Electric Power Systems Research 121 (2015) 295–301

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

Multi-frequency sweeping method for periodic steady-state computations on the graphics processor unit

Eric Morales-Aguilar[a,*], Abner Ramirez[a], Mahmoud Matar[b]

[a] Center for Research and Advanced Studies of Mexico (CINVESTAV), 1145 Av. del Bosque, Col. El Bajio, Zapopan, Jalisco 45019, Mexico
[b] Electrical Power and Machines Dept., Ain Shams University, Cairo, Egypt

ARTICLE INFO

Article history:
Received 29 August 2014
Received in revised form 4 November 2014
Accepted 7 November 2014
Available online 4 December 2014

Keywords:
Frequency domain analysis
Graphics processor unit
Large-scale systems
Parallel programming
Distribution network
Interharmonics

ABSTRACT

This paper presents the parallelization of the multi-frequency hybrid backward/forward sweeping (BFS) technique on a graphics processor unit (GPU). Primarily, the intrinsic layer structure of a radial network, typical topology of distribution systems, and its multi-frequency behavior are exploited for parallelization of the hybrid BFS method on the GPU. The less computational demanding tasks, e.g., error computation and simple vectorized operations, are assigned to the CPU. The network solution is performed in the Matlab® environment using compute unified device architecture (CUDA). The computational time required by the GPU/CPU BFS implementation is compared with a CPU-only program by solving four networks of different sizes. Validation of the multi-frequency BFS results is made through a CPU implementation of a Newton-type solution scheme. The significant reduction in the computational time of the parallelized GPU implementation of the hybrid BFS method combined with its ability to include a wide range of frequencies and to handle nonlinear components makes it suitable for real-time online applications.

1. Introduction

The high penetration of distributed energy resources (DERs) in distribution systems is driven by the strong environmental, economical, technical and social benefits that they provide. However, the integration of DER units brings new challenges to the operation, control, and protection of the distribution system [1]. Advanced power system analysis software tools in conjunction with high-performance computing platforms play a vital role in enabling the exploitation of the benefits of DER technologies while avoiding/minimizing their drawbacks.

Recently, parallel computing has demonstrated to be an appropriate tool for handling solution methods applicable to large-scale power systems [2]. A major goal of parallel computing is to speed up the solution of large networks. For example, the significant reduction in computational time is advantageous for online smart energy management systems as, for instance, it enables the real-time security assessment of the system based on evaluating the effects of hundreds of contingencies on the current operating conditions obtained from real-time telemetered data.

Moreover, the increasing number of frequency-producing electrical devices in distribution networks, e.g., arc furnaces, static converters, nonlinear reactors, and adjustable speed drives, has recently required large computational resources for the network analysis, especially when accounting for a wide range of frequencies [3–11]. Consequently, methods for computing the steady state of a network have evolved from considering fundamental power frequency [12–16], to processing a large number of frequencies [17–22].

Typically, a distribution network has a radial ("tree") topology [13,15,16]. Some distribution networks can additionally have few "transverse" lines; this type of networks has been called "weakly meshed" distribution networks [12,14]. Topologically, distribution networks can be schematically divided into in layers which represent the depth of the system's feeding loads; computationally, distribution networks are divided into layers for numbering purposes [13]. In this tenor, we propose a GPU implementation of the multi-frequency hybrid BFS technique (hereafter referred to as BFS only) which is mainly applicable to radial and/or weakly meshed networks [12]. The BFS method is able to: (i) include harmonics and interharmonics [17,18], since it is formulated in a modified harmonic domain (MHD) [22], (ii) avoid computation and

* Corresponding author. Tel.: +52 33 3777 3600x1021.
E-mail addresses: morales@gdl.cinvestav.mx (E. Morales-Aguilar), abner.ramirez@cts-design.com (A. Ramirez), mahmatar@ieee.org (M. Matar).

http://dx.doi.org/10.1016/j.epsr.2014.11.009
0378-7796/

inversion of Jacobian matrices and (iii) handle nonlinear components in a hybrid time/frequency domain solution scheme [17,18].

The calculation of electromagnetic transients in power systems, via the trapezoidal rule and based on an admittance matrix representation, is presented in [23]. In [24], the electrical network is expressed in single-instruction-multiple-data (SIMD) format, which is amenable for GPU implementation, in a transient stability simulation program. In [25], the power flow solution algorithms of Gauss–Seidel, *PQ*-decoupled and Newton–Raphson (NR) are implemented as sequential solvers on CPU and parallel solvers on GPU; they are numerically compared in terms of computational times. It is concluded in [25] that the NR parallel implementation on GPU speeds up more than 50 times its CPU sequential solver counterpart. It is worth noting that the NR implementation in [25] corresponds to fundamental frequency only. A similar study is presented in [26] for the NR and the Gauss–Jacobi techniques. In [15], a fast decoupled methodology is presented and numerically compared with the NR method, implicit Z_{bus}, and BFS power flow solution algorithms in terms of total number of floating point operations; however, the proposed techniques is limited to radial systems with one voltage-regulated bus adopted as the source. The methodologies in [15] are implemented as sequential solvers on CPU.

A recent implementation of the BFS method on the GPU is described in [27]. Theoretical speed-up relations are given for the parallelization processes of nodal current injections and backward/forward sweep. The implementation in [27] is, however, limited to linear networks and fundamental frequency.

