Parallel Algorithm for Transient Stability Simulation Based on Sherman-Morrison Formula

Hongyi Chen*, Man Xiao, Mengtao Hu, Ying Han, Tao Wang, Xiaodong Chen
State Grid Zhijiang Power Supply Company, Zhijiang 443200, Hubei Province, China

*Corresponding author e-mail: 651547599@qq.com

Abstract. Parallel computing is an effective approach to transient stability real-time simulation of large-scale power system. In this paper, a new parallel algorithm for power system transient stability simulation is proposed by combining Gauss method and the Sherman-Morrison formula. The algorithm adopts the Gauss method to convert the differential-algebraic equations into a set of nonlinear algebraic equations by multi-stages discretization, while the algebraic system is solved using rigorous Newton method. On this basis, the whole Jacobian matrix involved in Newton method is split to a block diagonal matrix and a block constant coefficients matrix according to s time points, and then based on the block diagonal matrix, the computing tasks at s time points are fully decoupled through the extended Sherman-Morrison matrix inverse formula. The proposed algorithm preserves the good convergence of rigorous Newton method and meanwhile has a high degree of parallelism both in time and in space.

1. Introduction
Transient stability simulation is important for planning, design, operation, control, and post-disturbances analysis in power systems [1-4]. It involves the solution of a large number of differential and algebra equations (DAEs), which is one of the most computation-intensive tasks in power system analysis. The parallel computing technique provides a new way to accelerate the power system transient stability simulation and much research on parallel algorithms of power system transient stability simulation have been conducted to improve the computational efficiency to meet with the requirement of on-line security analysis [5].

Exploring the transient stability literature reveals that the efforts on the parallel simulation of power systems have developed in two directions: numerical integration method [5-10] and hardware architecture. From the numerical integration method viewpoint there has been extensive research done on solving the ordinary differential equations with accuracy and efficiency, such as the explicit Runge-Kutta method [5], the implicit trapezoidal integration method [6, 7], and so on. However, the implicit trapezoidal integration method is only of order 2, and cannot use a larger step size, the explicit Runge-Kutta method is not A-stable, which can't get credible simulation results. Multi-stage implicit Runge-Kutta (RK) methods [11-13] is not only A-stable but also high-order numerical integration methods. Among implicit Runge-Kutta method families, s-stage Gauss method [8] is A-stable together with highest order of 2s. So making use of s-stage 2s-order Gauss method for time parallel computation of transient stability is a natural and effective idea.
In this paper, we present a novel parallel algorithm based on Gauss method for simulating transient stability. The basic idea of the parallel algorithm is, turning differential-algebraic equations into non-linear equations by multi-stages discretisation, and then solving non-linear equations by Newton method. The whole Jacobian matrix involved in Newton method is split to a block diagonal matrix and a block constant coefficients matrix according to s time points using block matrix characteristics, and then the computing tasks at s time points are fully decoupled through the extended Sherman-Morrison matrix inverse formula [14, 15].

The paper is organized as follows. In Section 2, the parallel computation formula of transient stability using Gauss method is deduced in detail. One test case with detailed dynamic models are used to evaluate the proposed parallel algorithm in Section 3. Finally, Section 4 gives the conclusion of the work.

2. Parallel algorithm of transient stability simulation based on Gauss method and Sherman-Morrison formula

2.1. Basic framework for transient stability computation based on Gauss method

The power system transient stability model is described by a group of first-order differential equations

\[
\dot{x} = f(x, V)
\]  

And a group of algebraic equations

\[
0 = V - I(x, V)
\]

Where \( x \) is the state variable vector of the dynamic units, and \( V \) the bus voltage vector. (1) includes the differential equations for all dynamic units, such as the generation unit. (2) includes the network equations for the whole power grid.

