GPU-Based Real-time N-1 AC Power Flow Algorithm With Preconditioned Iterative Method

Kunjie Tang, Shufeng Dong
College of Electrical Engineering
Zhejiang University
Hangzhou, China
tangkunjie1994@163.com

Bingquan Zhu, Qiulong Ni
Power Dispatch and Control Center
Zhejiang Electric Power Corporation
Hangzhou, China

Yonghua Song
College of Electrical Engineering
Zhejiang University
Hangzhou, China

***Abstract*—With the expansion of scale of power system, in order to satisfy the real-time and accuracy requirements of N-1 security, a GPU based real-time N-1 AC power flow algorithm with preconditioned iterative method was proposed. This algorithm concatenated independent power flow problems into one. The concatenated Jacobi matrix was formed by parallel processing, and the linear equations were solved by the direct method or the iterative method according to the scale of equations. The iterative method was designed and parallelly processed based on preconditioning method ILU(0) and Krylov theory. The case analysis shows that, the proposed algorithm has high efficiency, high accuracy and needs small memory footprint, which could be applied to engineering practice.**

***Index Terms*—N-1 security; GPU-CPU computing framework; parallel processing; concatenation algorithm; iterative method**

I. INTRODUCTION

With expansion of scale of power system, the static security problem is becoming more and more severe. Contingency simulation, especially N-1 security is an important means for stability analysis. Since N-1 security is used for real-time security analysis and decision-making support, N-1 power flow calculation has a high demand for computing accuracy and speed, especially in large scale system.

DC power flow method and sensitivity analysis method are widely used in N-1 security at present because of their high calculating speed [1-2]. DC power flow method simplifies the power flow model, and has no convergence problem. Sensitivity analysis method is based on the normal connection and normal operation mode, and there is no need to solve each power flow problem iteratively. Although recent researches have been studies on improving accuracy of these two methods, there are still some defects and limitations in practical applications because of the approximation of the methods themselves, especially in some circumstances requiring high computational accuracy [3].

Therefore, in order to ensure the accuracy of N-1 power flow calculation, a high-speed AC power flow method is an urgent need. Traditional methods such as Newton-Raphson method (hereinafter referred to as N-R method), via repeated iterating, forming Jacobi matrices and solving modified equations, suffers from large amount and slow speed of calculation, especially in large scale systems.

In recent years, with the rapid development of GPU, GPU-CPU computing framework capable of powerful parallel computing ability has been applied to scientific researches and engineering applications of many disciplines. In the AC power flow method, solving sparse linear equations occupy most of computation time, some researchers using GPU to accelerate solving process has achieved initial achievements [4-5]. However, these achievements mainly focus the parallel processing of direct methods and some traditional iterative methods, e.g. Gauss-Seidel method, having not deeply studied on the new iterative methods and preprocessing techniques. The residual time of solving equations is mainly for Jacobi matrix formation, and parallel acceleration of Jacobi matrix formation needs further study.

In consideration of above problems, N-R method is used to improve the calculation accuracy. On the basis of GPU-CPU computing framework, a concatenation algorithm for N-1 AC power flow (ACPF) is proposed to satisfy the real-time and accuracy requirement of N-1 security. This algorithm concatenates all independent power flow problems into one, which means that N-R method will be used only once, improving computational efficiency. Meanwhile, one iterative method, based on Krylov theory and ILU(0) preconditioning, plus parallel processing by GPU is proposed to solve concatenated large-scale equations. In addition, the formation of concatenated Jacobi matrix is also processed parallelly executing on GPU. The case analysis shows that, the proposed algorithm has high efficiency, high accuracy and needs small memory space, which could be applied to engineering practice.

II. CONCATENATION ALGORITHM FOR N-1 SECURITY

N-R method is a logical and sequential algorithm, while GPU is only suitable for situations that computation is intensive but with simple logic, which indicates that using GPU to parallelly solve every independent power flow problem in the N-1 security with N-R method is not feasible. In addition, memory of GPU is limited, and GPU cannot communicate directly with CPU. Executing the complete N-R method on GPU needs lots of space, which is not realistic. Based on this, a concatenation algorithm is proposed.

