7th WSEAS Int. Conf. on MATHEMATICAL METHODS and COMPUTATIONAL TECHNIQUES IN ELECTRICAL ENGINEERING, Sofia, 27-29/10/05 (pp229-234)

Driving Point Impedance Computation Applying Parallel Processing Techniques

A. MEDINA, A. RAMOS-PAZ
Facultad de Ingeniería Eléctrica, División de Estudios de Posgrado
Universidad Michoacana de San Nicolás de Hidalgo
Edificio de la División del Posgrado de Ingeniería Eléctrica, Ciudad Universitaria, 58030
Morelia, Michoacán, MEXICO

Abstract: - This contribution deals with the application of parallel processing techniques based on Parallel Processing Machine (PVM) and Multithreading to the Driving Point Impedances (DPI) of Electric Power Systems. It is demonstrated that the application of parallel processing techniques dramatically reduces the intensive computation effort required for the determination of the power system response represented by driving point impedances.

Key-Words: - Driving Point Impedance, Parallel Processing, Parallel Virtual Machine (PVM), Multithreading.

1 Introduction

In general, an intensive computation effort is required to reproduce the system response for a practical range of frequencies for useful transient and harmonic analysis. The frequency step used is directly proportional to the accuracy obtained. For practical applications, this information is further processed so that the system response obtained with the driving point impedance is adequately reproduced at resonance frequencies by a frequency dependent equivalent, *e.g.* [1-3].

In this contribution two parallel processing platforms based on Multithreads [4] and PVM [5] are applied to the efficient computation of the system frequency response. It is shown that as the complexity of the system increases, the application of parallel processing significantly reduces the computational effort required by a conventional sequential process to obtain the network frequency response. This is achieved by increasing the process relative efficiency of the parallel processing.

2 Driving Point Impedance

The power network frequency response can be obtained by assembling, at any particular frequency, its respective admittance or impedance matrix from the individual system components. The transfer function between nodal currents and voltages appearing throughout the system busbars is represented, for any frequency *f*, by the matrix equation,

$$\tilde{I}_f = Y_f \tilde{V}_f \tag{1}$$

The inverse of the frequency admittance matrix Z_f is the frequency impedance matrix Z_f, where each diagonal element $Z_{j,j}$ is known as driving point impedance of node *j*.

The combination of inductive and capacitive elements as seen from a particular bus, can result in either series resonance or a parallel resonance. The result of a series resonance may be the presence of unexpected large amounts of harmonic currents flowing through certain network elements, whereas the result of parallel resonance may be the presence of excessive harmonic voltages across network elements [6][7].

3 Parallel Processing Techniques

Parallel processing can be defined as a form of information processing where two or more processors in combination with some form of inter-processor communication system, cooperate on the concurrent solution of a large problem [8]. The emergence of massive parallel processors and distributed computation have paved the way to the wide acceptance and application of parallel processing for the solution of problems of considerable magnitude in diverse fields.

3.1 Parallel Virtual Machine

PVM (Parallel Virtual Machine) [5] is a platform that allows an heterogeneous network computer set to work as a large computer of multiple processors. Thus, a low cost and powerful virtual computer of multiple supercomputers can be created. PVM has several important advantages, *e.g.*

- Easy to set up.
- Many virtual machines can co-exist with the same hardware.
- The development of programs is based on message passing libraries.
- Supports C and Fortran.
- The software is very portable.
- The code source is available for free.
- PVM enables users to exploit their existing computer hardware to solve much larger problems with minimal additional cost.

It supports operating platforms such as Windows or LINUX. In PVM the information is transferred by mean a zip-send-reception-unzip protocol. This protocol is based on message passing libraries.

3.2 Multithreading

Multithreading [4] is the application of lightweight subprocesses executed within a process sharing code and data segments, but with their own program counter, machine registers and stack. Global and static variables are common to all threads.

4 Parallel DPI calculation Scheme Proposed

Conventionally, the computation of Z_f is carried-out by means of a sequential process where $Z_{j,j}$ is obtained at each frequency. An accurate computation of $Z_{j,j}$ requires a small enough frequency step size to be used, so that parallel and series systems resonances are appropriately reproduced. However, the frequency step size is inversely proportional to the computational effort needed to obtain $Z_{j,j}$ over the entire frequency range of analysis. However, the computation of Z_f, for $f = f_0, f_1, f_2, \cdots f_n$ can take advantage from the fact that Z_{f0} is independent from Z_{f1}, which in turn is independent from Z_{f2}, etc. The above process independence makes ideal the application of parallel processing, since concurrent processes are used for the computation of Z_f over the frequency range of interest. Figure 1 illustrates the parallel scheme proposed for the computation of Z_f. Here, a main process element collects all the information related to the electric network such as topology, elements and involved electric variables and parameters. This information is stored in a linked list from a defined class, who contains the relevant information associated with each individual element.

