last section presents a fast economic dispatch algorithm which is accurate and appears to have good convergence properties. The speed of the above algorithm can be further improved by parallel processing. To obtain on-line control as well as speed, distributed processing is essential. This section presents an algorithm for distributed processing which is also suitable for parallel processing.

5.2.1 Distributed dispatch without transmission losses

The basic idea in this section is similar to the improved economic dispatch algorithm [24] but the idea is expanded to multiprocessing. The overall production cost CT, to generate

the required load demand in the entire system can be expressed as:

$$CT = \sum_{j=1}^{p} CT_j$$

where

CT_j = total production cost in subsystem j in dollars/hour

p = number of subsystems in the system

and

$$CT_j = \sum_{i=1}^{m_j} C_{ij}$$

where

C_{ij} = the cost expressed in dollars/hour per ith unit in jth subsystem

m_j = number of generators in the jth subsystem

When there is no transmission loss, the total generation must match the total load in the system.

$$PD = \sum_{j=1}^{p} \sum_{i=1}^{m_j} PG_{ij} = \sum_{j=1}^{p} PD_j$$

where

PG_{ij} = power generated at ith unit in jth subsystem

PD = total demand

PD_j = total demand in jth system

The augmented production cost of energy is

$$CC = \sum_{j=1}^{p} \sum_{I-1}^{m_j} C_{ij} - \lambda\left(\sum_{j=1}^{p} \sum_{i=1}^{m_j} PG_{ij} - \sum_{j=1}^{p} PD_j \right) \qquad (5.16)$$

where

λ = Lagrange multiplier

From equation (5.16) minimum cost is obtained when the partial derivatives of CC with respect to generations are zero. That is

$$\partial C_{11}/\partial PG_{11} = \partial C_{12}/\partial PG_{12} = \ldots = \partial C_{ij}/\partial PG_{ij} = \ldots = \lambda$$

The optimum cost is obtained when all the generators operate at equal incremental production costs. That is

$$\partial C_{ij}/\partial PG_{ij} = ICC_{ij} = \lambda \qquad (5.17)$$

where

ICC_{ij} = incremental production cost of ith unit in jth subsystem

The power generated at ith unit in jth subsystem can be expressed as a function of incremental cost as follows.

$$PG_{ij} = a_{ij} + b_{ij} (ICC_{ij}) + c_{ij} (ICC_{ij})^2 + \ldots \quad (5.18)$$

The coefficients a's, b's and c's etc. are generally obtained by curve fitting the cost curve for ith generator in jth subsystem. From equation (5.17) and (5.18)

$$PG_{ij} = a_{ij} + b_{ij} \lambda + c_{ij} \lambda^2 + \ldots$$

and

$$\sum_{j=1}^{p} \sum_{i=1}^{m_j} PG_{ij} = \sum_{j=1}^{p} \sum_{i=1}^{m_j} a_{ij} + \lambda \sum_{j=1}^{p} \sum_{i=1}^{m_j} b_{ij} + \lambda^2 \sum_{j=1}^{p} \sum_{i=1}^{m_j} c_{ij} \quad (5.19)$$

$$= \sum_{j=1}^{p} PD_j$$

Let

$$A_j = \sum_{i=1}^{m_j} a_{ij} - PD_j$$

$$B_j = \sum_{i=1}^{m_j} b_{ij}$$

$$C_j = \sum_{i=1}^{m_j} c_{ij}$$

Therefore equation (5.19) becomes:

$$\sum_{j=1}^{p} A_j + \lambda \sum_{j=1}^{p} B_j + \lambda^2 \sum_{j=1}^{p} C_j = 0 \qquad (5.20)$$

Let

$$\sum_{j=1}^{p} A_j = A$$

$$\sum_{j=1}^{p} B_j = B$$

$$\sum_{j=1}^{p} C_j = C$$

Hence the equation (5.20) becomes:

$$A + \lambda B + \lambda^2 C + \ldots = 0$$

When the order of the above equation is less than or equal to four, a closed form solution can be obtained.

Figure 8 shows a flow chart for the above distributed economic dispatch algorithm. In this method, there is no iteration process for the Lagrange multiplier. Additionally, the subsystems are computed in parallel. Thus, the method will be fast compared to other algorithms.

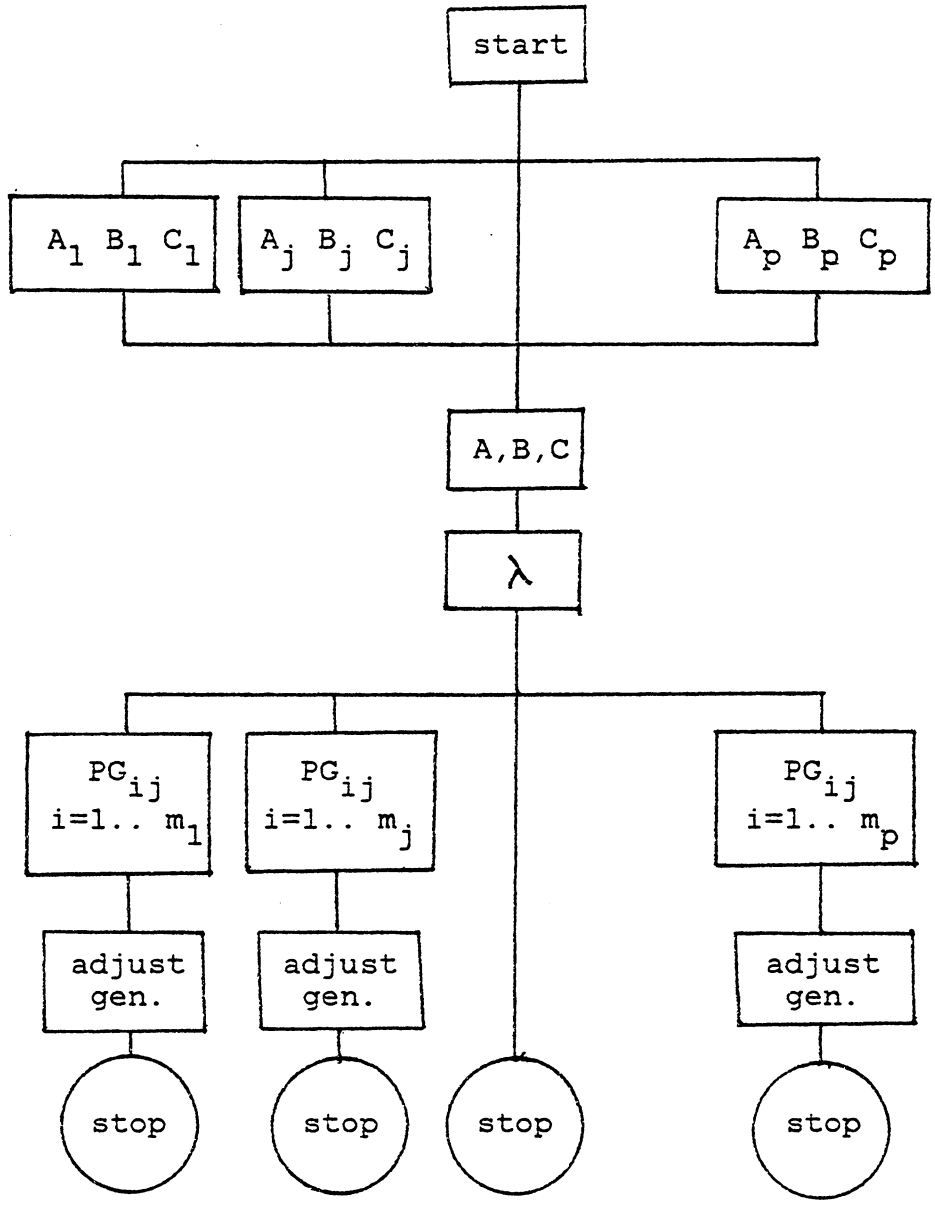

Figure 8: Distributed dispatch without transmission losses

5.2.2 Distributed dispatch with transmission losses

When the transmission losses are taken into account the power balance equation becomes

$$\sum_{j=1}^{p} \sum_{i=j}^{m_j} PG_{ij} = \sum_{j=1}^{p} PD_j + \sum_{j=1}^{p} PL_j \qquad (5.21)$$

where

PL_j = total transmission loss in jth subsystem

and

$$\sum_{j=1}^{p} PL_j = PL$$

The augmented production cost is

$$CC = \sum_{j=1}^{p} \sum_{i=1}^{m_j} C_{ij} - \lambda \left(\sum_{j=1}^{p} \sum_{i=1}^{m_j} PG_{ij} - \sum_{j=1}^{p} PD_j - PL \right)$$

By applying Lagrange condition the following results are obtained.

$$\partial CC/\partial PG_{ij} = \partial C_{ij}/\partial PG_{ij} - \lambda + \lambda \, \partial PL/\partial PG_{ij} = 0$$

$$ICC_{ij} = \lambda(1-ITL_{ij}) \qquad (5.22)$$

for

$$i = 1,2, \quad \ldots\ldots m_j$$

$$j = 1,2, \quad \ldots\ldots n$$

From equation (5.18) and (5.21)

$$PG_{ij} = a_{ij} + \lambda\, b_{ij}(1-\partial PL/\partial PG_{ij}) + \lambda^2\, c_{ij}(1-\partial PL/\partial PG_{ij})^2 + \ldots$$

and

$$\sum_{j=1}^{p}\sum_{i=1}^{m_j} PG_{ij} = \sum_{j=1}^{p}\sum_{i=1}^{m_j} a_{ij} + \lambda \sum_{j=1}^{p}\sum_{i=1}^{m_j} b_{ij}(1-ITL_{ij}) + \lambda^2 \sum_{j=1}^{p}\sum_{i=1}^{m_j} c_{ij}(1-ITL_{ij})^2 + \ldots \quad (5.23)$$

Let

$$D_j = \sum_{i=1}^{m_j} a_{ij} - PD_j - PL_j$$

$$E_j = \sum_{i=1}^{m_j} b_{ij}(1-ITL_{ij})$$

$$F_j = \sum_{i=1}^{m_j} c_{ij}\,(1-ITL_{ij})^2$$

from equations (5.21) and (5.23)

$$\sum_{j=1}^{p} D_j + \lambda \sum_{j=1}^{p} E_j + \lambda^2 \sum_{j=1}^{p} F_j \ldots\ldots\ldots = 0 \quad (5.24)$$

Let

$$\sum_{j=1}^{p} D_j = D$$

$$\sum_{j=1}^{p} E_j = E$$

$$\sum_{j=1}^{p} F_j = F$$

From equation (5.24)

$$D + E\lambda + F\lambda^2 + \ldots\ldots = 0 \qquad (5.25)$$

When the above equation is second order the roots are

$$\lambda = (-E \pm \sqrt{E*E - 4FD})/2F$$

Once the Lagrange multiplier is known, generations can be calculated. Except for the solution for the equation

(5.25), all other calculations can be done in parallel. So this algorithm is very well suited for distributed and parallel processing. Figure 9 presents the flow chart of the above algorithm.

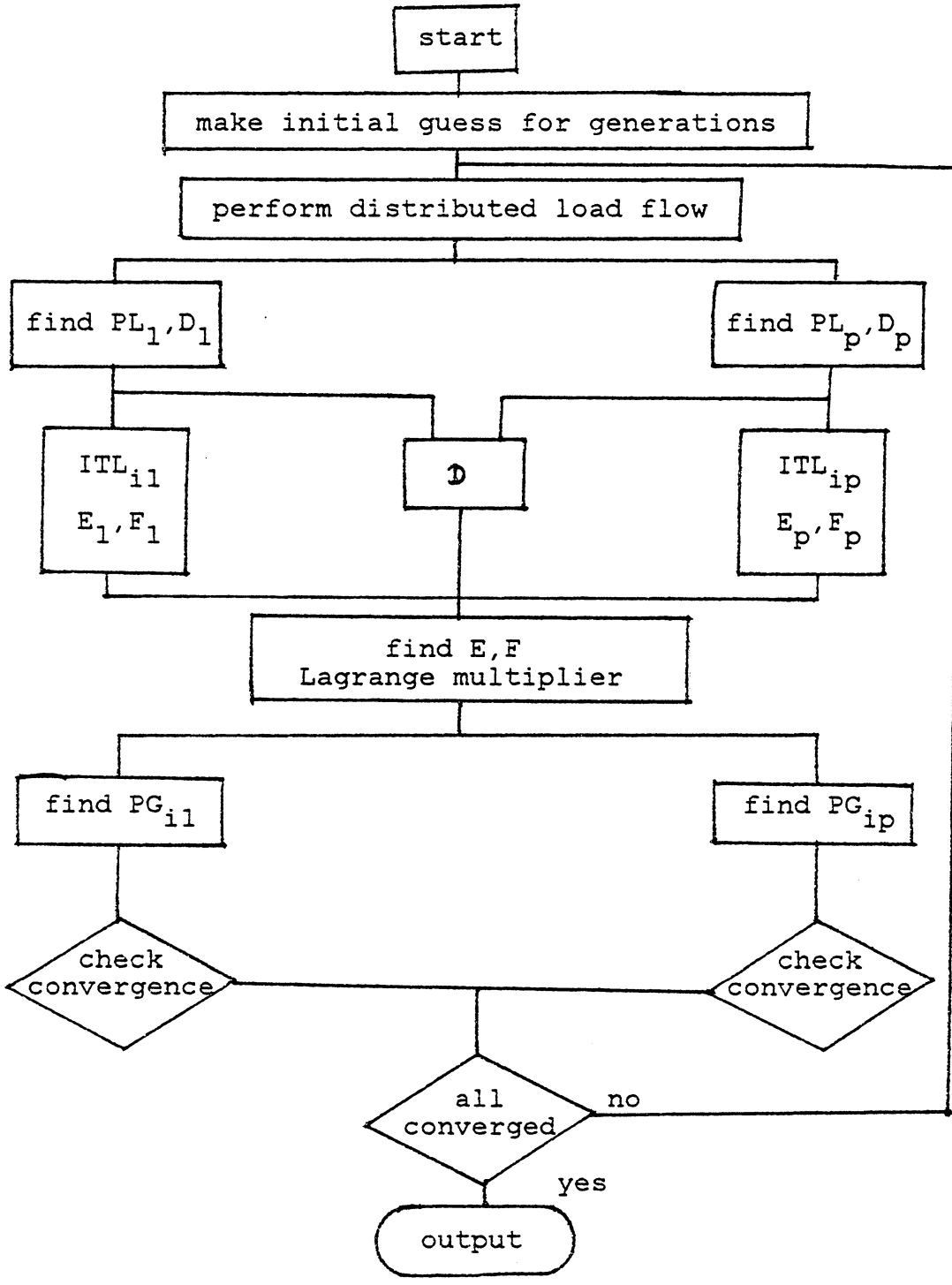

Figure 9: Distributed economic dispatch with losses

Chapter VI

MULTI-MICROPROCESSOR ARCHITECTURE

Several recent developments in mini and microprocessor technology have opened up the idea of multiprocessing. Some of the benefits of multiprocessing are enhanced system performance, improved system reliability, modular system expansion capabilities, and improved system real-time response. For power system problems, for the reasons described in chapter II, multi-microprocessors seem very attractive. It may also be necessary to install more than one processor in a subsystem to get the required throughput or to use the capabilities of special function processors [32-35]. Therefore bus architectures are explained. Different distributed processing architectures using Intel 8080, 8085 and 8086 are presented in this chapter [38,39].

6.1 CHARACTERISTICS OF DISTRIBUTED MICROCOMPUTER SYSTEM

The distributed microcomputer system is one that applies multiple interconnected microcomputers, possibly separated by some distance, to obtain a system solution.

The following are important characteristics of a distributed microcomputing system. They are:

1. Dependency

2. Performance
3. Synchronicity
4. Staticity
5. Operating system and network requirements
6. Cost

6.1.1 Dependency

In order to perform distributed processing, the system has to be first decomposed into subsystems. The subsystems require some data transfer between them in a real system. The less the processor has to know about other processors, the less the dependency. The less the dependency, the better the system. The processors can know about the other systems through communication interfaces. Hence it is an important building block in distributed processing.

In power flow analyses, the equations are nonlinear and they have to be solved iteratively. Hence the subsystems need some data transfer. To minimize data transfer, more than one iteration can be done before each data transfer. The system can also be decomposed using the optimal decomposition technique.