Using \( s \)-stage Gauss method to discretize (1) leads to

\[
x_{n+1} = x_n + h \sum_{j=1}^{s} b_j f(\bar{x}_j, \bar{V}_j)
\]  

And

\[
\bar{x}_i = x_n + h \sum_{j=1}^{s} a_{ij} f(\bar{x}_j, \bar{V}_j), i \in (1, s)
\]

Define

\[
\bar{z}_i = \bar{x}_i - x_n, \quad A^{-1} = q = (q_{ij}), i, j \in (1, s), \quad d^T = [d_1, d_2, \ldots, d_s] = b^T q
\]

Making use of (5), (3) and (4) reduce to

\[
\sum_{j=1}^{s} q_{ij} \bar{z}_j - hf(\bar{z}_i + x_n, \bar{V}_i) = 0, \quad i \in (1, s)
\]

\[
x_{n+1} = x_n + h \sum_{j=1}^{s} d_j \bar{z}_j
\]
Combining (6) and (2) results in
\[
\begin{align*}
\sum_{j=1}^{s} q_{ij} \tilde{z}_{j} - h f (\tilde{z}_{i} + x_{n}, \tilde{V}_{i}) &= 0, i \in (1,s) \\
g (\tilde{z}_{i} + x_{n}, \tilde{V}_{i}) &= 0
\end{align*}
\] (8)

Define
\[
\begin{align*}
\tau_{i} &= \frac{\partial f (\tilde{z}_{i} + x_{n}, \tilde{V}_{i})}{\partial \tilde{z}_{i}} \\
\mu_{i} &= \frac{\partial f (\tilde{z}_{i} + x_{n}, \tilde{V}_{i})}{\partial \tilde{V}_{i}}, i \in (1,s) \\
\sigma_{i} &= \frac{\partial g (\tilde{z}_{i} + x_{n}, \tilde{V}_{i})}{\partial \tilde{z}_{i}} \\
\eta_{i} &= \frac{\partial g (\tilde{z}_{i} + x_{n}, \tilde{V}_{i})}{\partial \tilde{V}_{i}}
\end{align*}
\] (9)

\[
\begin{align*}
\Delta f_{i} &= h f (\tilde{z}_{i} + x_{n}, \tilde{V}_{i}) - \sum_{j=1}^{s} q_{ij} \tilde{z}_{j} , i \in (1,s) \\
\Delta g_{i} &= -g (\tilde{z}_{i} + x_{n}, \tilde{V}_{i}) \\
\Delta H_{i} &= \begin{bmatrix} \Delta f_{i} \\ \Delta g_{i} \end{bmatrix}, \Delta w_{i} = \begin{bmatrix} \Delta \tilde{z}_{i} \\ \Delta \tilde{V}_{i} \end{bmatrix}, i \in (1,s)
\end{align*}
\] (10)

Applying the Newton method to (8) yields
\[
J \Delta X = \Delta F
\] (12)

Where
\[
\begin{align*}
\Delta X &= \begin{bmatrix} \Delta w_{1} & \cdots & \Delta w_{s} \end{bmatrix}^{T} \\
\Delta F &= \begin{bmatrix} \Delta H_{1} & \cdots & \Delta H_{s} \end{bmatrix}^{T}
\end{align*}
\] (13)

\( J \) is the whole Jacobian matrix, and its concrete expression form is as follows
\[
J = J_{d} + \Delta J
\] (14)

\[
J_{d} = \text{diag}(J_{i}), i \in (1,s)
\] (15)

\[
J_{i} = \begin{bmatrix} q_{ii} I_{m} - h \tau_{i} & -h \mu_{i} \\ \sigma_{i} & \eta_{i} \end{bmatrix}, i \in (1,s)
\] (16)
\( \Delta J = [\Delta \tilde{J}_j], i, j \in (1, s) \) \hfill (17)

\( \Delta \tilde{J}_j = 0, \Delta \tilde{J}_j = \begin{bmatrix} q_i I_m & 0 \\ 0 & 0 \end{bmatrix}, i \neq j \) \hfill (18)

### 2.2. Parallel algorithm based on the Sherman-Morrison Formula

The dimension of the block Jacobian matrix \( J \) is set to \( \nu \), then the dimension of the whole Jacobian matrix \( J \) is \( \nu \times \nu \). The vector \( u_s, v_s \in \mathbb{R}^{(s \times \nu)} \) of dimension \( s \times \nu \) is as follows:

\[
\begin{align*}
U &= \begin{bmatrix} u_1, u_2, \ldots, u_{s\times\nu} \end{bmatrix} \\
Y^T &= \begin{bmatrix} v_1, v_2, \ldots, v_{s\times\nu} \end{bmatrix}
\end{align*}
\]