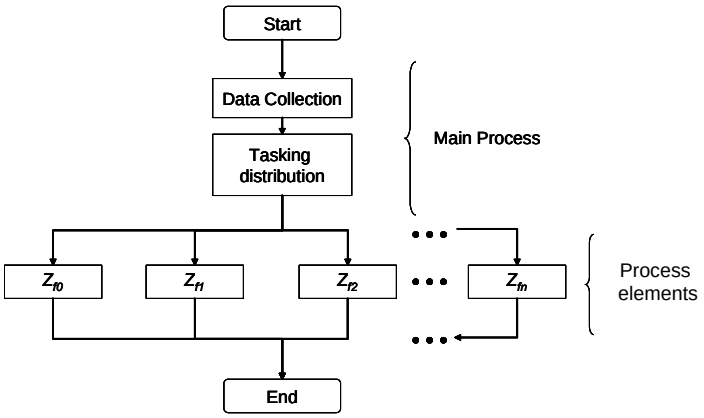

Figure 1. Proposed parallel scheme

Once the electric network information has been collected, the main process element determines the frequency range assigned to each thread for the computation of the system frequency behavior, *e.g.*

$$\text{Task by process element} = \frac{\dfrac{\text{No. of frequencies}}{\text{Frequency step}}}{\text{No. of process elements}} \quad (2)$$

If the frequency number is larger than the number of process elements in (2), then each process element computes more than one inverse of Y_f. However, if the number of threads is equal to the number of frequencies, then each element process calculates one inverse of Y_f. On the other hand, if the number of element process is larger than the number frequencies, then the additional threads are not used. Once the number of tasks to be carried-out by every process element is defined, the main element process sends a starting signal to the different threads involved in the process in the case of multithreads or executed every slave process in the case of PVM. Figure 2 illustrates the task assignment to process elements. Here, each process element builds-up the admittance matrix for a frequency f with the data stored in the linked list. Once the matrix has been assembled, the process element computes the inverse. The elements from the inverse are sent to the shared memory to be stored in a common variable in the case of Multithreads or are sent to the master

processor (in the case of PVM) to be stored in a common variable. This variable is protected against overwriting using the mutual exclusivity mechanism, which prevents simultaneous information access from multiple process elements.

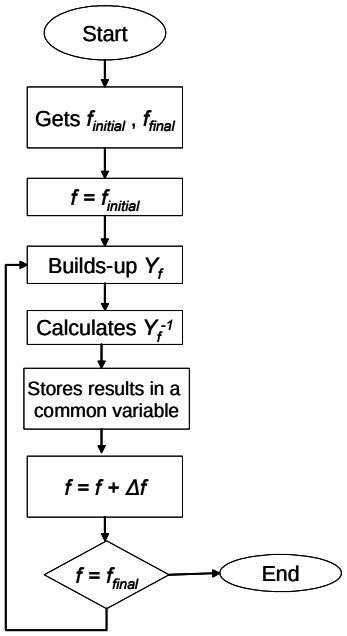

Figure 2. Task assignment to process elements

The process to calculate the inverse of Y_f is based on a LU bifactorization process, a forward and backward process and the use of sparse matrices. The inverse of Y_f is build up by columns by mean a forward – backward substitution process. For each column of Y_f a forward – backward substitution process is needed, where vector *b* contains zero elements, with 1 in the position associated with the column to be calculated. Figure 3 illustrates the inverse process.