6.1.2 Performance

Overall system performance is affected by the performance of the subsystems. It is therefore important to find out the critical operations in the overall solution. The critical operations can be processed by a dedicated microcomputer. In general, higher performance give rise to higher cost and so it is beneficial to place performance criteria where they are needed.

In power flow analysis, it is better to make the subsystems to have equal number of nodes when the subsystem processors are identical. More than one processor can be installed in a subsystem to increase the throughput. For the most part, it can be decided only after knowing the size of the subsystem and the response time.

6.1.3 Synchronicity

Tight synchronization can increase cost for the same performance. For example, if one processor cannot proceed until another has completely processed the information that was sent, more wait states are introduced and the overall solution will be delayed. Hence it is necessary to find out if tight synchronization is essential for the application.

The power flow analysis requires tight synchronization. It cannot perform the entire analysis without receiving data from the other subsystems.

6.1.4 Staticity

The staticity refers to the degree to which the system configuration changes. In a general case solutions are dynamic. This allows for dynamic interconnections or process creation and deletion. In many distributed microcomputer systems, for example, power flow applications do not require such dynamics and in fact, may incur either cost or performance penalties for their inclusion in the system. There are three types of interprocessor communication links. They are:

1. Run time
2. Initialization time and
3. Configuration time

In the run time method, the communication links are created dynamically during the execution stage. The initialization time method requires that communication links are created when the process is formed. In the configuration time method, the communication links are static and are created when the load module is developed. The static links at configuration time simplifies the processor and memory requirements. For the power flow analysis configuration time static links are sufficient. The initialization links can be used when new subsystems are added to the existing system.

6.1.5 Operating system and network requirements

The operating system must support interprocessor communication and execution. In power system analyses it must also be able to support human interfacing, complex data base, graphics etc.

6.1.6 Cost

The processor is not a very expensive system within a microcomputer system. Hence it seems reasonable to apply many processors if the problem can be decomposed into many separate functions that use a single microprocessor. When compared to the cost of the power system equipments, a microcomputer system is much less expensive. The main advantage of the technique is that the processors are in local control centre. Hence they can be used for on-line control and self diagnosis.

6.2 BUS ARCHITECTURE

There are two types of bus architecture for multiple processor system. They are multiple master / single bus structure and the multibus system. A 'master' is any element existing on the system bus that may take control of the bus. In a multiple master / single bus structure every master utilizes the common bus data path to fetch instructions or data from

memory and I/O. Hence, the common system bus may become the bottleneck for overall system throughput.

The system bus utilization can be minimized by using multibus structure. In this arrangement, each master within the system has its own local memory and I/O that it uses for most of the operations. In this arrangement, during the local operations the individual processors do not require the system bus. This greatly reduces the service request for the system bus. The physical address differentiates the local and global operations. When there is a system bus request, it is granted only when the bus is not in use. The master has to wait when it is in use. It may also happen that more than one master may request the system bus at the same time. Hence this arrangement must provide the technique for bus arbitration.

The bus arbitration can be done by either serial or parallel techniques. The serial technique is called 'daisy chain' and the parallel one is called 'encoded'. In the daisy chain technique, the priority is decided on the basis of bus location. The highest priority master in the system receives the system bus when it needs. In the parallel bus arbitration technique, the priority is decided by the external hardware. In this technique, the system bus is granted to the highly critical processor.

6.3 INTEL 8080 OR 8085 VERSION OF DISTRIBUTED PROCESSING

In this section, distributed processing using two microprocessors is described. The same idea can be extended to include more processors. Intel 8080 and 8085 are software compatible and the control signals used here are available in both.

The schematic diagram given in figure 10 shows the basic building blocks which constitute the system. In this arrangement, there are two microprocessors and some logic blocks. The logic blocks like the 'Interrrupt generating block' and the 'RST instruction generating blocks' can be designed using standard TTL chips.

In this arrangement when the system starts operating, the two processors I and II simultaneously work on their parts of the computation. Once processor I is ready with the data it desires to pass to II it moves the data into the accumulator and the software OUT instruction is executed. The OUT instruction causes the OUT and the specified address line to go high. At this time the INTERRUPT ACKNOWLEDGE signals form processor II is low. All these signals, properly gated together, generate the INTERRUPT signal to processor II.

At this instant the processor II acknowledges the INTERRUPT. This signal enable the 8212 buffer which then jams the RST instruction on to the data bus of processor II.

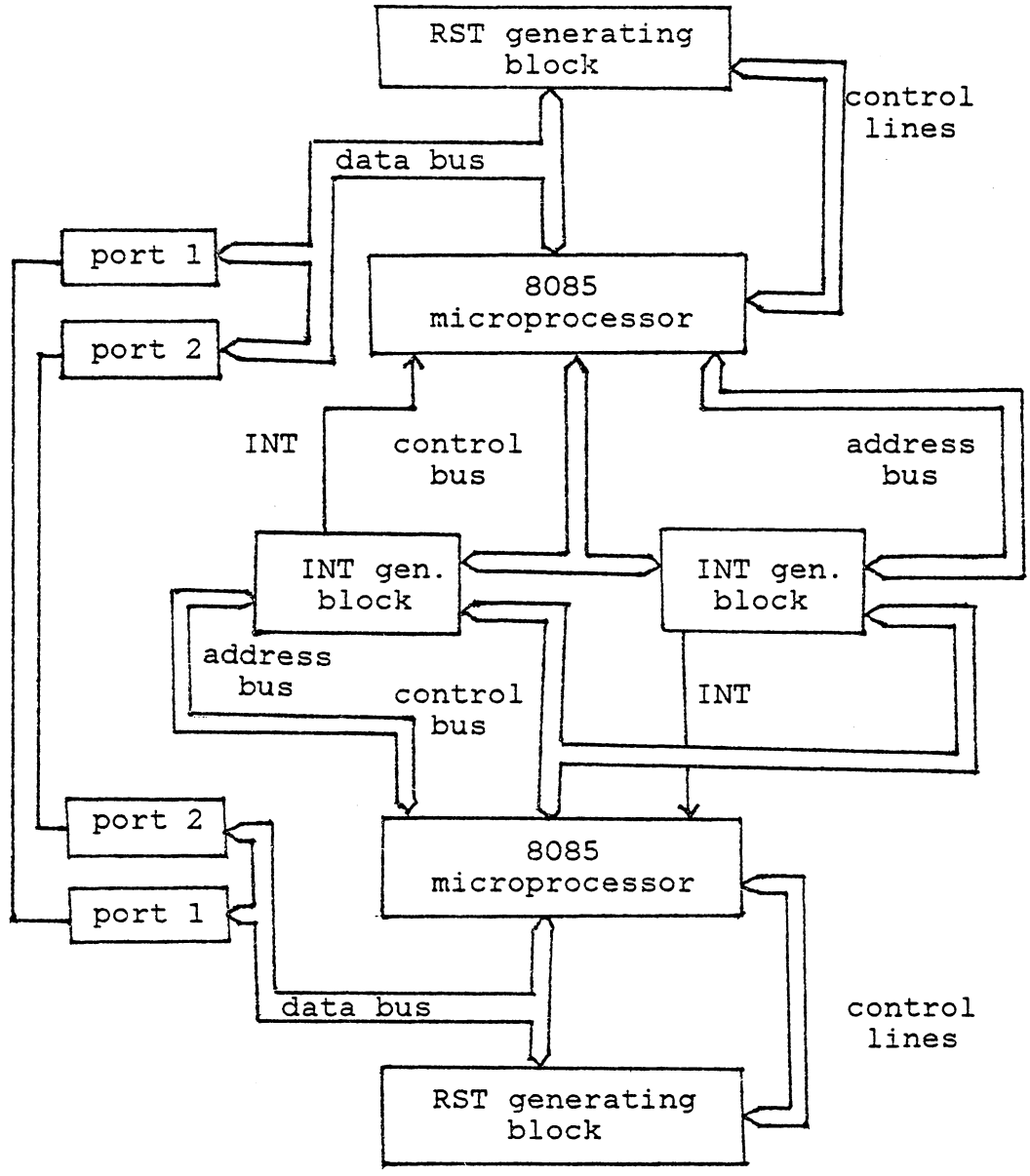

Figure 10: Intel 8085 block diagram for distributed processing

This causes a software branch to location where the interrupt service routine is stored.

The interrupt service routine has an IN instruction to accept the data passed from processor I and then it enables the interrupts again just before returning to its normal operation. The same way processor II can interrupt processor I and pass data.

In this arrangement, whenever data is to be passed, the other processor has to be interrupted. This will add some overhead to the operations. The 8080 or 8085 processor is slow compared to the third generation microprocessors. Hence the next section presents multiprocessing with third generation microprocessors.

6.4 INTEL 8086 VERSION OF DISTRIBUTED PROCESSING

The third generation 16 bit microprocessors were introduced during 1978. They use NMOS high density technology. These third generation devices brought general purpose microprocessors into the performance class of traditional minicomputers. The 'supermicros' are Intel 8086, Zilog Z8000, Motorola 68000 and National semiconductor 16000. In this section, distributed processing using Intel 8086 is described. The idea can be expanded to other third generation processors.

The characteristics of Intel 8086 processor are:

1. Multiprocessing capabilities are inherent in the hardware.
2. Upto one megabyte of memory can be addressed along with a separate 64k byte I/O space.
3. Uses pipelined architecture.
4. Supports coprocessors.
5. System functions are distributed among specialized components.
6. Suitable for modular fashion expansion.
7. Special purpose mathematical chips like the Intel 8087 and the Intel 8089 I/O processor can be used with it.

The Intel 8086 or 8088 processor can be used similar to the Intel 8080 or 8085 processor described in the previous section. The performance of 8086 varies from application to application. In general, the 8086 is roughly seven to ten times more powerful than 8085[39].

Further improvement in performance can be obtained by using 8087-the numeric data processor-as a coprocessor[34]. A coprocessor extends the capabilities of the central processing unit to which it is attached. Both processors execute from a single instruction stream. In the 8087 and 8086 combination, both monitor the same instruction stream and execute selected instructions from it. Figure 11 shows the co-

processor arrangement. The coprocessor increases system throughput because no overhead is incurred in setting up the 8087 for a computation and because the 8086 does not have to wait for results from the 8087. The 8087 can perform data transfer, arithmetic, logical, transcendental, constants and processor control instructions. Table 1 compares the approximate execution time between 8087 and 8086 software emulation of the same instruction[34].

In a distributed processing scheme, it would be advantageous to have a separate I/O processor like Intel 8089 to take care of all I/O operations.

Figure 12 shows the arrangement for a subsystem. In this, the 8086 is a master, the 8087 is a coprocessor and the 8089 is an I/O processor. Every subsystem in a system will have a similar configuration. Further, these subsystems can be easily expanded by adding extra processors to the local subsystem.

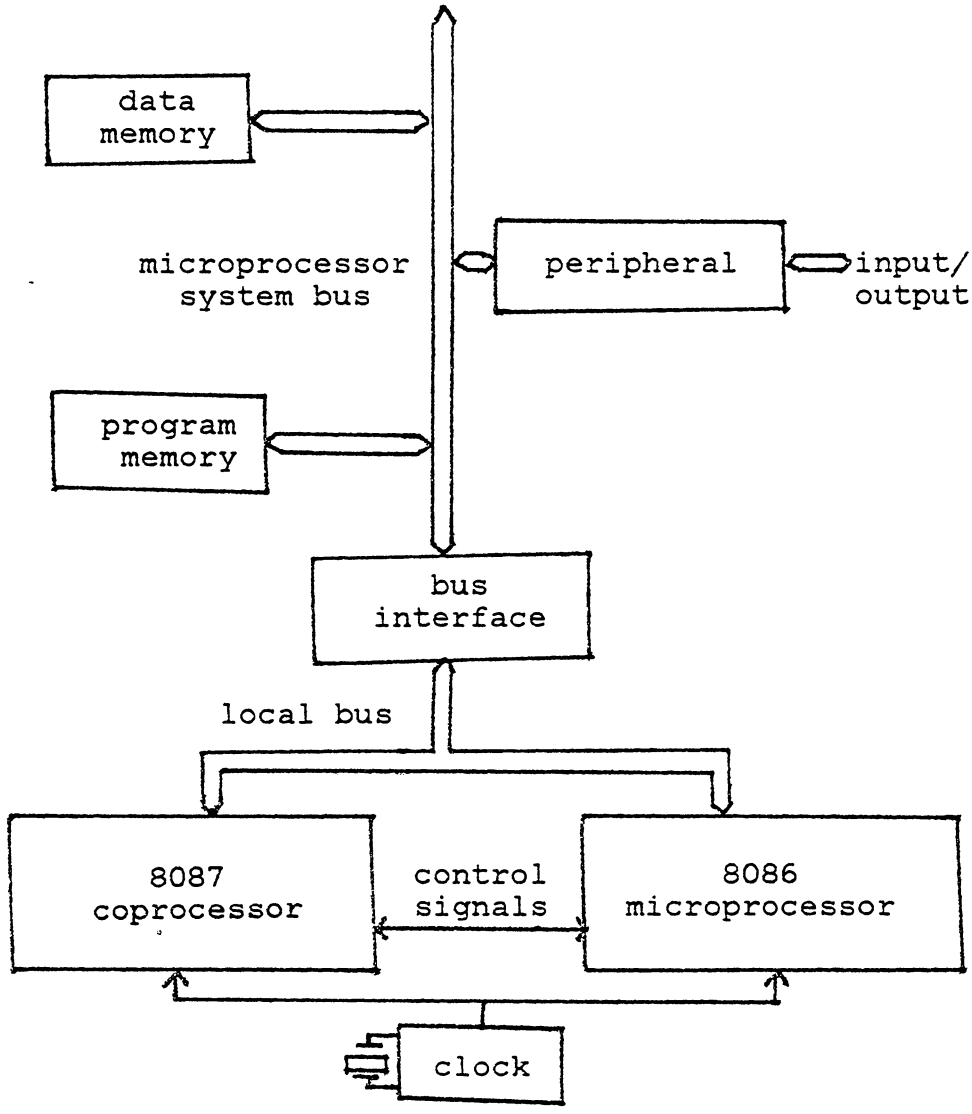

Figure 11: Intel 8087 coprocessor arrangement

TABLE 1

Intel 8087 and 8086 comparison

Instruction	Approximate execution time in micro sec.	
	8087 (5MHz clock)	8086 emulation
Add	14	1600
Multiply	18	1600
Divide	39	3200
Square root	36	19600
Tangent	110	13000
Raise to the power	130	17100

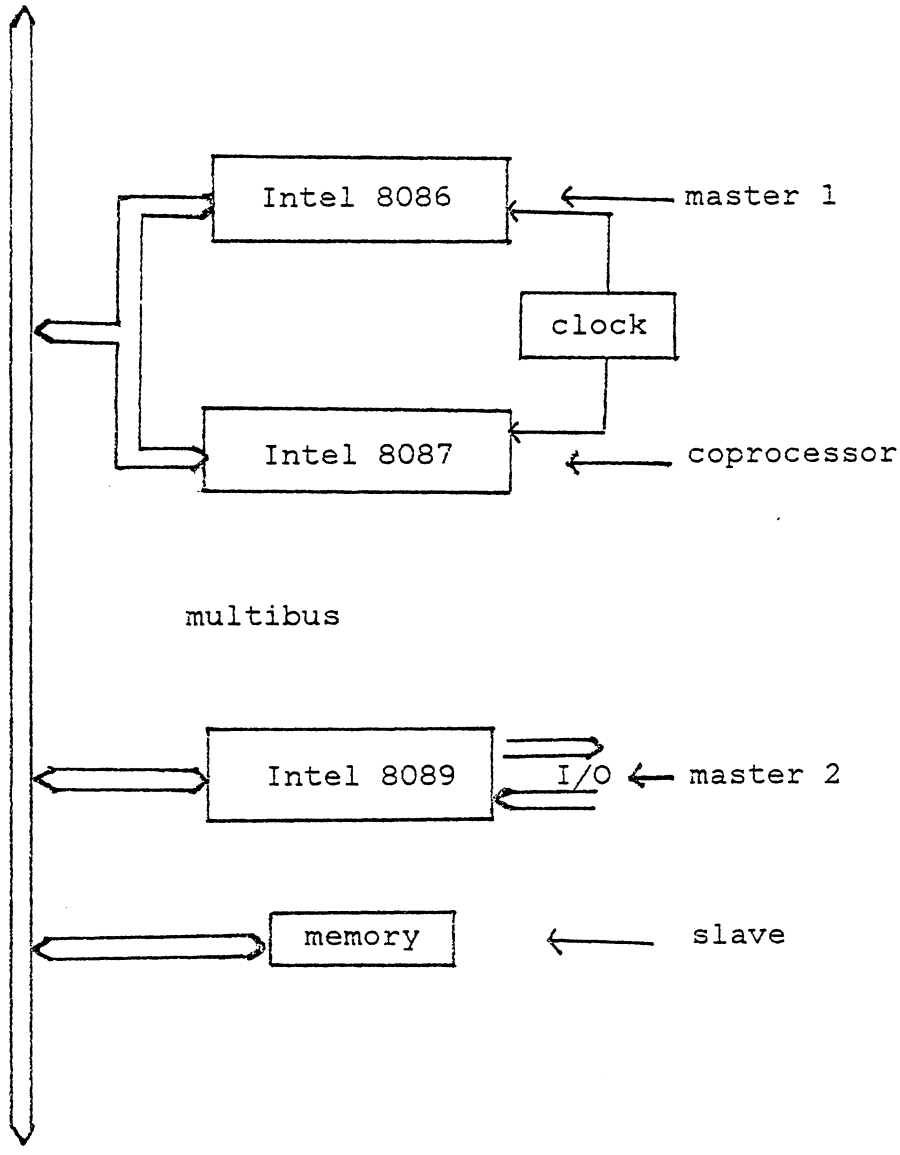

Figure 12: Intel 8086 version of distributed processing

Chapter VII

RESULTS

In this chapter, different test cases are studied to test the algorithms presented in chapter 3,4 and 5. The first section presents the distributed processing simulation results for IEEE 30 and 57 bus systems. The next section presents the microprocessor load flow resluts for 5 and 6 bus systems. In the third section, a distributed microprocessor system is studied using the 6 bus case. The optimal decomposition technique presented in chapter 4 is applied to an IEEE 14 bus test system and results are discussed in section four. Finally, section five discusses the improved economic dispatch algorithm as applied to the IEEE 14 bus system. It is also compared with different nonlinear programming techniques.