\[
U = \begin{bmatrix}
I_m & 0 \\
0 & I_m & 0 \\
& & \ddots & \ddots & \ddots \\
& & 0 & I_m & 0
\end{bmatrix}
\]

\[
Y^T = \begin{bmatrix}
0 & \tilde{y}_{1,2} & \cdots & \tilde{y}_{1,s} \\
\tilde{y}_{2,1} & 0 & \cdots & \tilde{y}_{2,s} \\
\vdots & \ddots & \ddots & \ddots \\
\tilde{y}_{s,1} & \cdots & \tilde{y}_{s,s-1} & 0
\end{bmatrix}
\]

\[
\tilde{y}_{ij} = (q_i I_m, 0), i, j \in (1, s)
\]

\[
\sum_{k=1}^{s\times\nu} u_k v_k^T = UY
\]

With (19)-(24), it produces

\[
J = J_d + \sum_{k=1}^{s\times\nu} u_k v_k^T
\]

On the basis of the above, the extended Sherman-Morrison matrix inversion formula [15] is applied to the Eq. (25):

\[
\Delta X = \Delta \tilde{X} - J_d^{-1}P \tilde{R}^{-1} \tilde{Q}^T \Delta \tilde{X}
\]

Where

\[
\Delta \tilde{X} = J_d^{-1} \Delta F = [\Delta \tilde{w}_1, \ldots, \Delta \tilde{w}_s]^T
\]

\[
\Delta \tilde{X} = J_d^{-1} \Delta F = [\Delta \tilde{w}_1, \ldots, \Delta \tilde{w}_s]^T
\]
\[ \Delta \tilde{w}_k = J_k^{-1} \Delta H_k, \quad k \in (1, s) \]  
(28)

\[ \tilde{P} = [\tilde{p}_i], \quad \tilde{Q} = [\tilde{q}_i], \quad k \in (1, s \times m) \]  
(29)

\[ \bar{R} = \text{diag}(\bar{r}_i), \quad k \in (1, s \times m) \]  
(30)

\[
\begin{aligned}
\tilde{p}_i &= u_i - \sum_{i=1}^{s} \tilde{q}_i^T (J_i u_i) \tilde{p}_i, \\
\tilde{q}_i &= v_i - \sum_{i=1}^{s} \frac{(v_i^T J_i u_i)}{r_i} \tilde{q}_i, \\
\bar{r}_i &= 1 + \tilde{q}_i^T (J_i u_i), \quad k \in (1, s \times m)
\end{aligned}
\]  
(31)

Summarizes the derived algorithm converted the sparse triangular decomposition of whole Jacobian matrix \( J \) into the sparse triangular decomposition of \( s \) sub-matrix \( J_i, i \in (1, s) \), which has a good time parallelism and space parallelism, moreover, maintained a good convergence of rigorous Newton method in the overall solving process.

3. Tests results

3.1. Basic performance for proposed method

In this section, the IEEE145-bus system was used to demonstrate our proposed parallel algorithm, a three-phase fault happened at 0s and the branch tripped at 0.1s, and the simulation time was 3.0s. The stop criterion for the Newton iteration was set as 0.0001.

Fig. 1 is the number of Newton iterations in the process of step-by-step integration after clearing the fault \( (t \geq 0.1) \). It can be seen from Fig. 1, the number of Newton iterations is smooth, and it satisfies the convergence precision with the 2~3 iterations. Obviously, the proposed algorithm has good convergence.

Fig. 2 gives the comparison of the relative power angle error between Gauss method and the implicit trapezoidal integration method. As can be seen from Fig. 2, the s-stage Gauss method still has higher computational accuracy when the step size is \( s \) time of the implicit trapezoidal integration method.

![Figure 1](image_url)

**Figure 1.** Newton iterations at each integration step.
3.2. Parallel performance using OpenMP

In this paper, both the traditional CPU sequential computing model (model-1) and the OpenMP-based multi-core parallel computing model (model-2) were used for test. The computation time of the two computing models is given in Table 2. The speedup $\gamma$ is defined as the ratio of the computation time in model-1 compared with model-2. The configuration is the AMD FX-6200 six-core computer for experiment.