978-1-5386-7703-2/18/$31.00 ©2018 IEEE

N-1 ACPF needs to respectively iterate to solve the independent problems in the case of each transmission line and transformer branch being removed. When scale of nodes and branches is large, the number of independent problems is large too. The concatenation algorithm is proposed to reduce the computational time by reducing the times of using N-R method.

Assumed that a system, via connectivity check, has n transmission lines or transformer branches need to do contingency analysis. If N-R method is used for the case of each branch removed, such equations need to be solved for branch k removed:

$$\begin{bmatrix} \Delta \boldsymbol{P}_{ki} \\ \Delta \boldsymbol{Q}_{ki} \end{bmatrix} = \boldsymbol{J}_{ki} \boldsymbol{X}_{ki} (1 \le i \le p_k) \tag{1}$$

$\Delta \boldsymbol{P}_{ki}$, $\Delta \boldsymbol{Q}_{ki}$, $\boldsymbol{J}_{ki}$ and $\boldsymbol{X}_{ki}$ respectively represent node active power injection correction vector, node reactive power injection correction vector, Jacobi matrix and solution of modified equations in the i-th iteration for the case of branch k removed. If the problem converges and iteration times are below than the preset maximum iteration times, p_k represents iteration times for the case of branch k removed, or it is equal to the preset maximum iteration times.

The accuracy of Newton-Raphson is gradually increasing with the increment of iteration times. If the iteration times for the case of any branch removed is set as P, and:

$$P = \max\{p_1, p_2, \cdots, p_{k-1}, p_k\} \tag{2}$$

Then, in the case of branch k removed, such equations need to solve:

$$\begin{bmatrix} \Delta \boldsymbol{P}_{ki} \\ \Delta \boldsymbol{Q}_{ki} \end{bmatrix} = \boldsymbol{J}_{ki} \boldsymbol{X}_{ki} (1 \le i \le P) \tag{3}$$

In order to reduce times of N-R method used during the N-1 security process, the node power injection correction vectors and Jacobi matrices can be concatenated respectively. According to the linear algebra theory, such equations need to solve:

$$\begin{bmatrix} \Delta \boldsymbol{P}_{1i} \\ \Delta \boldsymbol{Q}_{1i} \\ \Delta \boldsymbol{P}_{2i} \\ \Delta \boldsymbol{Q}_{2i} \\ \cdots \\ \cdots \\ \Delta \boldsymbol{P}_{ni} \\ \Delta \boldsymbol{Q}_{ni} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_{1i} & & & \\ & \boldsymbol{J}_{2i} & & \\ & & \cdots & \\ & & & \boldsymbol{J}_{ni} \end{bmatrix} \begin{bmatrix} \boldsymbol{Y}_{1i} \\ \boldsymbol{Y}_{2i} \\ \cdots \\ \boldsymbol{Y}_{ni} \end{bmatrix} (1 \le i \le P) \tag{4}$$

The solutions of the above equations satisfy:

$$\boldsymbol{Y}_{ki} = \boldsymbol{X}_{ki} (1 \le i \le P, 1 \le k \le n) \tag{5}$$

It can be seen that the concatenated power flow problem needs N-R method solving for only once. However, the scale of the new problem will be much larger than the independent problems, which means that when the scale is large enough, using direct method such as LU decomposition to solve equations needs to take a long time. Thus, matrix preprocessing technique, iterative method and parallel processing can be used to accelerate equations solving, which will be discussed in detail in Section IV. Section V-C further proves that the concatenation algorithm has higher computational efficiency than solving the independent problems one by one.