7.1 DISTRIBUTED PROCESSING SIMULATION RESULTS

A simulation program was written to perform the distributed load flow on the IBM 370/158 computer. The simulation program simulates multiprocessing in a serial processor. The program executes all the subnetworks one after the other in a serial fashion. A timing subroutine was used to determine the times required to execute the subnetworks between the

data passes. To obtain the multiprocessing time the maximum time is calculated. In this analysis, the communication time between the subsystems are not taken into account.

A test case consisting of IEEE 30 and 57 bus test systems shown in figures 13 and 14 was studied. The line data and the loads are presented in reference [36]. The systems are made into different subnetworks and studied. Figures 15 and 16 show the computer execution time versus the number of subnetworks. It is clear from the figures that it is advantageous to solve the subsystems parallely to increase the throughput. It is also clear that the time does not decrease linearly as the number of processors increase. This is because the total number of iterations increase as the subsystems is made smaller and smaller. This is shown in figures 17 and 18. In this analysis, the Gauss-Seidal technique is used in the subsystems. Hence the entire process is the combination of Gauss and Gauss-Seidal techniques. If there are n buses and n processors then the technique used will be equivalent to the Gauss iteration technique. When there is only one processor, then the technique used is Gauss-Seidal.

In the next test, systems are divided into two subnetworks. Instead of passing the data after every iteration, the data is passed after a few iterations within the subsystems. From the results in figure 19 and 20 it is clear that