Table 1. The parallel performance of the proposed method.

| System   | $s/h$ | Computation time $/ms$ | $\gamma$ |
|----------|-------|------------------------|----------|
|          |       | model-1                | model-2  |          |
| IEEE-145 | 3/0.2s| 419                    | 184      | 2.28     |
|          | 5/0.3s| 387                    | 102      | 3.79     |

As can be seen from Table 1, when 3 ($s=3$) and 5 ($s=5$) parallel threads were used, the proposed algorithm can achieve a better speedup for the two different power systems.

In summary, the proposed algorithm can achieve better parallel performance. This is mainly due to the two points: (1) the algorithm decouples the computing task according to the $s$ time points, which was evenly assigned to each thread or the core, (2) the threads of the computing task are relatively concentrated in integration step, thereby the switching time between sequence execution and parallel execution can larger reduce.

4. Conclusion

In this paper, a new kind of parallel algorithm for transient stability computation is derived by combining $s$-stage Gauss method of order $2s$ and the extended Sherman-Morrison formula. The following conclusions can be drawn:

1) The parallel algorithm proposed in this paper is based on the rigorous Newton method, the convergence of the algorithm is good in the global solution for different scale systems.

2) The proposed algorithm is based on Gauss methods to determine the number of concurrent threads. From the view of the fact that Gauss method is a series of integration methods, so the proposed algorithm in this paper is not only possessed of stronger applicability and generality.

3) The proposed algorithm is more suitable for shared memory multi-core parallel computing model. The proposed parallel algorithm can achieve a good speedup for the large-scale power system transient stability calculation, which can satisfy the requirements of large-scale power systems transient stability analysis.

References

[1] Sauer P W, Pai M A. Power system dynamics and stability. Prentice Hall, 1998.
[2] IEEE Std 1110-2002. IEEE guide for synchronous generator modeling practices and applications in power system stability analyses. IEEE Power Eng Soc 2003; 1-81.

[3] Review of on-line dynamic security assessment tools and techniques, CIGRE Working group C4.601, Tech. Brochure 325, 2007.

[4] Abhyankar S, Flueck A J. Real-time power system dynamics simulation using a parallel block-Jacobi preconditioned Newton-GMRES scheme. High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion. IEEE, 2012: 299-305.

[5] Qin Z, Hou Y. A GPU-based transient stability simulation using Runge-Kutta integration algorithm. International Journal of Smart Grid & Clean Energy, 2013, 2 (1): 32-39.

[6] Wang F Z. Parallel-in-time relaxed Newton method for transient stability analysis. IEEE Proceedings-Generation, Transmission and Distribution, 1998, 145 (2): 155-159.

[7] Iavernaro F, Scala M L, Mazzia F. Boundary values methods for time-domain simulation of power system dynamic behavior. IEEE Transactions on Circuits & Systems I Fundamental Theory & Applications, 1998, 45 (1): 50-63.

[8] Wen B, Guo W, Hu J, et al. Parallel computation of transient stability using symplectic Gauss method and GPU. Third International Conference on Mechanic Automation and Control Engineering. IEEE Computer Society, 2012: 1974-1978.

[9] Pessanha José E O, Paz Alex A, Prada Ricardo, Poma Carlos P. Making use of BDF-GMRES methods for solving short and long-term dynamics in power systems. International Journal of Electrical Power & Energy Systems, 2013, 45 (1): 293-302.

[10] Fu C, Mccalley J D, Tong J. A Numerical Solver Design for Extended-Term Time-Domain Simulation. IEEE Transactions on Power Systems, 2013, 28 (4): 4926-4935.

[11] Hairer E, Norsett S P, Wanner G. Solving ordinary differential equations I: nonstiff problems. New York: Springer-Verlag, 1987.

[12] Hairer E, Wanner G. Solving ordinary differential equations II: stiff and differential-algebraic problems. New York: Springer-Verlag, 1996.

[13] Butcher J C. Numerical methods for ordinary differential equations. Wiley, New York, 2008.

[14] Bru R, Cerdán J, Marín J, et al. Preconditioning sparse nonsymmetric linear systems with the Sherman-Morrison formula. Siam Journal on Scientific Computing, 2003, 25 (2): 701-715.

[15] Maponi P. The solution of linear systems by using the Sherman-Morrison formula. Linear Algebra & Its Applications, 2007, 420 (420): 276-294.