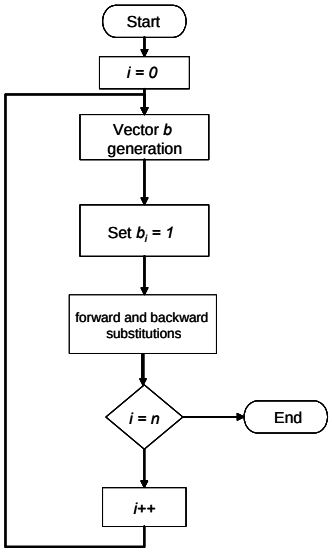

Figure 3. Inverse process of Y_f

Every single process element builds-up the Y_f matrix using the relevant information of the stored circuit in a linked list of structures, which contains information on the reception and sending nodes, the resistance, inductance and capacitance values and finally a number representing the element arrangement. This number is used to build-up the impedance value on function of the resistor, capacitor and inductance arrangement. Figure 4 illustrates the storage scheme used in this investigation. This scheme consists on an arrangement of pointers to a basic storage structure which contains three elements; an integer variable that represents the receiving node, a complex variable that stores the admittance value and a pointer to this structure. This pointer is used to link the different elements connected to a specific busbar. Based on the storage scheme each thread builds-up the Y_f at the different frequencies assigned.

The process of bifactorization is based on a LU procedure, where, a matrix is converted into the product of two matrices, a lower and an upper matrix respectively, *e.g.*

$$Y_f = LU \tag{3}$$

where the elements of the matrices L and U are calculated with (4) and (5) respectively.

$$l_{ij} = a_{ij} - \sum_{k<j} l_{ik} u_{kj} \qquad i \geq j \tag{4}$$

$$u_{ij} = \frac{\left(a_{ij} - \sum_{k<i} l_{ik} u_{kj} \right)}{l_{ii}} \qquad i < j \tag{5}$$

Once the matrix is bifactorizated, a forward – backward substitution of (6) and (7) is used to obtain, by columns, the elements of the matrix inverse of Y_f. During this process a zero-valued vector is used, except for a 1 in the position of the column to be obtained.

$$Y_i = b_i + \sum_{j=0}^{n} l_{ij} Y_j \quad \text{forward substitution} \tag{6}$$

$$X_i = l_{ii} Y_i + \sum_{j=i}^{n-1} r_{ij} X_j \quad \text{backward substitution} \tag{7}$$

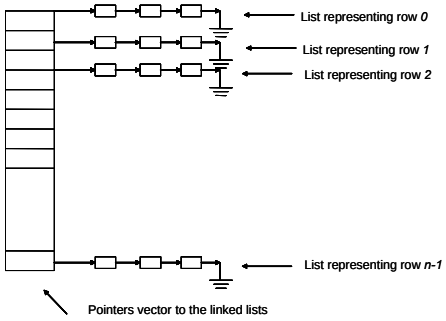

Figure 4. Storage scheme using linked lists

Before starting the bifactorization process an ordering scheme is used to generate the minimum number of non-zero elements.

5 Test Case

Figure 5 illustrates the test case to be analyzed. It contains five nodes, seven transmission lines, two generation units, represented by voltage injections of 1.0 *p.u*. The test system data are $r_1=r_2=r_3=r_4=r_5=r_6=10m\Omega$, $l_1=l_2=l_3=l_4=l_5=l_6=20mH$ for the transmission lines, $r_7=20m\Omega$, $l_7=2mH$, $C_1=200\mu F$, $l_8=l_9=0.69mH$, $C_2=C_3=300\mu F$, $l_{10}=11.3mH$ and $C_4=100\mu F$.

The programming code was developed in C++ with multithreading [4]. For the case with PVM, the PVM3 library [10] was used. Two processors 794.675 Mhz computer was used to executed the code associated with multithreads. The operative system used was Linux Ubuntu. The developed code reads a data file containing the system configuration and their parameters, the simulation data such as number of nodes, branches, harmonics, frequency step and the number of threads to be used in the simulation. The parallel virtual machine used in this investigation was build using 3 two processors 794.675 Mhz computers.

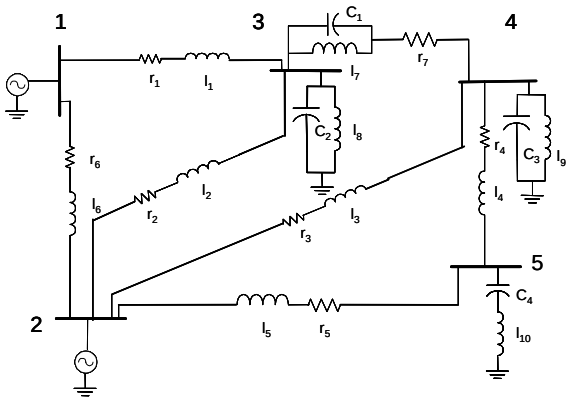

Figure 5. Test case

Appendix I illustrates the models used for representation of the synchronous machine and the transformer [6]. GNUPLOT [9] was used for the graphical representation of the driving point impedance.