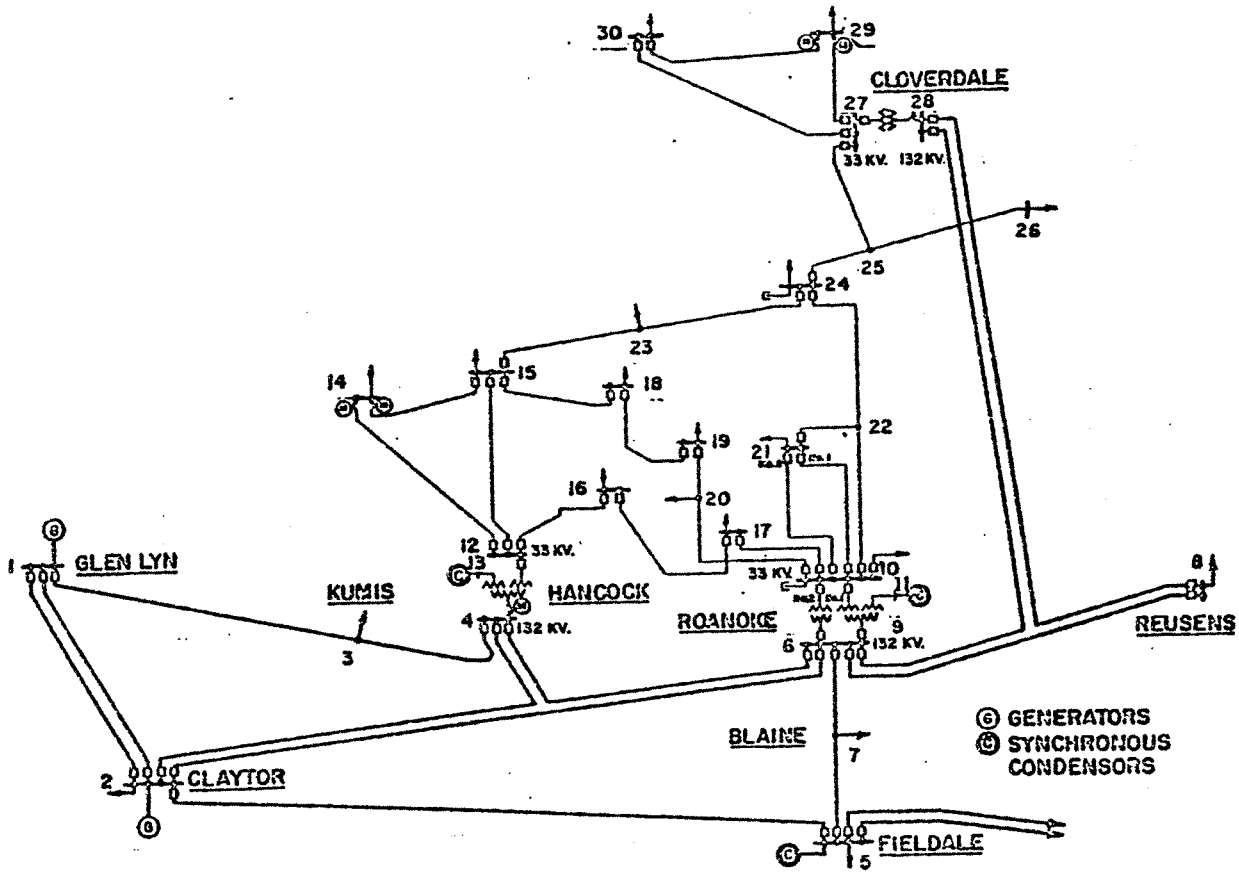

Figure 13: IEEE 30 bus system diagram

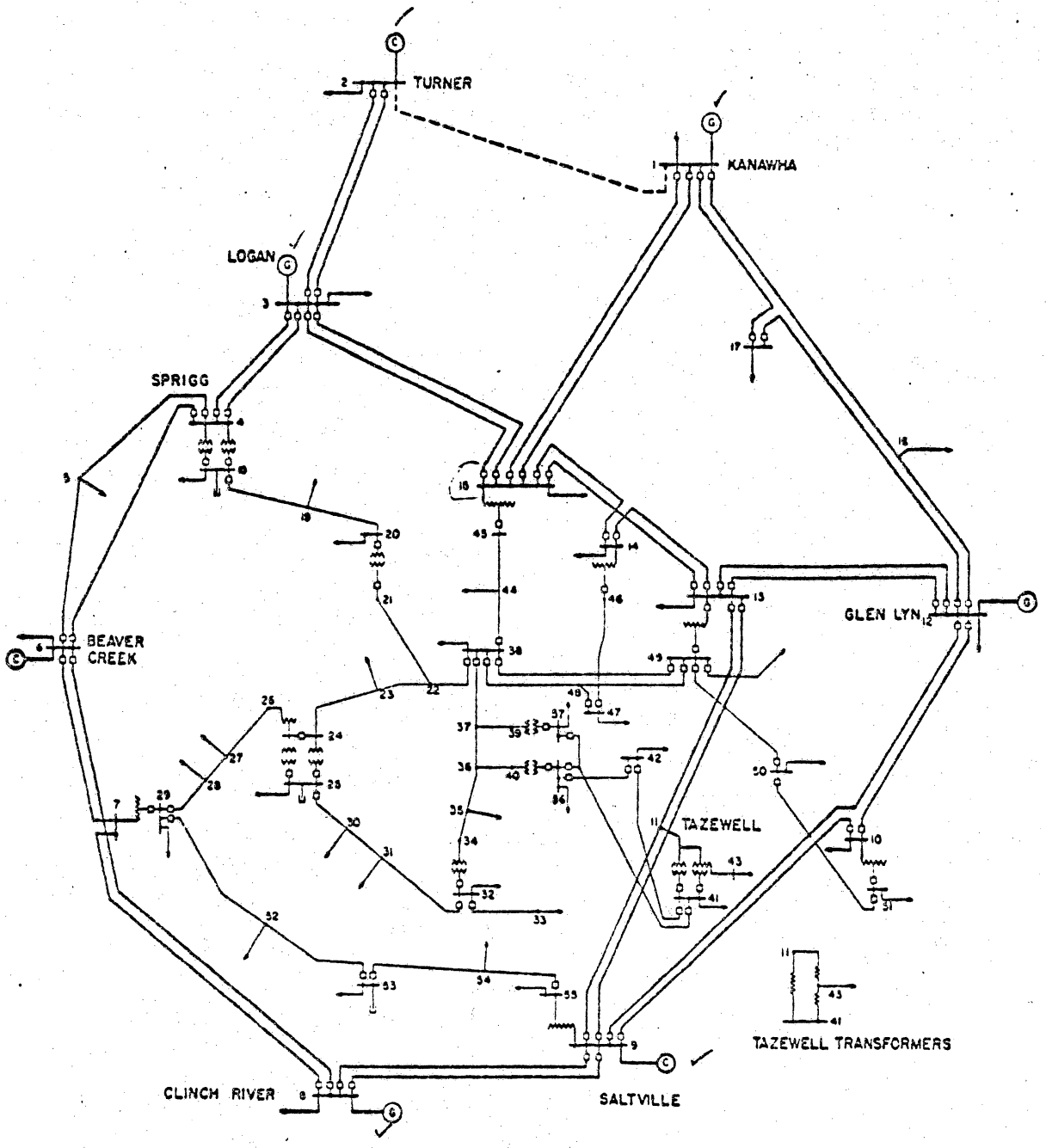

Figure 14: IEEE 57 bus system diagram

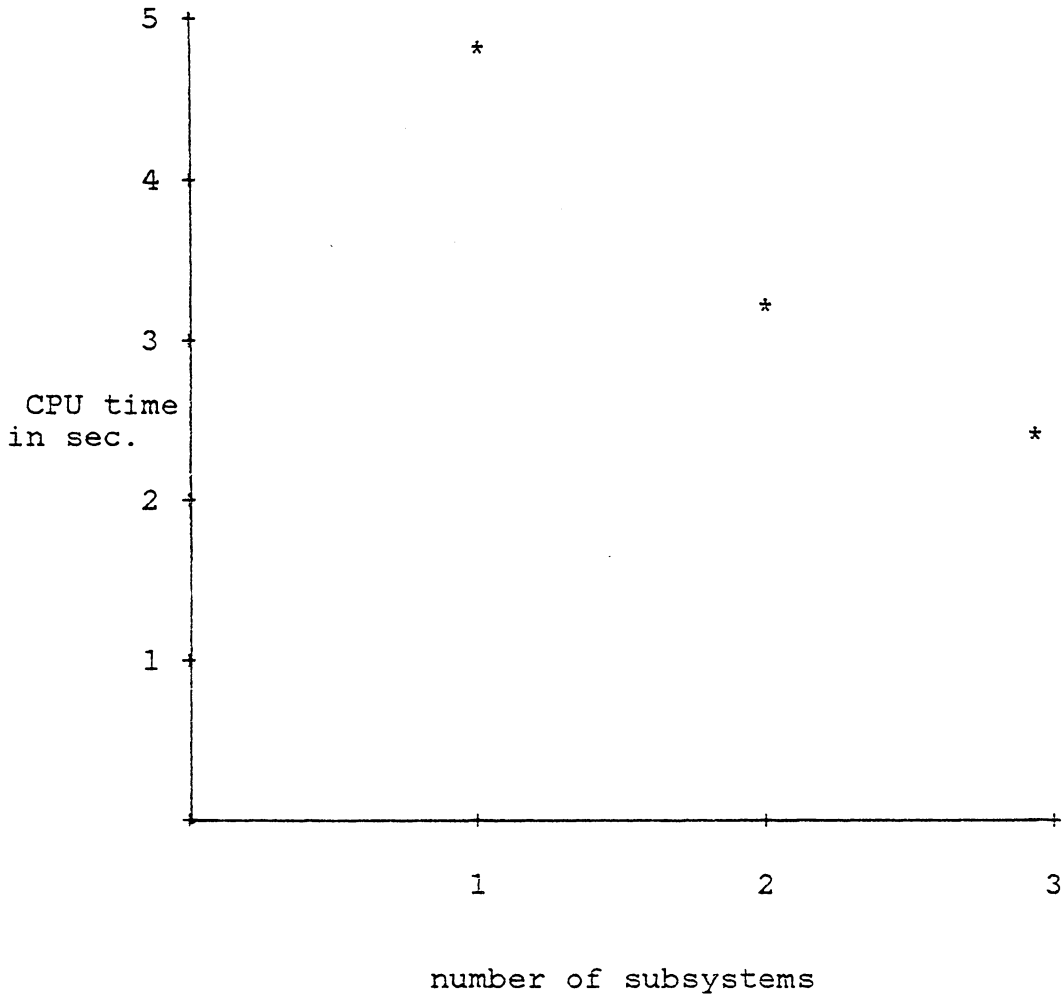

Figure 15: CPU time versus no. of subsystems for the 30 bus system

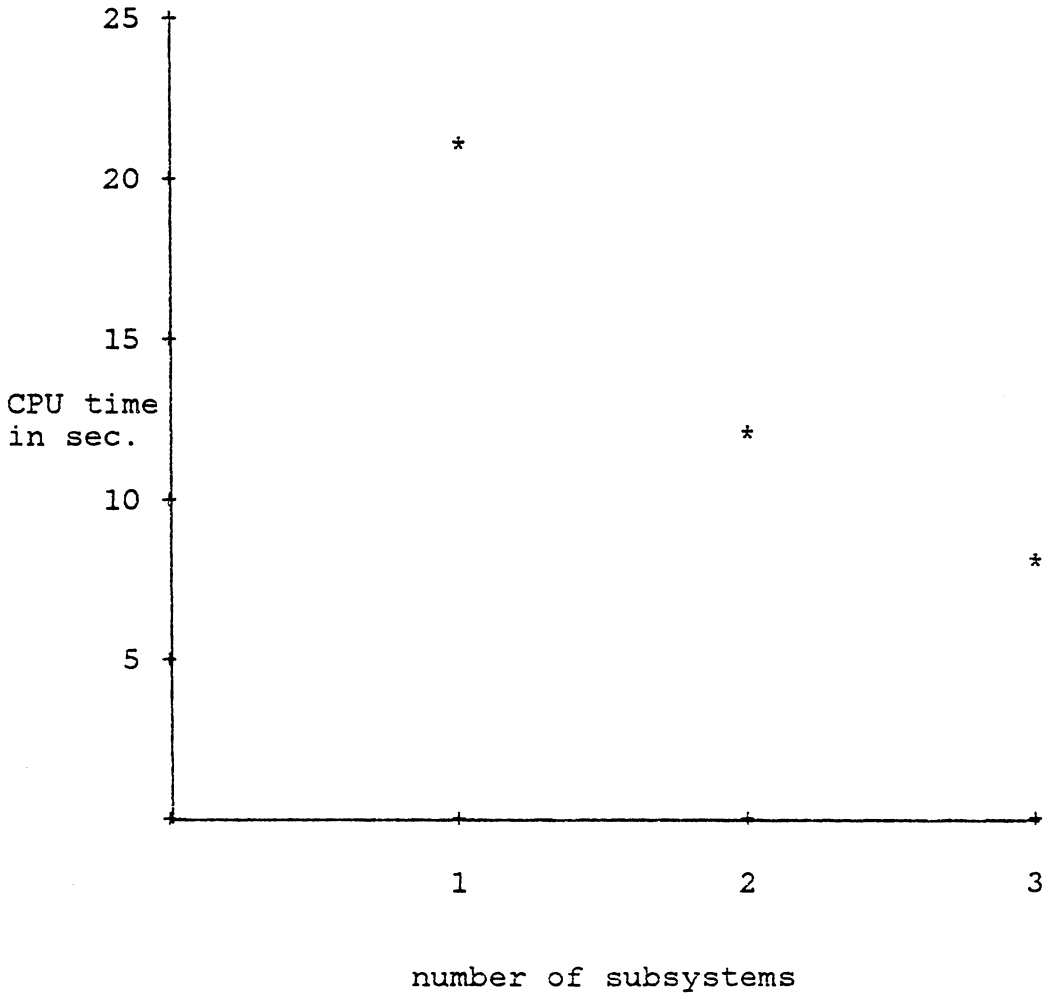

Figure 16: Execution time versus number of subsystems for the 57 bus system

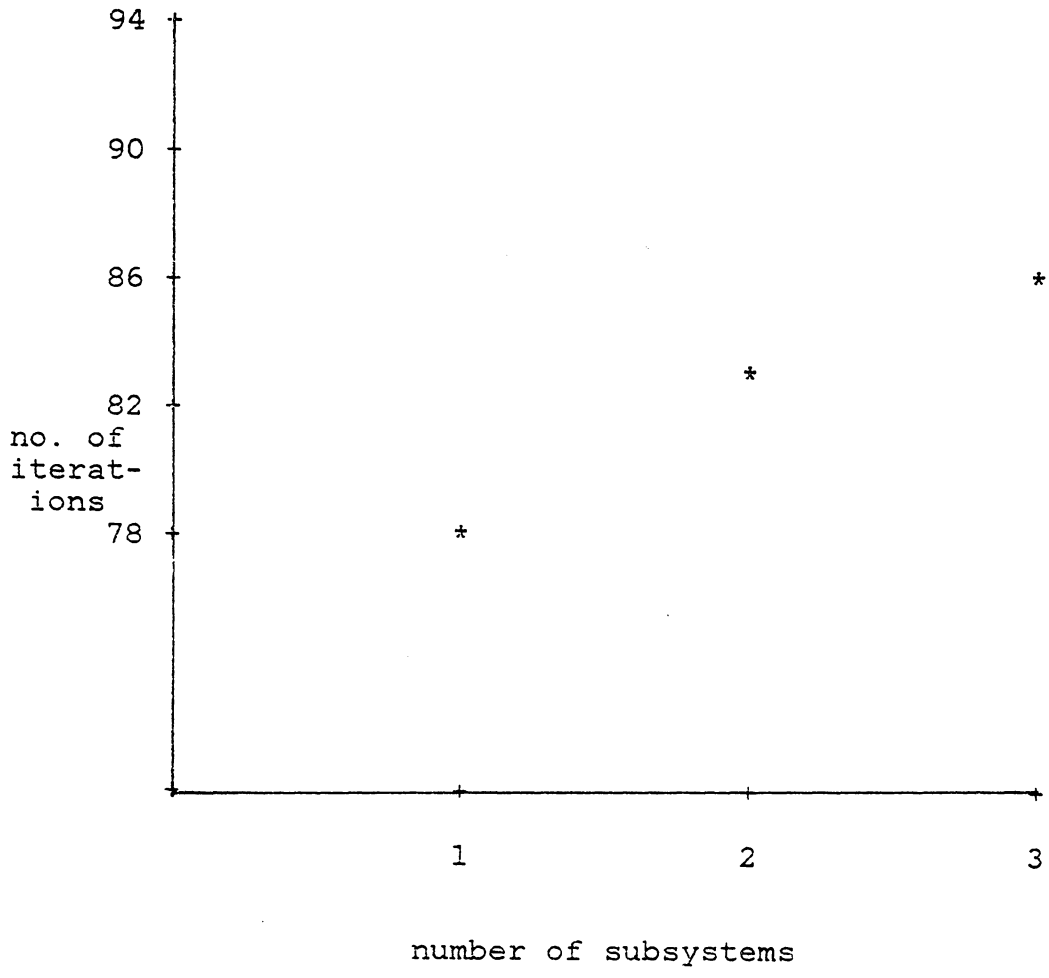

Figure 17: Number of iterations versus number of subsystems for the 30 bus system

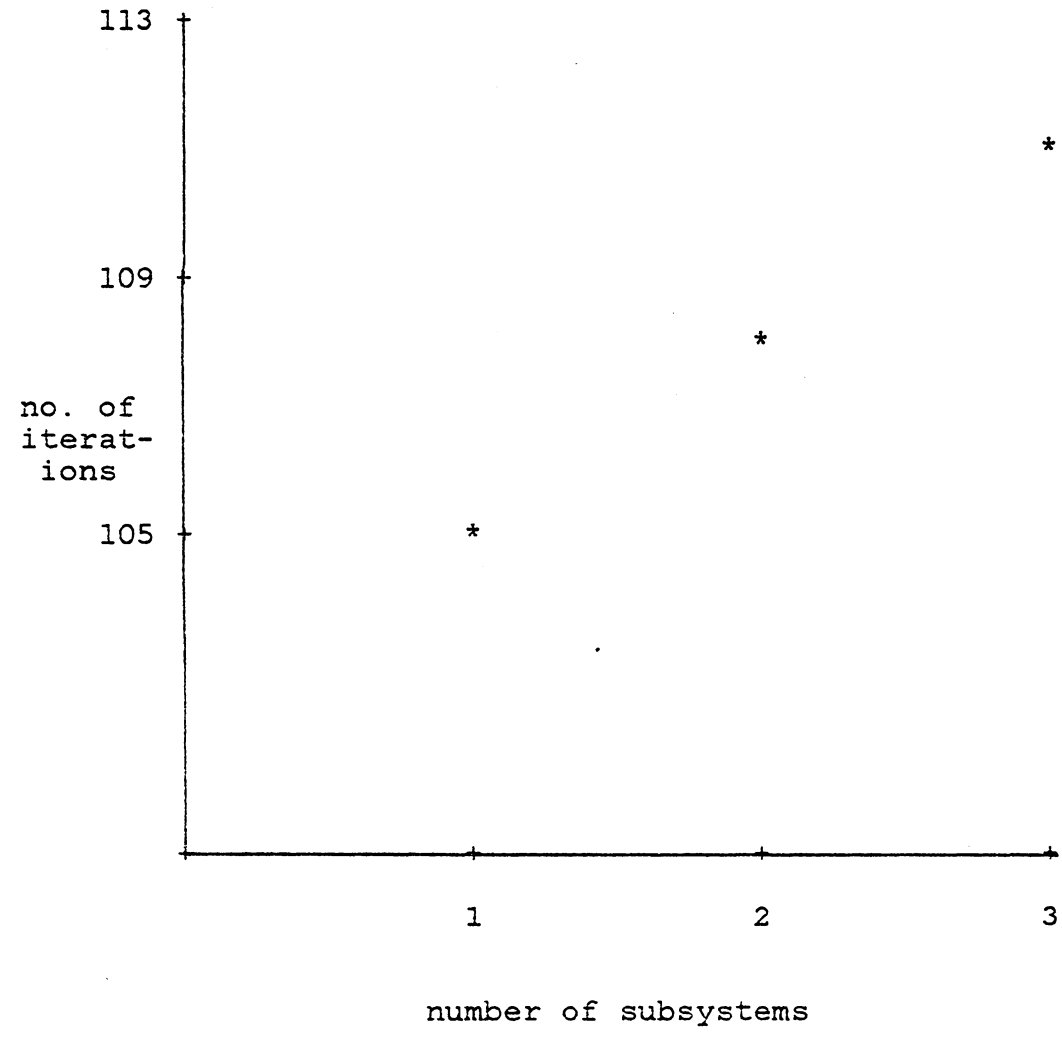

Figure 18: Number of iterations versus number of subsystems for the 57 bus system

in some cases it is advantageous to do more than one iteration before receiving the data from the other subsystems. The major advantage of the technique is that it requires less data passes and this is shown in figures 21 and 22. Hence, communication and overhead time can be reduced.

The figures 23 to 26 show the results for a three subsystem case. These results are similar to the two subsystem case. It is also interesting to note that there is no advantage in iterating on the local subsystem many times before receiving the data from the other subsystems. After some number of iterations within the subsystem, the total number of data passes will remain approximately constant. Hence, there is no real advantage to doing this.

The speedup of the loadflow algorithm for the 30 bus and 57 bus systems is shown in figures 27 and 28. In this case, data is passed after every iteration. From the results it is clear that speedup for these cases is greater than one. The efficiency of the algorithm is presented in figures 29 and 30.

These two factors, speedup and efficiency, give the measure of how good the multiprocessing algorithm is compared to a serial processing one. From the simulation results, it is clear that the algorithm presented in chapter 4 for distributed power flow analysis is feasible and improves the throughput.