In this paper, two network characteristics are exploited in the parallelization of the BFS on the GPU, i.e. (a) the decoupling of currents when sweeping layers, intrinsic to a radial network, in backward fashion and (b) the linearity of the network when sweeping layers in forward fashion for calculating node voltages [12]. Within the BFS method, a frequency-by-frequency network solution scheme is also exploited, parallelized, and implemented on the GPU.

Regarding the BFS implementation described in this paper, four case studies consisting of networks of different sizes are evaluated via the proposed GPU/CPU implementation of the BFS technique. All case studies are compared in terms of computational times with a CPU-only implementation of the BFS and validated with a CPU-only implementation of the Newton–Raphson (NR) technique.

2. Characteristics of solution techniques

Techniques for computation of the periodic steady-state of a network utilize iterative solution methods [17–20,22,23,25–28]. The most commonly employed solution methods are Newton-type methods and fixed-point algorithms, e.g., Gauss–Seidel. Generally speaking, Newton-type methods require fewer iterations as compared to fixed-point algorithms; however, appropriate initial values have to be carefully selected in the former whereas random initial values can be assigned to the latter. However, despite the fewer number of iterations required by Newton-type methods, fixed-point algorithms are computationally more efficient since they do not rely on computation and inversion of large matrices, e.g. Jacobian, especially when involving a large number of frequencies. These natural characteristics of Newton-type solution schemes seem to be unattractive for parallelization.

The BFS technique belongs to the fixed-point algorithms category. Its application to a radial network solution involves sweeping back and forth each of the paths (limited by the farthest elements and the feeder) of the radial network structure. Since the network sweeping involves simple algebraic relations between nodal voltages and branch currents, large matrix operations are avoided. Furthermore, due to the separate handling of linear and nonlinear network components, the BFS method can readily be structured in a frequency-by-frequency solution scheme. The aforementioned characteristics of the BFS method allow its straightforward implementation on the GPU, as presented in the next sections of the paper.

Compared to existent research, our implementation of the BFS method on the GPU considers a wide range of frequencies (harmonics/interharmonics) and nonlinear loads. The linear network is formulated in the MHD to account for any number of frequencies, as described in Section 3. The nonlinear components are resolved in the time domain and interfaced to the MHD via discrete Fourier transform (DFT) operations.

3. Basics of the modified harmonic domain

The BFS technique has recently been extended to include interharmonic frequencies [17,18]. The extension, i.e., the MHD, is based on the DFT and its inverse transform (IDFT), expressed as [29]:

$$x_n = \frac{1}{\Delta t}\left[\frac{1}{N}\sum_{k=0}^{N-1} X_k \sigma_k e^{jk\Delta\omega n\Delta t}\right], \quad n = 0, 1, \ldots, N-1, \tag{1a}$$

$$X_k = \Delta t\left[\sum_{n=0}^{N-1} x_n e^{-jk\Delta\omega n\Delta t}\right], \quad k = 0, 1, \ldots, N-1, \tag{1b}$$

where N is the number of samples and σ is a data window for diminishing the Gibbs phenomenon due to truncation. The discretization variables used in (1) are: $\omega = k\Delta\omega$ and $t = n\Delta t$ where $\Delta\omega = 2\pi/T$ (T corresponds to the observation period of time).

The MHD permits the conversion of ordinary differential equations (ODEs) to a set of algebraic equations arranged in a vector/matrix form [22], similar to the harmonic domain (HD) technique [20]. For instance, and without loss of generality, consider the scalar ODE given in (2) where we assume that coefficients a and b are constant.

$$\dot{x}(t) = ax(t) + bu(t). \tag{2}$$

Each variable in (2) is expressed via its DFT representation, as in (1a), yielding

$$\frac{1}{\Delta t}\left[\frac{1}{N}\sum_{k=0}^{N-1} jk\Delta\omega X_k e^{jk\Delta\omega n\Delta t}\right] = a\frac{1}{\Delta t}\left[\frac{1}{N}\sum_{k=0}^{N-1} X_k e^{jk\Delta\omega n\Delta t}\right] + b\frac{1}{\Delta t}\left[\frac{1}{N}\sum_{k=0}^{N-1} V_k e^{jk\Delta\omega n\Delta t}\right]. \tag{3a}$$

Note that the left-hand side in (3a) corresponds to the derivative of variable x with respect to $n\Delta t$. Arrangement of the DFT coefficients in (3a) using a vector/matrix structure, and dropping out the exponential terms, results in

$$DX = aX + bU, \tag{3b}$$

where the structure of X (and U) is:

$$X = \begin{bmatrix} X_0 & X_1 & \cdots & X_{N-1} \end{bmatrix}^{Tr}, \tag{4}$$

where Tr denotes transpose. The derivative terms in (3b) result in the diagonal matrix of differentiation D [20], given by

$$D = \mathrm{diag}\{0, \quad j\Delta\omega, \quad j2\Delta\omega, \quad \cdots, \quad j(N-1)\Delta\omega\}. \tag{5}$$

It is mentioned that the data window, σ, is used in this paper only when transforming an MHD variable into its time-domain counterpart. Further details of the MHD can be found in [22].