For the case with PVM, the PVM3 library [5] was used whereas for the Multithreading case the PTHREAD library was applied [10]. The mutual exclusivity mechanism is applied to appropriately control the access from multiple threads [10][11].

Figure 6 illustrates the impedance system seen from node 2. It can be noticed that there are four points of parallel resonance, *e.g.* at 120, 180, 230 and 360 hz respectively.

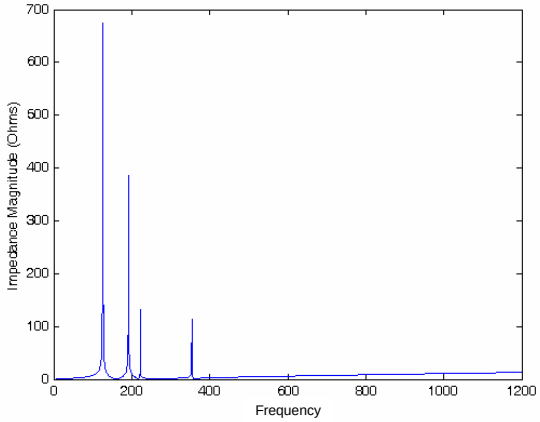

Figure 6. System frequency response, as seen from node 2

Figure 7 illustrates the system frequency response, represented by the DPI, given as the impedance magnitude *versus* frequency, as seen from node 4. Two parallel resonances take place at 120 and 350 Hz, respectively. A 0.1 Hz frequency step size was used with a 60 Hz base frequency.

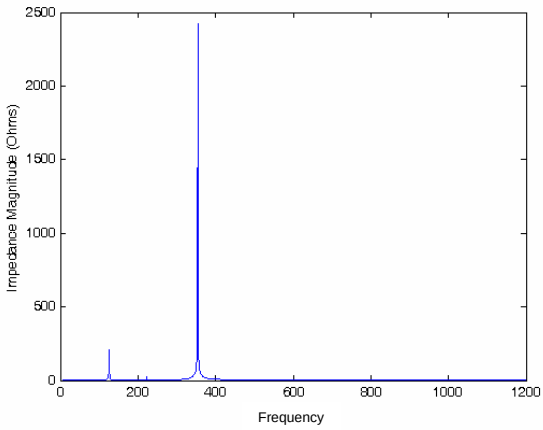

Figure 7. System frequency response, as seen from node 4

Figure 8 illustrates the effect of the frequency step in the driving point calculation. It can be observed that frequency steps of *1.0fund* (*fund* = fundamental) and *0.5fund* do not identify the parallel resonances associated with the analyzed electric power system, whereas the use of *0.1fund* and *0.01fund* identify all resonances points associated with the electric power systems analyzed.

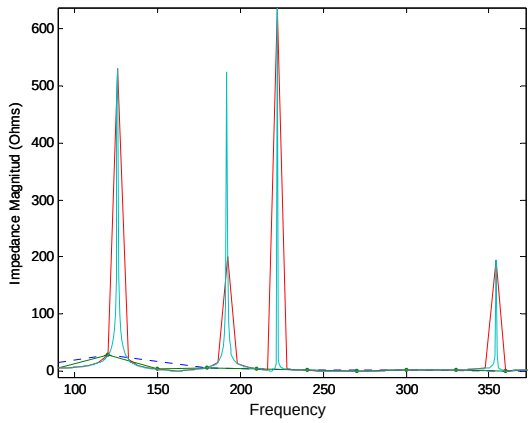

Figure 8. Effect of the frequency step in the DPI evaluation

The parallel processing relative efficiency is computed as [12],

$$E_{relative} = \frac{T_1}{T_p} \qquad (8)$$

where,

T_1 execution time with *1* process element.

T_p execution time with *p* process elements.

Tables 1-2 illustrate the relative efficiency achieved with the application of parallel processing based on multithreading to the computation of the frequency system response using two different frequency steps.

With $\Delta f = f_{fund}$ (=60 Hz) no improvement is obtained in the relative efficiency for the DPI computation, since it remains in 1.0. With $\Delta f = 0.1 f_{fund}$ the relative efficiency increases from 1.0 to 1.2727 with ten harmonics (times the fundamental frequency) and to 1.5238 with forty harmonics, see Table I. For $\Delta f = 0.01 f_{fund}$ the relative efficiency increases from 1.0 to 1.7631 with ten harmonics and to 1.9140 with forty harmonics, see Table II. The application of a third thread does not increase the relative efficiency, since for this investigation a two processors computer was used. The relative efficiency increases in direct proportion to the size of the problem to be solved and the number of threads and processors used, as seen from Tables 1-2.