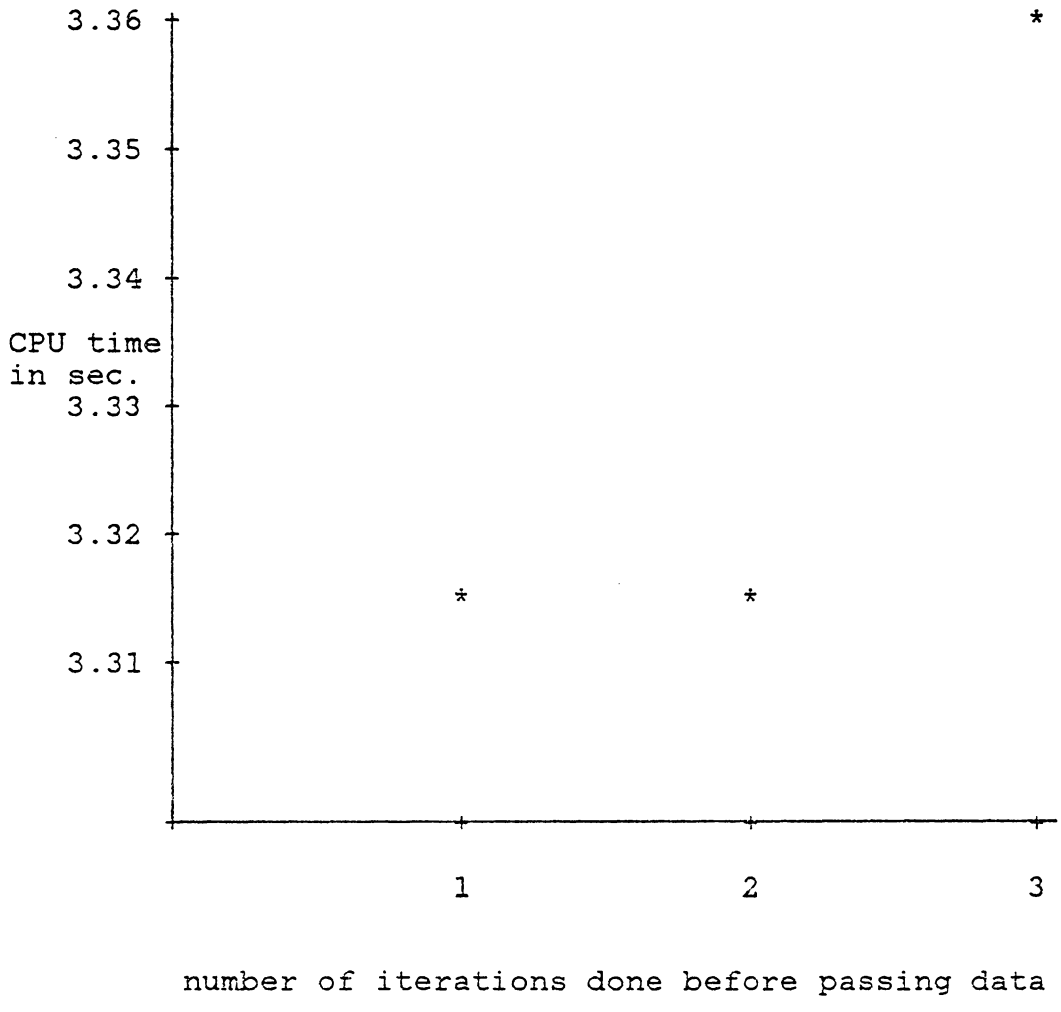

Figure 19: CPU time versus no. of iterations in a two subsystem 30 bus system

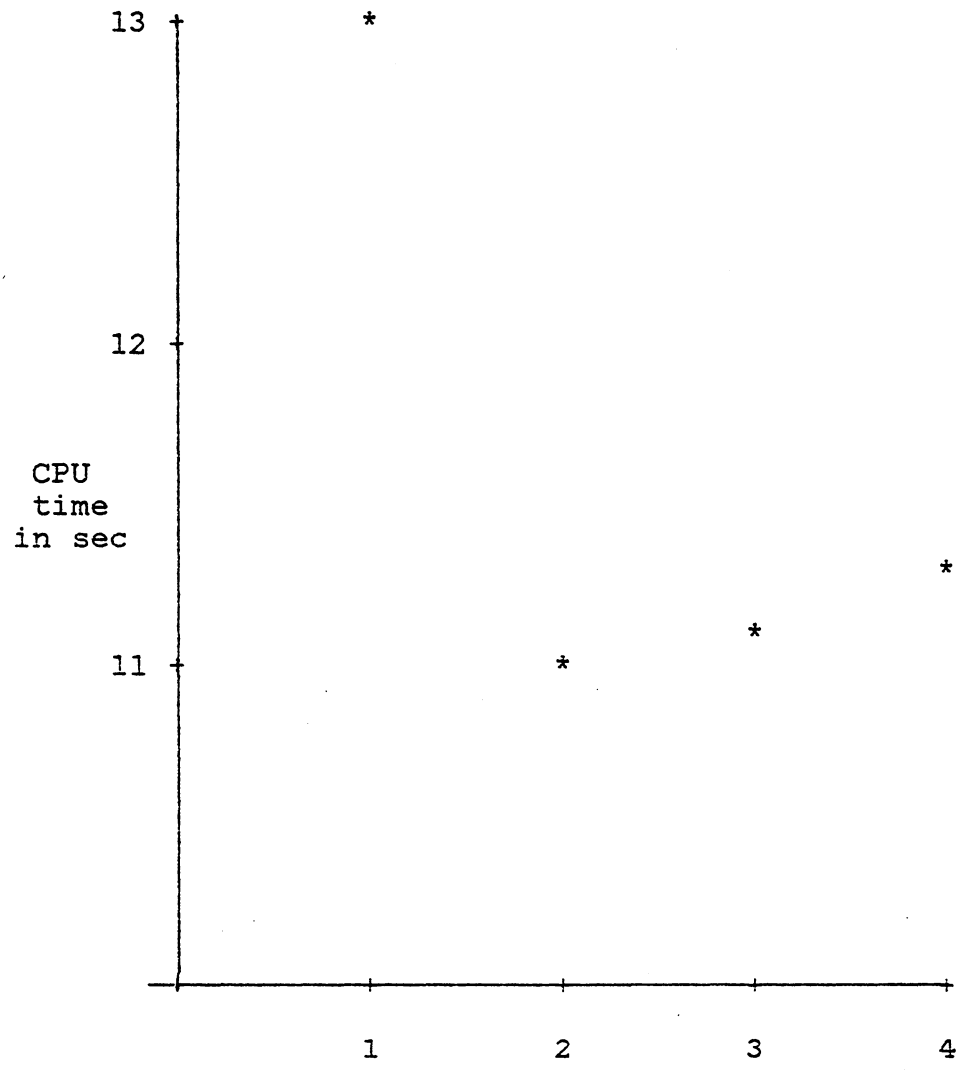

Figure 20: CPU time versus no. of iterations in a two subsystem 57 bus system

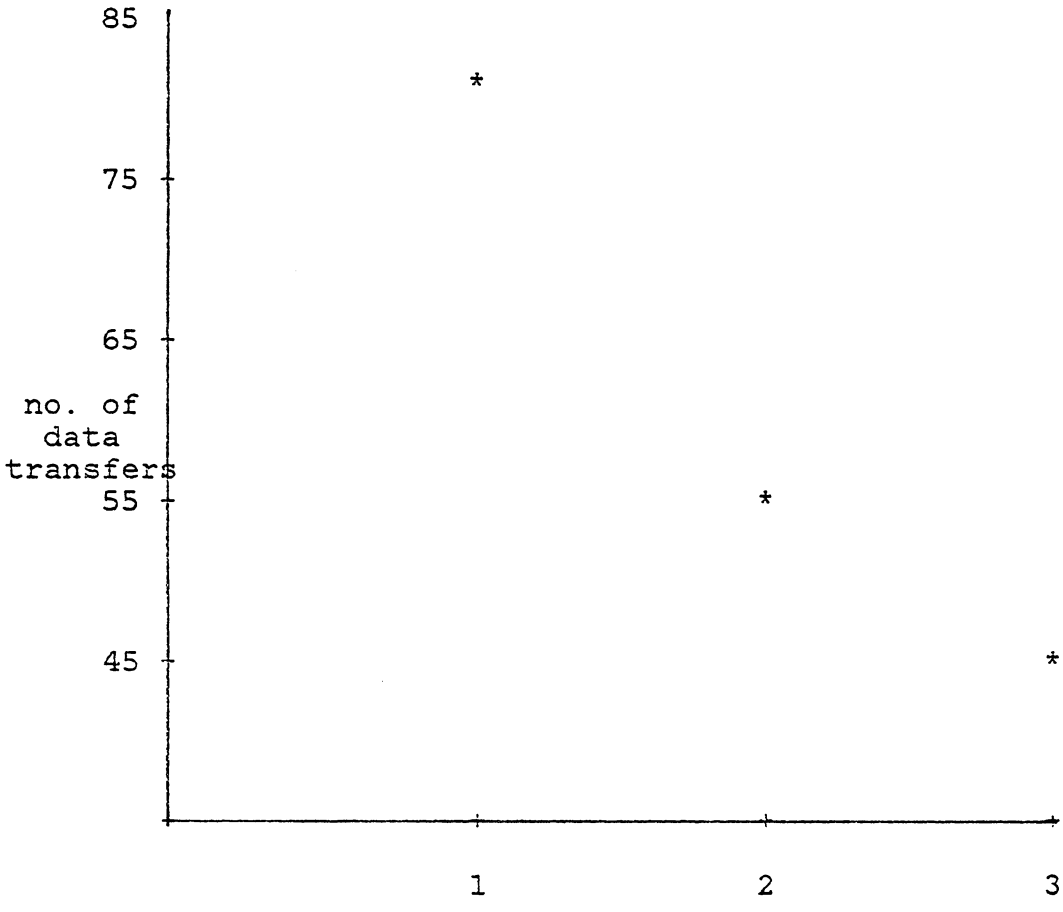

Figure 21: No. of data transfers for a 2 subsystem 30 bus system

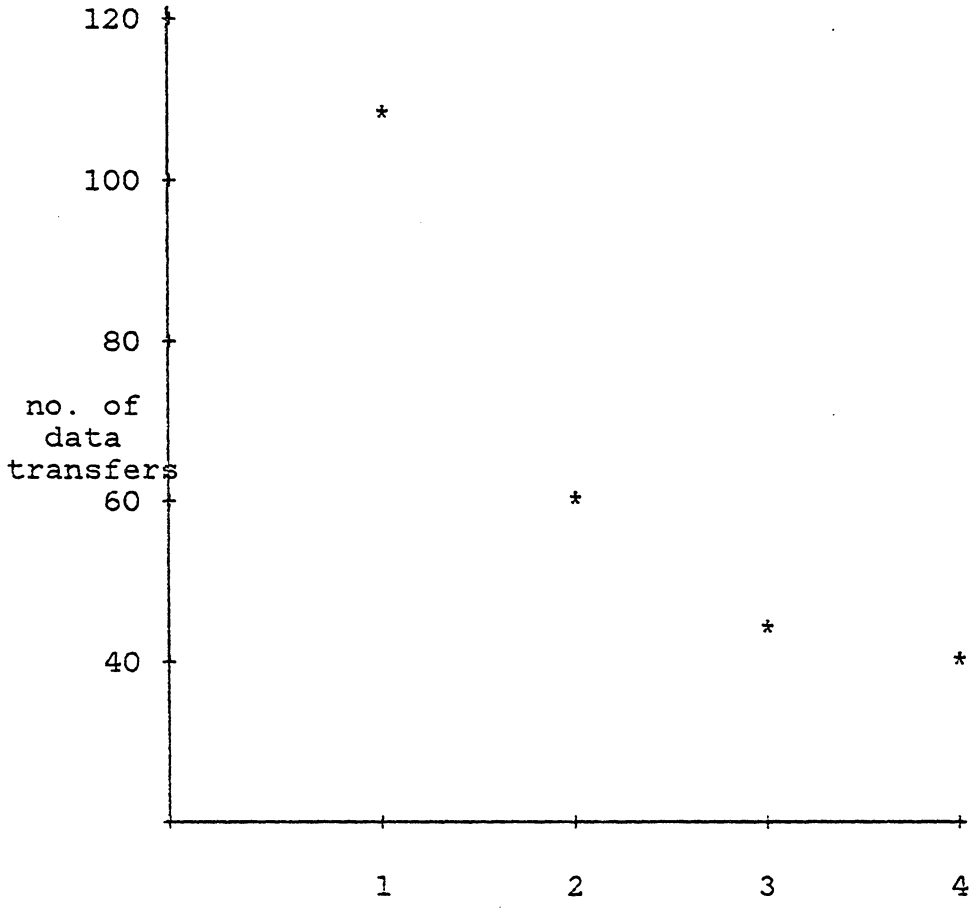

Figure 22: No. of data transfers for a two subsystem 57 bus system

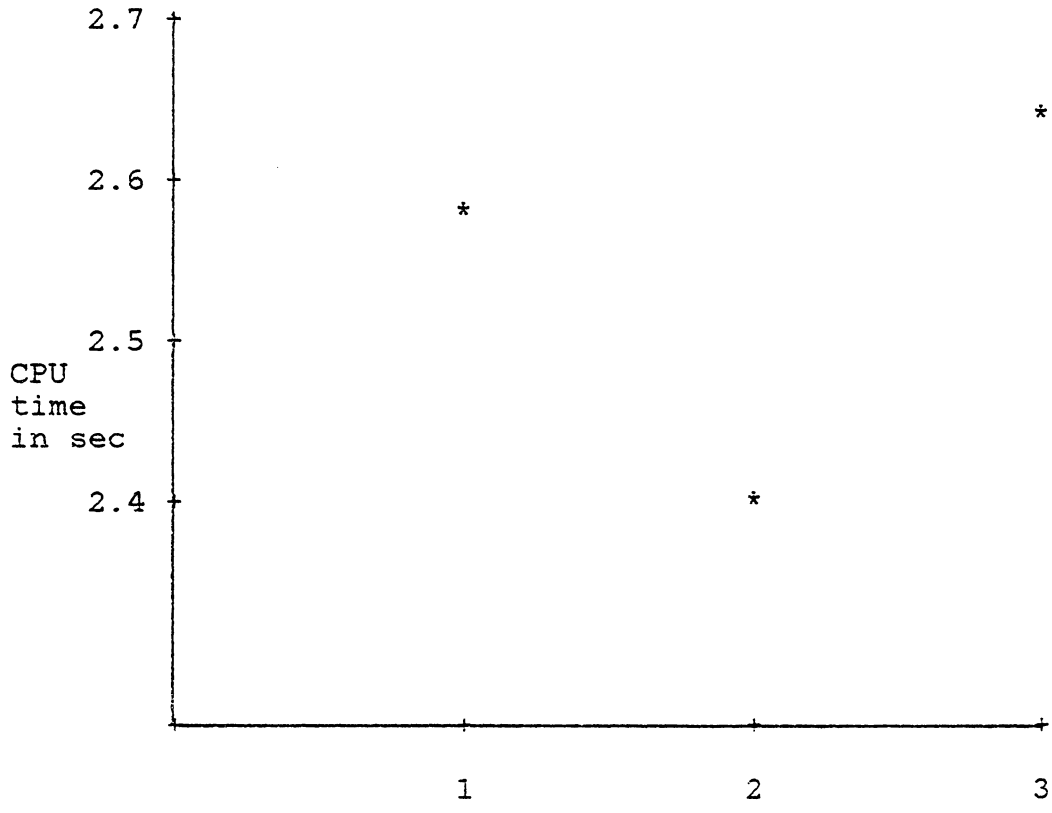

Figure 23: CPU time versus no. of iterations done before passing data for a three subsystem 30 bus system

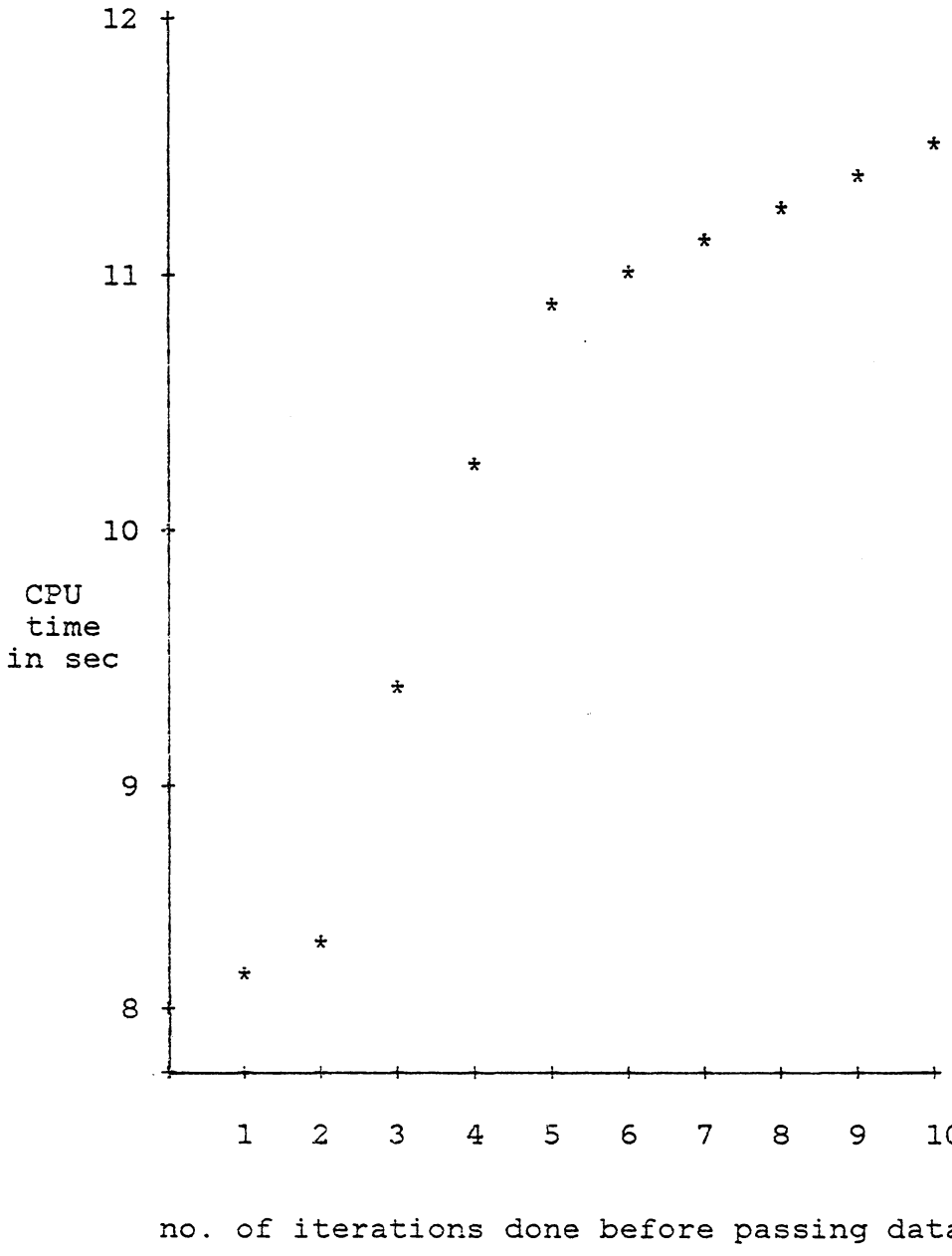

Figure 24: CPU time versus no. of iterations done before passing data for a 3 subsystem 57 bus system

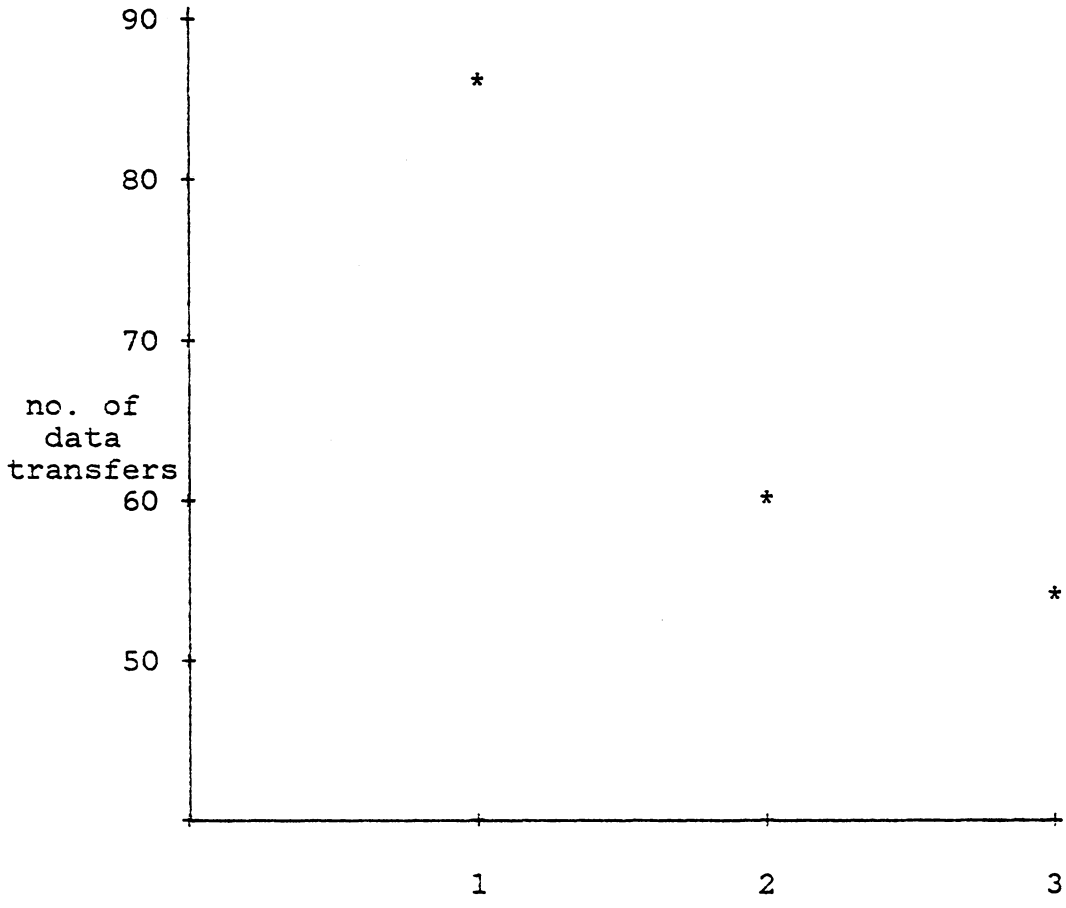

Figure 25: No. of data transfers versus no. of iterations done before passing data for a 3 subsystem 30 bus system

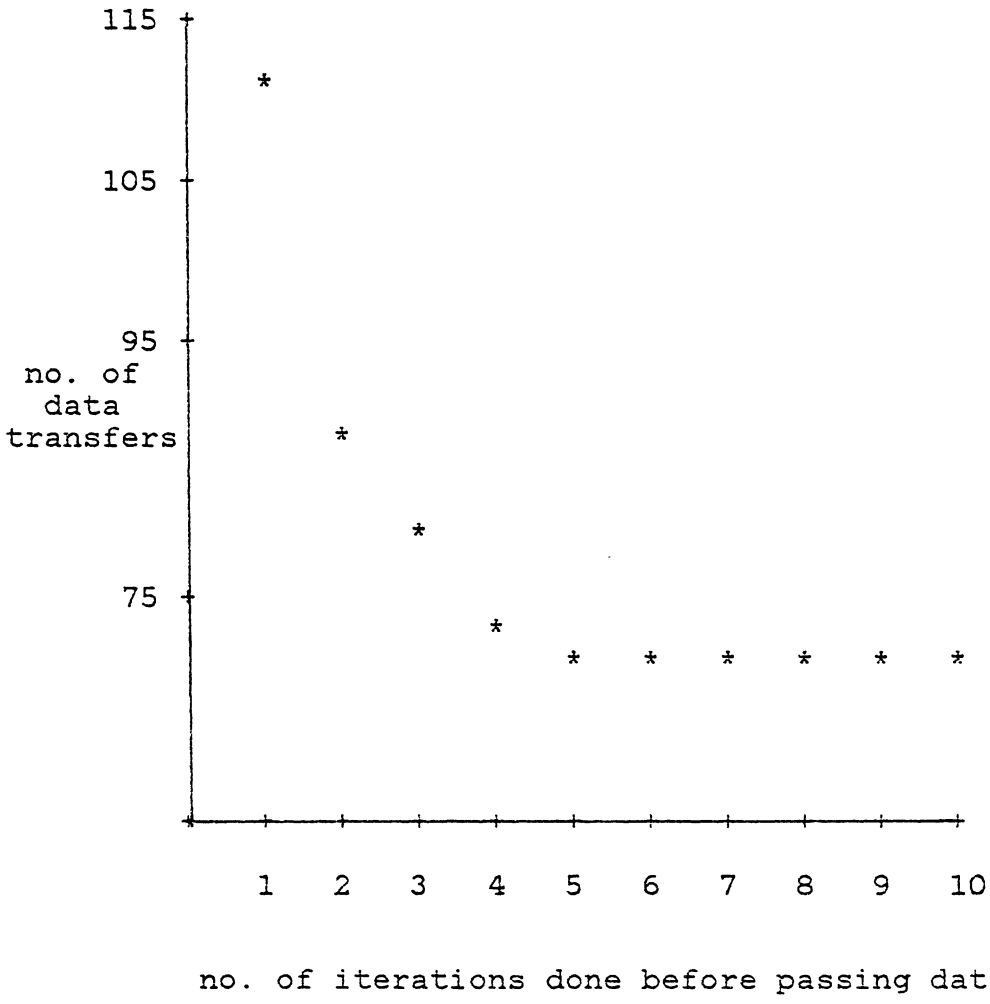

Figure 26: No. of data transfers versus no. of iterations done before passing data for a 3 subsytem 57 bus system

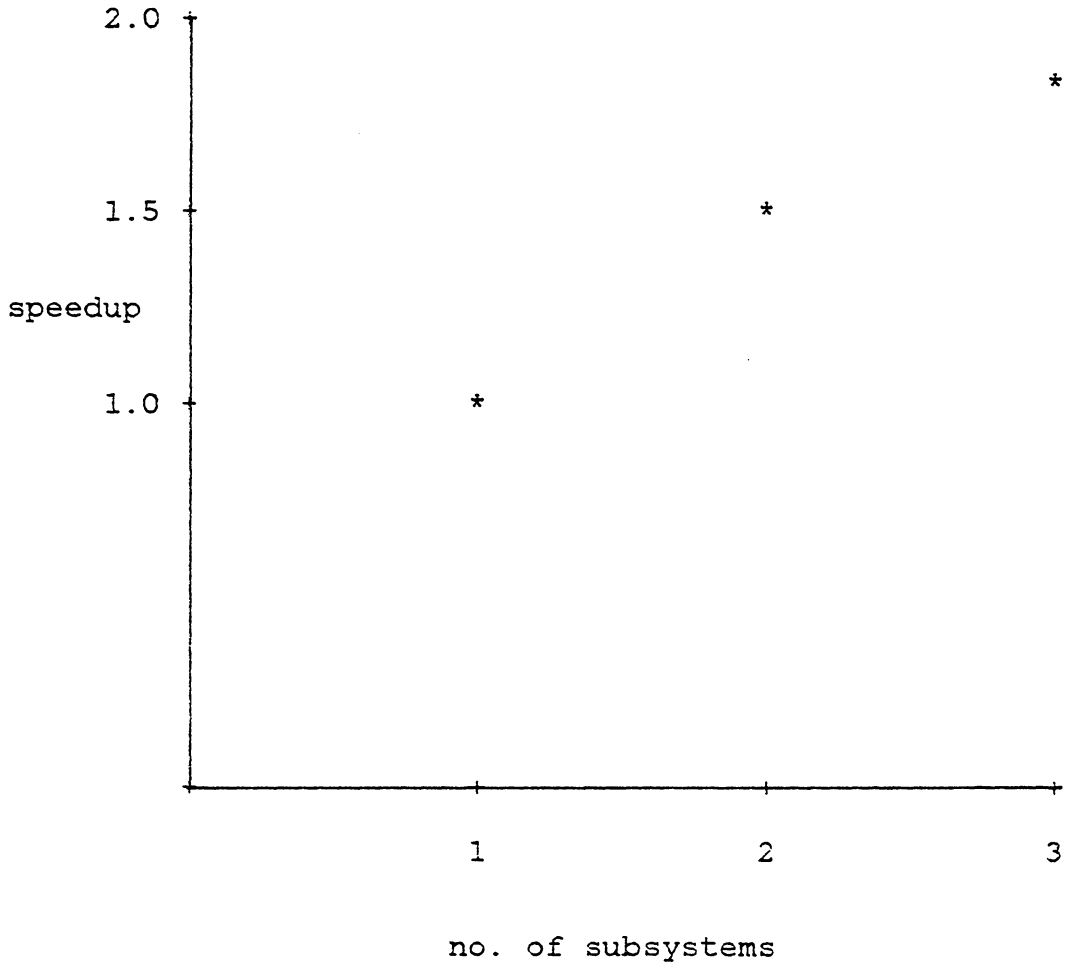

Figure 27: Speedup for the 30 bus system

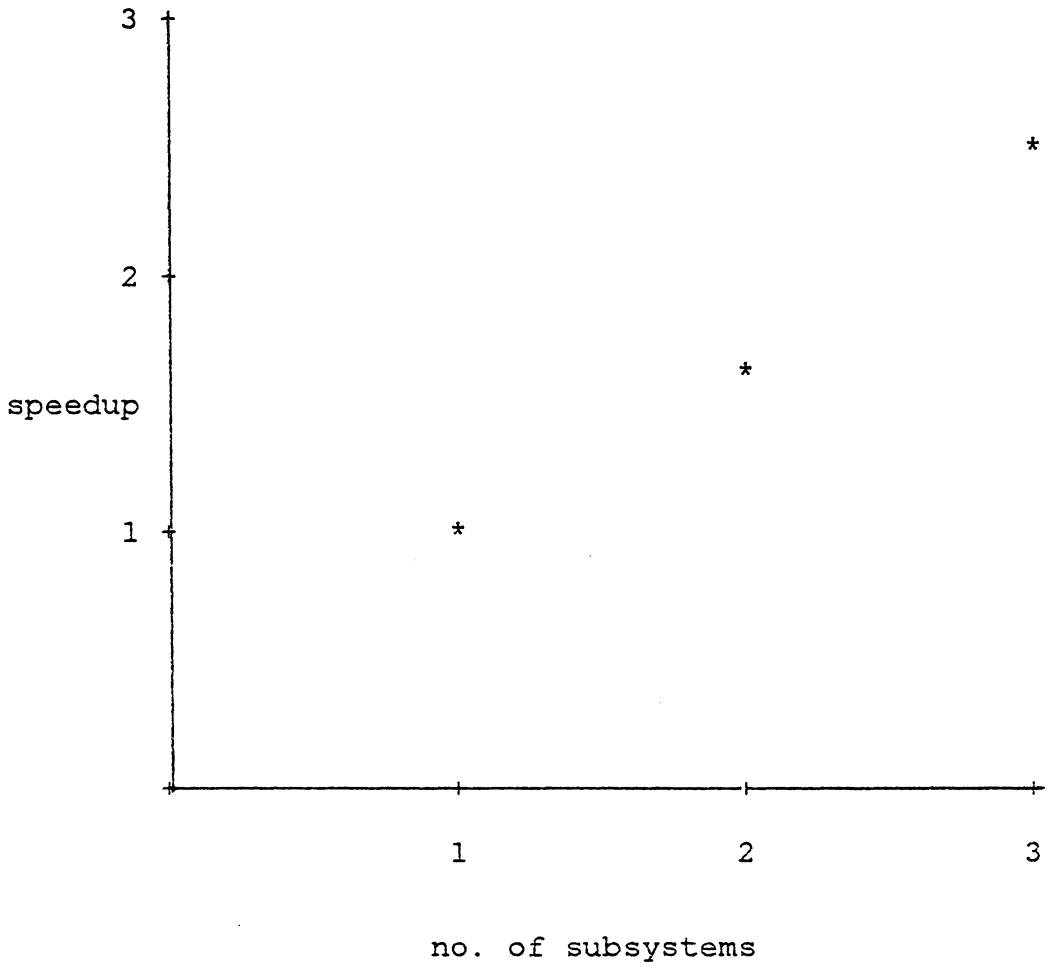

Figure 28: Speedup for the 57 bus system

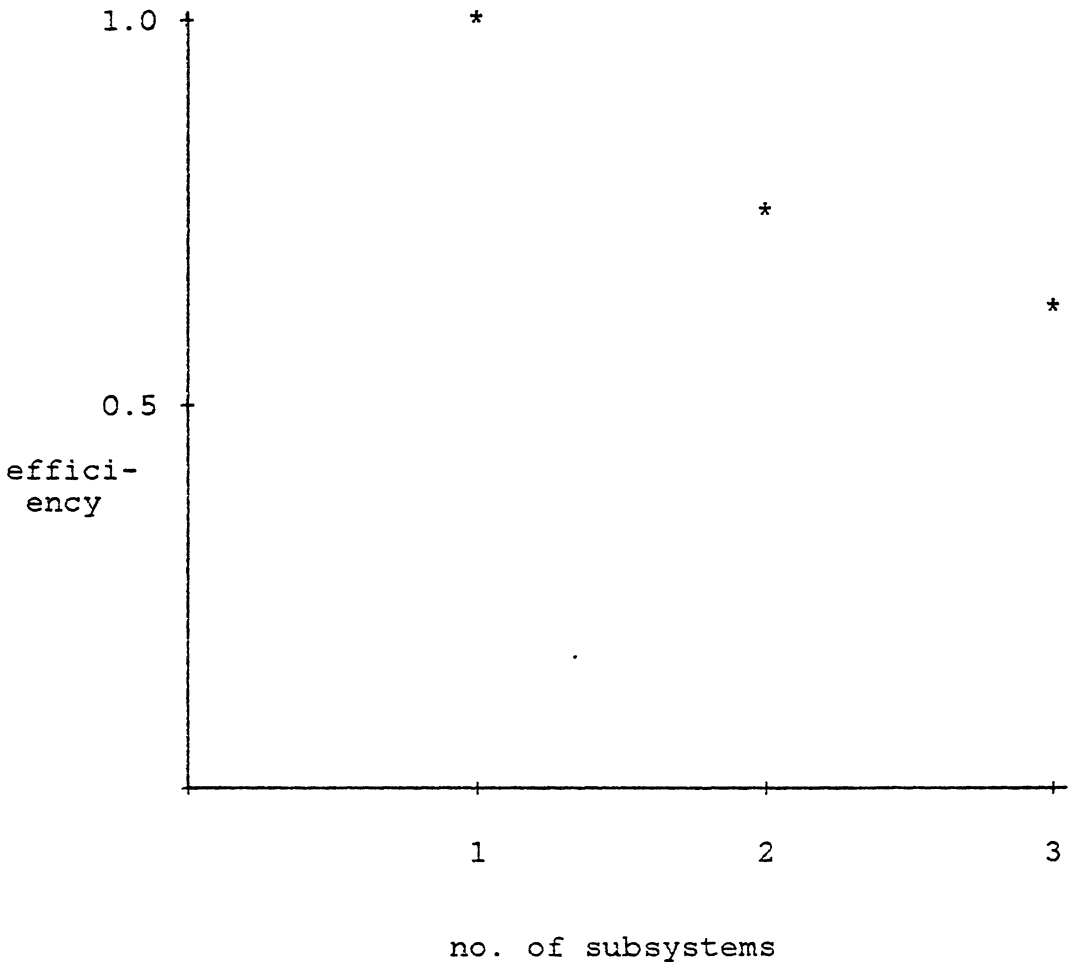

Figure 29: Efficiency of the 30 bus system

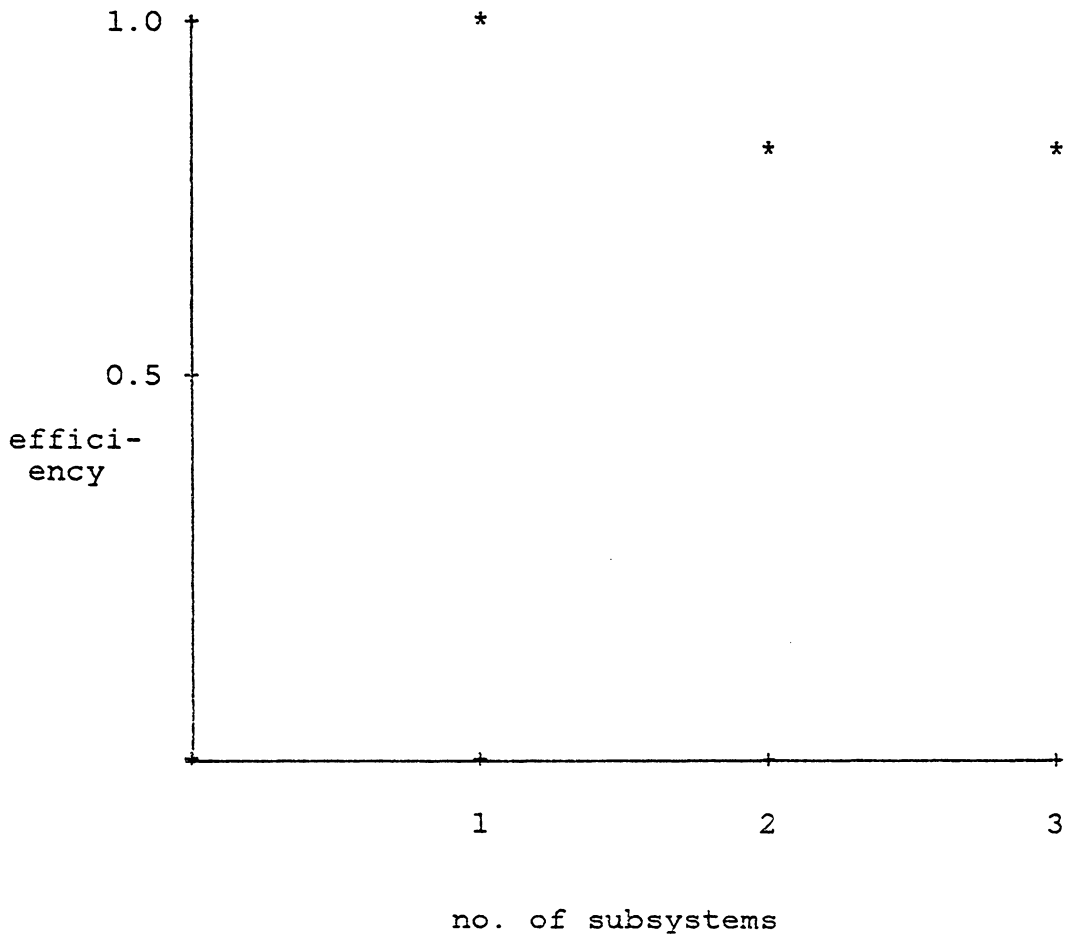

Figure 30: Efficiency of the 57 bus system

7.2 RESULTS FOR MICROPROCESSOR LOAD FLOW

The microprocessor load flow results for 5 bus and 6 bus systems are presented in this section. Figures 31 and 32 show the system arrangement. The line data and loads for the 5 bus system are given in reference [23]. The Intel 8085 emulator output results in per-unit values are presented in table 2. The line data and loads for the 6 bus system[41] are presented in tables 3 and 4. The load flow results in per-unit values are given in table 5.