Table 1. Relative Efficiency with *Δf=0.1fund*

Number of threads	Number of harmonics			
	10	**20**	**30**	**40**
1	1.0000	1.0000	1.0000	1.0000
2	1.2727	1.4285	1.5294	1.5238
3	1.2727	1.2500	1.3000	1.3333

Table 2. Relative Efficiency with *Δf=0.01fund*

Number of threads	Number of harmonics			
	10	**20**	**30**	**40**
1	1.0000	1.0000	1.0000	1.0000
2	1.7631	1.8529	1.8979	1.9140
3	1.7631	1.8260	1.8979	1.8992

Table 3 illustrates the relative efficiency obtained with the use of 1-3 slave processors. It can be noted that relative efficiency increases with the increase of slave processors and frequency step used. The maximum relative efficiency obtained is 2.97 for a *Δf=0.001* and with the use of 3 slave processors.

Table 3. Relative Efficiency obtained with PVM and 3 slave processors

Numper of slave processors	Relative Efficiency			
	Frequency Step			
	1.0	**0.1**	**0.01**	**0.001**
1	1.0000	1.0000	1.0000	1.0000
2	1.0556	1.1842	1.4474	1.9916
3	1.0000	1.5000	2.1154	2.9720

6 Conclusions

This contribution has introduced the application of parallel processing based on multithreading, to the fast calculation of driving point impedances in electric networks.

In particular, this investigation has demonstrated that the application of parallel processing techniques significantly increases the relative efficiency for the computation of the frequency dependent system response in the form of driving point impedances, as seen by any system busbar. The efficiency will increase in direct proportion with the system dimension, the size of problem and the number of process elements used.

Acknowledgements

The authors acknowledge the Universidad Michoacana de San Nicolás de Hidalgo (UMSNH) through the División de Estudios de Posgrado of the Facultad de Ingeniería Eléctrica the facilities granted to carry-out this investigation. A. Ramos-Paz acknowledges financial support received by the UMSNH and CONACYT to carry-out his Doctoral studies.

References:

[1] N.R. Watson, and J. Arrillaga, "Frequency-Dependent System Equivalents for Harmonic Studies and Transient Convertor Simulation", IEEE Trans.on Power Delivery, Vol. 3, No.3, pp. 1196-1203, July 1987.
[2] A. Medina, J. Arrillaga, and N.R. Watson,"Detivation of Multi-Harmonic Equivalent Models of
Power Networks", Fourth International Conference on Harmonics in Power Systems, Budapest, Hungary, Oct. 1990, pp. 290-297.
[3] A. Abur, and H. Singh, "Time Domain Modeling of External Systems for Electromagnetic Transients Programs", IEEE Trans. on Power Systems, Vol. 8, No. 2, pp. 671-679, May, 1993.
[4] Sun Microsystems, Inc. "Multihreaded Programming Guide", 1997.
[5] A. Geist, A. Beguelin, and J. Dongarra, J., "PVM: Parallel Virtual Machine", MIT Press 1994.
[6] J. Arrillaga. And B. C. Smith. Power System Harmonic Analysis. John Wiley and Sons.
[7] IEEE Recommended Practice for Industrial and Commercial Power Systems. IEEE.
[8] F. Alvarado, F.; R. Betancourt, G.T. Heydt "Parallel Processing in Power Systems Computation", IEEE Transactions on Power Systems, Vol. 7 No. 2. pp. 629-638. 1992.
[9] [online]. Avaliable: www.gnuplot.info
[10] Sun Microsystems, Inc. "*Multihreaded Programming Guide*".
[11] C.M. Pancake, "Multithreaded Languages for Scientific and Technical Computing', Proceedings of the IEEE, Vol.2 No. **81**, pp. 288-304. 1993.
[12] I. Foster, Designing and Building Parallel Programs Addison Wesley, 1994.

Appendix I

Power System Component Models

Synchronous Machine

$$Z_{generator} = r\sqrt{h} + jX_d^{"}h$$

where

$X_d^{"}$	generator subtransient reactance.
r	resistence
h	harmonic order

Transformer

$$Z_{generator} = r\sqrt{h} + jX_d^{"}h$$

where

X_t	transformer short circuit reactance.
r	resistence
h	harmonic order