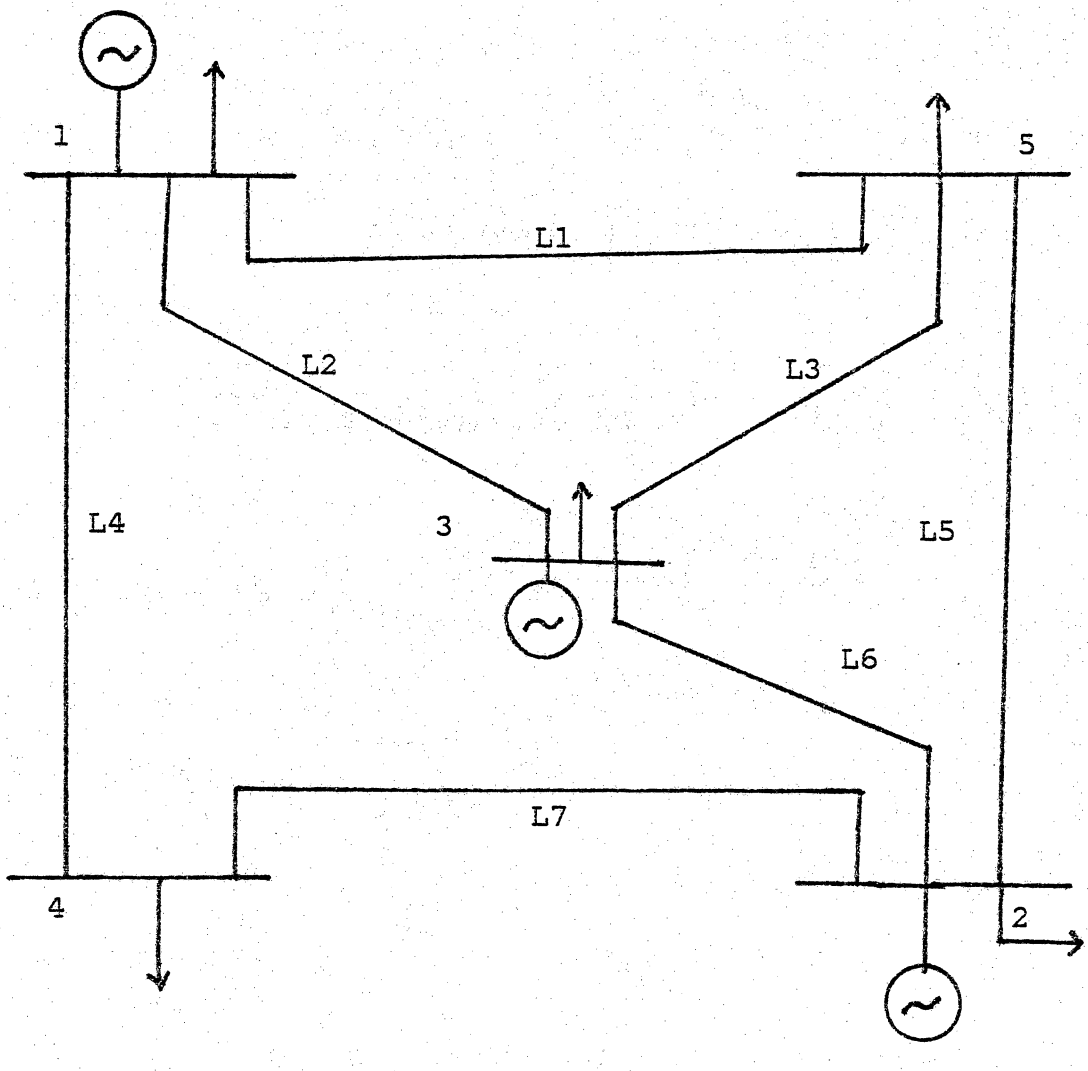

Figure 31: 5 bus system diagram

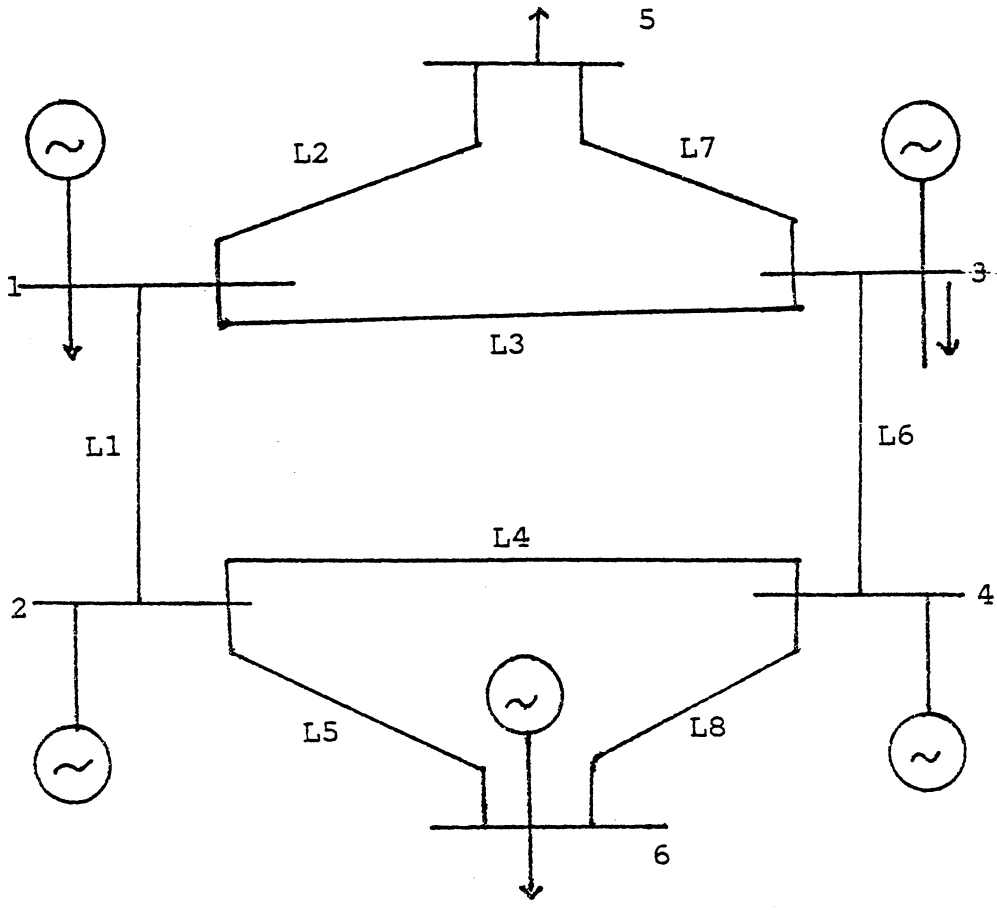

Figure 32: 6 bus system diagram

TABLE 2

Load flow result for the 5 bus system

Bus no.	Voltage		Power	
1	1.0500	0.0000	1.0466	1.0829
2	0.9203	-0.1145	-2.1100	-0.7600
3	1.0170	0.0776	1.5516	0.2366
4	0.8178	-0.2604	-1.6100	-0.4200
5	1.0182	0.0590	1.2723	0.6298

TABLE 3

Line data for the 6 bus system

Line	Line designation From	To	Series impedance (per unit)		Shunt admittance (per unit)	
1	1	2	0.00063	0.00446	0.0	0.001047
2	1	5	0.00063	0.00452	0.0	0.001066
3	1	3	0.00057	0.00441	0.0	0.001047
4	2	6	0.00068	0.00446	0.0	0.001028
5	2	4	0.00063	0.00441	0.0	0.001047
6	3	6	0.00057	0.00436	0.0	0.001025
7	5	3	0.00063	0.00457	0.0	0.001047
8	6	4	0.00063	0.00446	0.0	0.001047

TABLE 4

Load flow data for the 6 bus system

Bus number	Load MW	Load MVar
1	90.0	40.0
2	0.0	0.0
3	120.0	70.0
4	140.0	50.0
5	100.0	50.0
6	150.0	80.0

TABLE 5

Load flow result for the 6 bus system

Bus no.	Voltage		Real power	Reactive power
1	1.0000	0.0000	0.5258	0.5744
2	1.0461	0.0894	1.2970	0.9067
3	0.9295	-0.0299	0.6300	0.0022
4	0.8761	-0.1233	-1.4000	-0.5000
5	0.8577	-0.1398	-1.0000	-0.5000
6	0.9498	0.0117	0.0300	-0.2991

7.3 DISTRIBUTED MICROPROCESSOR LOAD FLOW

The distributed processing algorithm is implemented in the Intel 8085 microprocessor. The algorithm is implemented in Fort/80 and machine language. The 6 bus system shown in figure 32 is tested using the microprocessor. The 6 bus system is made into two subsystems each having 3 buses. The subsystem I has buses 1, 3 and 5. The subsystem II has buses 2, 4 and 6.

To time the algorithm, the processor is run with the 2 MHZ clock. Intel has a 10 MHZ version in the market. All the results are converted to the 10 MHZ version.

When the system is solved as a complete system it takes 11 iterations and requires 23.54 seconds. But when it is solved parallely as two subsystems, it takes 12 iterations and requires 13.68 seconds. In this case, the number of iterations is increased by one but the time is decreased by approximately half. Various cases were studied using different values for the number of iterations before a data pass. Figures 33 and 34 show the execution time and the number of data passes for these cases.

All the floating point and trigonometric functions were carried out in software. The software algorithms are very slow compared to hardware circuits. The power flow algorithms require a large number of floating point operations.

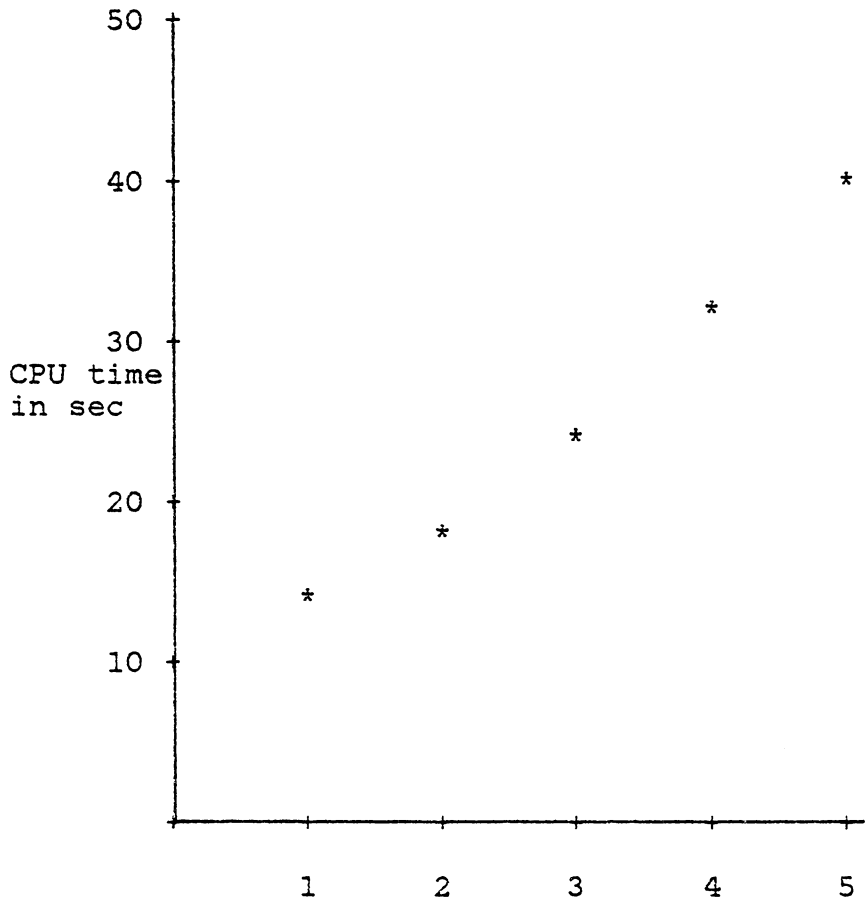

Figure 33: CPU time versus no. of iterations before passing data for a two subsystem 6 bus system

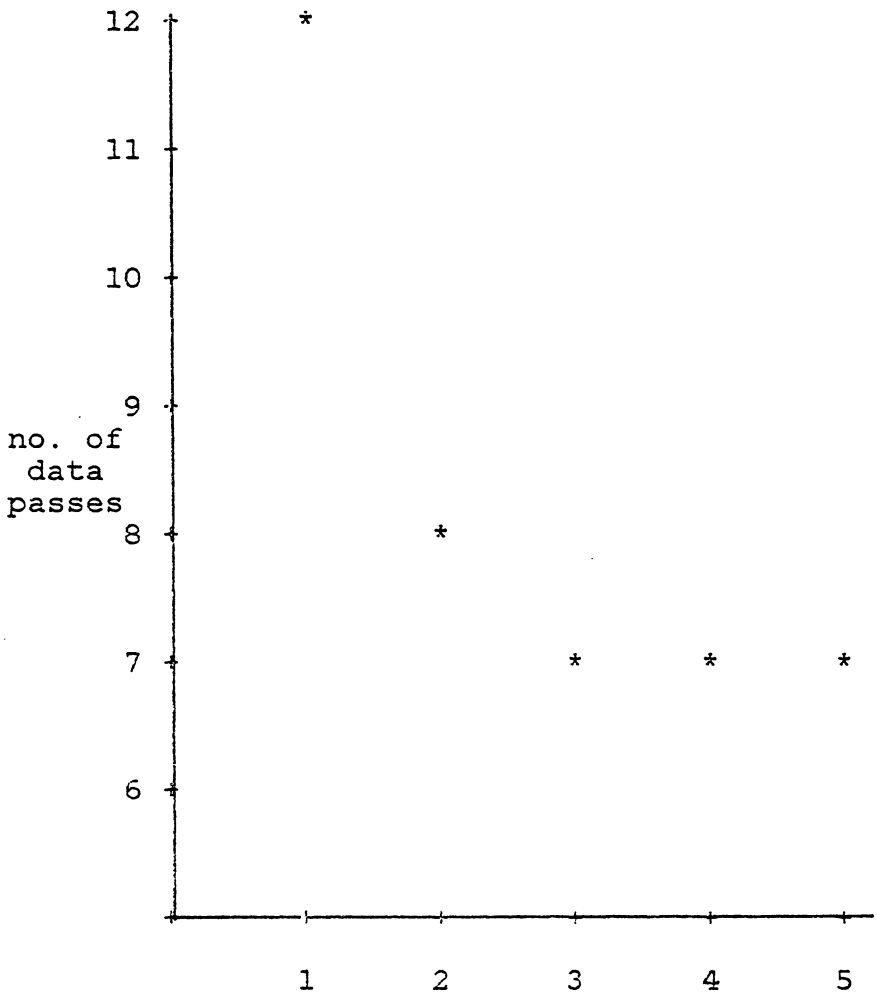

Figure 34: No. of data passes versus no. of iterations before passing data for a two subsystem 6 bus system

Hence it is advantageous to go in for math processor chips to do arithmetic operations.

Further reduction in execution time can be achieved by using third generation microprocessors like Intel 8086, Zilog z8000 or Motorola 68000.

7.4 RESULTS FOR OPTIMAL DECOMPOSITION

This section presents the results for the optimal decomposition technique discussed in chapter 4. The IEEE 14 bus system as shown in figure 35 is tested. The line data, the transformer data, the static capacitor data and the loads are presented in reference [36]. The system is divided into two subsystems for this analysis. In the first case, the subsystem I contains the buses 1, 2, 3, 4, 6, 7 and 8 and the subsystem II contains the buses 5, 9, 10, 11, 12, 13 and 14. In the second case, the subsystem I contains buses 1, 2, 3, 5, 6, 8 and 12 and the subsystem II contains the buses 4, 7, 9, 10, 11, 13 and 14. In the third case, the subsystem I contains buses 1, 2, 3, 4, 7, 8, 9 and the subsystem II contains buses 5, 6, 10, 11, 12, 13 and 14.

In the first case, subsystems are weakly coupled. In the third case, they are tightly coupled. In the second case,

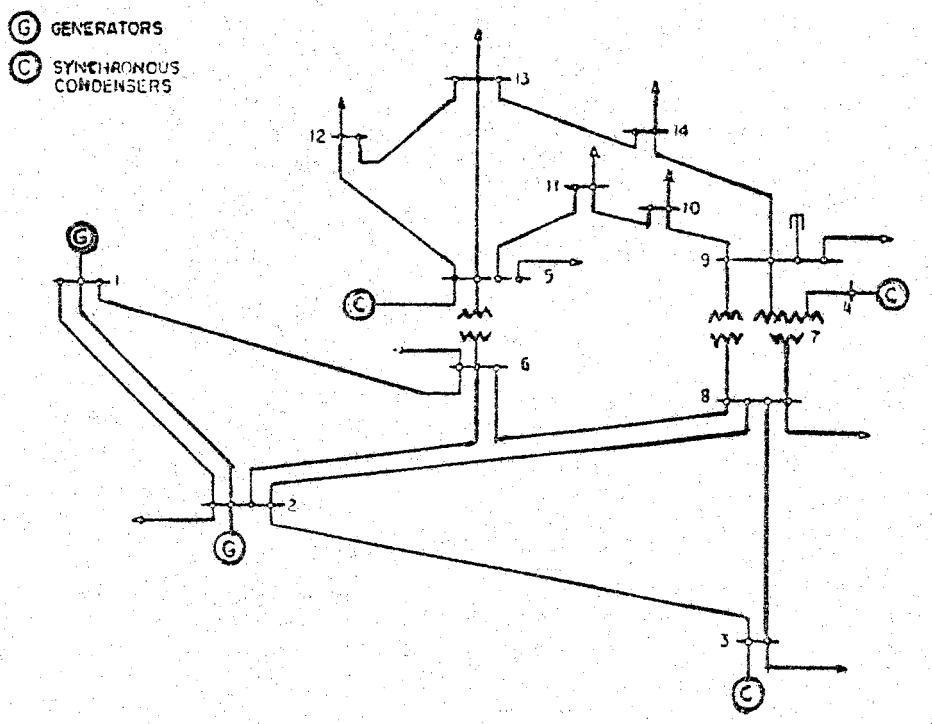

Figure 35: IEEE 14 bus test system diagram

the coupling is in between that of case I and case II. The tables 6, 7 and 8 show the results for these three cases. From the results it is clear that the number of data passes as well as the number of iterations will be less in the weakly coupled system. The time taken to obtain the final solution is proportional to the number of iterations in a system. Hence when the number of iterations decreases, the time taken to obtain the final result will also decrease.

In the physical decomposition technique, whenever there is a choice of lines to be selected for the cutset branches it is better to choose the ones which have the highest impedances.

TABLE 6

Case I result for the IEEE 14 bus system

No. of iterations before data pass	No. of data passes	total iterations
1	41	41
2	26	52
3	24	72
4	22	88
5	22	110
6	21	126

TABLE 7

Case II results for the IEEE 14 bus system

No. of iterations before data pass	No. of data passes	Total iterations
1	42	42
2	31	62
3	27	81
4	26	104
5	26	130
6	26	156

TABLE 8

Case III results for the IEEE 14 bus system

No. of iterations before data pass	No. of data passes	Total iterations
1	44	44
2	33	66
3	30	90
4	30	120
5	30	150
6	30	180

7.5 RESULTS FOR IMPROVED ECONOMIC DISPATCH

The improved economic dispatch algorithm is tested using the standard IEEE 14 bus system as shown in figure 35. In the first case, generators are located at buses 1 and 2. The cost curves for the generators are identical. The values used for the constants a, b and c mentioned in chapter 5 are 3.00, 1.5 and -0.05 respectively. For the total cost to be minimum, generator 1 must produce 130.95 MW and generator 2, 138.5 MW. The load flow for the corresponding results is given in table 9.

In case II, the same test system is used but the generators are located at the buses 1, 2 and 5. The incremental cost curves for the generators are given in table 10. In this study, different nonlinear programming techniques [25] are compared with the improved method. All the test cases were started with a flat voltage profile of one per unit.

Table 11 compares the results of the improved method with those of the nonlinear programming techniques [37]. It has already been stated that the Steepest-Descent is faster than either the Fletcher-Powell or Hessian-Matrix for this application. However, it is also well known that the Steepest-Descent method may oscillate in the proximity of the optimum point. Therefore it may be necessary to relax the error criterion or include some techniques to stop the program if it starts oscillating.

TABLE 9

Economic dispatch load flow result for the IEEE 14 bus system

Bus	Voltage		Real power	Reactive power
1	1.0600	0.0000	1.3095	0.1825
2	1.0442	-0.0397	1.1681	0.4562
3	0.9558	-0.1678	-0.9420	-0.1900
4	0.9830	-0.1948	0.0000	0.0000
5	0.9884	-0.2156	-0.1120	-0.0750
6	0.9892	-0.1128	-0.0760	-0.0160
7	0.9830	-0.1948	0.0000	0.0000
8	0.9797	-0.1349	-0.4780	0.0390
9	0.9731	-0.2247	-0.2950	-0.1660
10	0.9669	-0.2268	-0.0900	-0.0580
11	0.9735	-0.2230	-0.0350	-0.0180
12	0.9696	-0.2284	-0.0610	-0.0160
13	0.9645	-0.2287	-0.1350	-0.0580
14	0.9463	-0.2407	-0.1490	-0.0500

TABLE 10

Incremental cost curve data for generators

Bus no.	a	b	c
1	-2.45	0.01	0.0
2	-3.51	0.01	0.0
3	-3.89	0.01	0.0

TABLE 11

Comparison of nonlinear techniques with the improved method

	Steepest Descent	Fletcher Powell	Hessian Matrix	Improved Method
Total cost in $/hour	1136.44	1135.79	1135.40	1135.95
No. of load flow solutions	4	3	3	3

A direct comparison of the execution time is not available at present. It is clear from the improved method that no iteration processes are involved in calculating power generations. Thus, there will not be any oscillation or convergence problem during this step. The Lagrange multiplier can be evaluated in closed form. Therefore the only major time involved in this technique is that for load flow. However, all the methods have to perform the load flow. In this method the number of times that the load flow operation is performed is less than or equal to that of the other methods. Therefore this method will be faster.

It is also well known that the Hessian-Matrix and Fletcher-Powell methods are more accurate than the Steepest-Descent method but are slower. From the results it is clear that the improved method is accurate as well as fast.

Chapter VIII

CONCLUSIONS

In this work, different areas that are conducive to multiprocessing in power systems as well as different computer organizations are presented. This work also outlines the need for multiprocessing and the different openings available in power system analysis. A suitable computer architecture for power system problems is presented. To perform distributed processing, an efficient decomposition technique is developed using the physical arrangement of the system. An optimal decomposition method is also presented which minimizes the interactions between the subnetworks. The decomposition technique presented is very versatile and the system expansion is very easy to implement.

The decomposition technique developed is applied to the load flow analysis. To study the effectiveness of the decomposition algorithm, simulation is carried out using the IBM 370/158 computer. The simulation results show distributed processing is very attractive for power flow analysis.

To implement the algorithm in the microprocessors, multimicroprocessor architectures are studied. Distributed processing architectures using Intel 8080, 8085 and 8086 are presented. Fort/80 version of load flow algorithm is devel-

oped for Intel 8080 and 8085 microprocessors. The distributed microprocessor load flow algorithm is implemented in Intel 8085 microprocesors for a 6 bus system. Further reduction in execution time can be achieved by using third generation microprocessors and coprocessors to do arithemetic operations.

Different types of nonlinear programming techniques for economic dispatch were discussed. An improved method, which uses a direct method of calculating the Lagrange multiplier in a closed form with and without transmission losses has been presented. Some of the advantages of the method are:

1. The method utilizes a closed form expression for the calculation of the Lagrange multiplier.
2. The algorithm is fast and has no convergence problems.
3. No initial guess for the Lagrange multiplier is necessary.
4. The mathematical operations involved are simple.

From the above reasons it is seen that the algorithms presented in this work have potential for on-line implementations.

The distributed processing algorithms with and without transmission losses are developed. The distributed processing algorithm also uses the closed form solution of the La-

grange multiplier. Hence it has all the advantages of the improved method as well as that of a higher speed.

8.1 SUGGESTIONS FOR FURTHER WORK

In this work multiprocessing load flow algorithm is studied using the Gauss-Seidal technique. It will be interesting to study the effect of other load flow algorithms on distributed processing and their effect on the rate of convergence, speedup and efficiency.

In the area of the optimal load flow problem, economic dispatch with and without transmission losses are considered. Other possible areas for further study of optimal load flow are minimum pollution dispatch, minimum reactive compensation etc. New multiprocessing algorithms are needed for these applications.

The decomposition technique presented here can be applied to other areas where the system configuration is fixed by the physical arrangement.

The idea of multiprocessing can also be extended to other areas of power system analysis like stability, state estimation, system control, data acquisition, system security, energy management and load management. This initial investigation of the multiprocessor power flow analysis indicates that there are potential advantages to be gained by using

such a system. Further work should concentrate on implementing the algorithm in the third generation microprocessors to do on-line control.

BIBLIOGRAPHY

1 F. M. Brasch, J. E. Van Ness, S. C. Kang, Simulation of a multiprocessor network for power system problems. IEEE Trans. PAS, Vol 101, No. 2, pp. 295-301, Feb. 1982.

2 Diane M. Detig, Effects of special purpose hardware in scientific computation with emphasis on power system applications. IEEE Trans. PAS, Vol 101, No. 2, pp. 265-270, Feb. 1982.

3 C. W. Brice and R. K. Cavin, Multiprocessor static state estimation. IEEE Trans. PAS, Vol 101, No. 2, pp. 302-308, Feb. 1982.

4 R. Ramanathan, S. Rahman and L. L. Grigsby, Applications of multiprocessing to power system problems. 14th Southeastern Symposium on System theory, 1982.

5 D. E. Woods and R. D. Serafin, A multi-microcomputer based distributed front end communication subsystem for a power control center. IEEE Trans. PAS, Vol 101, No. 1, pp. 180-184, Jan. 1982.

6 E. C. Housos and O. Wing, Parallel optimization with application to power systems. IEEE Trans. PAS, Vol 101, No. 1, pp. 244-248, Jan. 1982.

7 R. Pritchard and C. Pottle, High speed power flows using attached scientific array processors. IEEE Trans. PAS, Vol 101, No. 1, pp.249-253, Jan. 1982.

8 J. P. Hulskamp, S. M. Chan and J. F. Fazio, Power flow outage studies using an array processor. IEEE Trans. PAS, Vol 101, pp. 254-261, Jan. 1982.

9 J. Fong and C. Pottle, Parallel processing of power system analysis problems via simple parallel microcomputer structures. IEEE Trans. PAS, Vol 97, No. 5, Sept./Oct. 1979.

10 Jang G. Lee, William G. Vogt and Marlin H. Mickle, Optimal decomposition of large-scale networks. IEEE Trans. Man and Cybernetics, Vol SMC-9, No. 7, July 1979.

11 Fernando L. Alvarado, Parallel solution of transient problems by trapezoidal integration. IEEE Trans. PAS, Vol 98, No. 3, pp. 1080-1090, May/June 1979.

12 Jang G. Lee, William G. Vogt and Marlin H. Mickle, Optimal network decomposition and load flow analyses for large scale power systems. IEEE PES Winter power meeting, 1978.

13 J. Fong and C. Pottle, Parallel processing of power system analysis problems via simple parallel microcomputer structures. IEEE Trans. PAS, Vol 97, No. 5, pp. 1834-1841, Sept./Oct. 1978.

14 W. L. Hatcher, F. M. Brasch and J. E. Van Ness, A feasibility study for the solution of transient stability problems by multiprocessor structures. IEEE Trans. PAS, Vol 96, No. 6, pp. 1789-1797, Nov./Dec. 1977.

15 Marlin H. Mickle, William G. Vogt and R. Gerald Colclaser, Parallel processing and optimal network decomposition applied to load flow analyses and related problems. EPRI EL-566-SR Special report, pp. 171-182, Oct. 1977.

16 Robert G. Andretich, Homer E. Brown, Harvey H. Happ and Conrad E. Person, The piecewise solution of the impedance matrix load flow. IEEE Trans. PAS, Vol 87, No. 10, Oct. 1968.

17 D. J. Kuck, The structure of computers and computations. Wiley and sons, 1978.

18 Jean Loup Baer Computer systems architecture. Computer science press, 1980.

19 Control Data Corportation, Star Fortran language version 2. Reference manual.

20 A. M. Erisman, Decomposition and sparsity with application to distributed computing. EPRI EL-566 SR Special Report, pp. 183-208, Oct. 1977.

21 F. L. Alvarado, D. K. Reitan and M. Bahari Kashani, Sparsity in diakoptic algorithms. IEEE Trans. PAS, Vol 96, No. 5, pp. 1450-1459, Sept./Oct. 1977.

22 W. D. Stevenson, Elements of power system analysis. John Wiley and sons, Inc., Third edition, 1975.

23 Olle I. Elgerd, Electric energy systems theory. McGraw-Hill Book Company, New York, 1971.

24 R. Ramanathan and L. L. Grigsby, Novel approach to economic dispatch. IEEE International Conference on Electric Energy, 1981.

25 R. Ramanathan and L. L. Grigsby, A comparison of an improved economic dispatch algorithm to nonlinear programming techniques. 13th Southeastern Symposium on System theory, 1981.

26 F. L. Alvarado, Penalty factors from Newton's method. IEEE Trans. PAS, Vol 97, No. 6, Nov./Dec. 1978.

27 H. H. Happ, Optimal power dispatch - a comprehensive survey. IEEE Trans. PAS, Vol 96, No. 3, May/June 1977.

28 A. M. Sasson, Some applications of optimization techniques to power system problems. IEEE Proceedings, Vol 62, No. 7, pp. 959-972, July 1974.

29 H. H. Happ, Optimal power dispatch. IEEE Trans. PAS, Vol 93, pp. 820-830, May/June 1974.

30 G. F. Reid and L. Hasdorff, Economic dispatch using quadratic programming. IEEE Trans. PAS, Vól 92, pp. 2015-2023, Nov. 1973.

31 H. W. Dommel and W. F. Tinney, Optimal power flow solutions. IEEE Trans. PAS, Vol 87, pp. 1866-1876, Oct. 1968.

32 D. R. Smith, Obsolecence evaluation of power control computer systems. IEEE Trans. Pas, Vol 101, No. 1, pp. 191-194, Jan. 1982.

33 Dennis Moralee, Microprocessor architectures: ten years of developments. Electronics and Power, pp. 214-221, March 1981.

34 John Palmer, Rafi Nave, Charles Wymore, Robert Koehler and Charles McMinn, Making mainframe mathematics. Electronics, pp. 114-121, May 8, 1980.

35 George Adams and Thomas Rolander, Design motivations for multiple processor microcomputer systems. Computer Design, pp. 81-89, March 1978.

36 IEEE standard test data. Courtesy of American Electric Power, New York, 1962.

37 Craig Alan Keeler, A study of optimal load flow using nonlinear programming. M.S, Thesis, Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, May 1977.

38 MCS-80/85 family user's manual. Intel corporation, Santa Clara, CA, Oct. 1979.

39 The 8086 family user's manual. Intel Corporation, Santa Clara, CA, Oct. 1979.

40 Fort/80 Fortran IV for the 8080 reference manual. Unified Technologies Incorporated, Ontario, Canada, 1976.

41 L. L. Grigsby, Load flow class notes. Unbublished

The vita has been removed from the scanned document

MULTIPROCESSING WITH MICROPROCESSORS FOR FOR POWER FLOW ANALYSES

by

Ramanathan Ramanathan

(ABSTRACT)

This work explores the need for multiprocessing in the power industry. A suitable computer architecture for power system problems is presented. An efficient decomposition technique is developed to solve the problem parallely.

Simulation of distributed processing using IBM 370 computer is considered. A microprocessor version of load flow program is developed and distributed load flow for on-line and off-line applications are studied.

A novel algorithm for economic dispatch with and without transmission losses is presented. The algorithm utilizes a closed form expression for the calculation of the Lagrange multiplier thereby avoding any iterative process in the calculation. Different nonlinear programming techniques are compared to the improved method. The algorithm presented is fast and appears to have good convergence properties. The improved algorithm is extended to distributed processing.

Characteristics of a distributed microcomputer system and a multiple processor system are discussed. Intel 8080, 8085 and 8086 versions of distributed and multiprocessing methods are presented to solve power system problems.