III. Acceleration of Linear Equations Solving

A. Feature Analysis of Concatenated Linear Equations

When scale of a system is small, the scale of concatenated linear equations is also small, and direct method, such as LU decomposition, can be adopted to solve the equations. LU decomposition can be implemented by some mature libraries like SuperLU [6]. However, when scale of a system is large, the concatenated Jacobi matrix is large, sparse and asymmetric, and the condition number of the matrix is much larger than 1. The large scale of the system makes the direct method difficult to satisfy computing requirement, so that the iterative method is one of the important approaches to solve large-scale linear equations. In addition, due to the large condition number, it is necessary to select stable iterative methods as well as appropriate preprocessing techniques. The cut-off point of the direct method and the iterative method needs to be further determined by cases.

B. An Iterative Method with Incomplete LU Decomposition Preprocessing Based on Krylov Subspace Theory

The iterative method based on Krylov subspace theory is a kind of important iterative method for solving large linear equations [7]. The general projection method for solving linear equations like $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is to seek an approximate solution x_m from the m-dimensional affine subspace $x_0 + K_m$ (called search space), which uses the Petrov-Galerkin condition:

$$\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_m \perp \boldsymbol{L}_m \tag{6}$$

In this expression, L_m is another m-dimensional subspace (called constraint space). Here, x_0 represents one initial guess of the solution. Krylov subspace means the subspace K_m

$$\boldsymbol{K}_m(\boldsymbol{A}, \boldsymbol{v}) = span\{\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v}, \boldsymbol{A}^2\boldsymbol{v}, \cdots, \boldsymbol{A}^{(m-1)}\boldsymbol{v}\} \tag{7}$$

v can be chosen as the initial residual r_0. Meanwhile, the choice of constraint space L_m will have important influence on the iterative method. In this paper, considering that the concatenated Jacobi matrix is asymmetric, the biorthogonal method, e.g. the stabilized biorthogonal conjugate gradient (BICGSTAB) method, can be applied, and let:

$$\boldsymbol{L}_m = \boldsymbol{K}_m\left(\boldsymbol{A}^T,\ r_0\right) \tag{8}$$

The specific steps of the iterative method proposed in this paper are shown as follows:

- (a) First, use the ILU(0) decomposition to obtain the preconditioner M, which is a matrix. Processing speed of ILU(0) preconditioning is fast, and it will not create non-zero element injection jeopardizing sparsity of the coefficient matrix [8].
- (b) Take the initial guess of $\boldsymbol{X}$ x_0, and tolerance ε, to calculate $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{J}\boldsymbol{x}_0$. Let $\boldsymbol{r}_0^* = \boldsymbol{r}_0$, $j = 1$.
- (c) Calculate $\rho_{j-1} = (\boldsymbol{r}_{j-1}, \boldsymbol{r}_0^*)$. If $\rho_{j-1} = 0$, the method fails, otherwise go to Step (d)。
- (d) If $j = 1$, let $\boldsymbol{p}_j = \boldsymbol{r}_{j-1}$, otherwise, let $\beta_{j-1} = (\rho_{j-1} / \rho_{j-2})(\alpha_{j-1} / \omega_{j-1})$, $\boldsymbol{p}_j = \boldsymbol{r}_{j-1} + \beta_{j-1}(\boldsymbol{p}_{j-1} - \omega_{j-1}\boldsymbol{v}_{j-1})$.
- (e) Solve $\boldsymbol{M}\boldsymbol{p} = \boldsymbol{p}_j$ to obtain $\boldsymbol{p}$, and calculate $\boldsymbol{v}_j = \boldsymbol{J}\boldsymbol{p}$, $\alpha_j = \rho_{j-1} / (\boldsymbol{v}_j, \boldsymbol{r}_0^*)$, $\boldsymbol{s} = \boldsymbol{r}_{j-1} - \alpha_j \boldsymbol{v}_j$.
- (f) If $\|\boldsymbol{s}\| \le \varepsilon$, let $\boldsymbol{x}_j = \boldsymbol{x}_{j-1} + \alpha_j \boldsymbol{p}$, and exit iterations.
- (g) Solve $\boldsymbol{M}\hat{\boldsymbol{s}} = \boldsymbol{s}_j$ to obtain $\hat{\boldsymbol{s}}$, let $\boldsymbol{t} = \boldsymbol{J}\hat{\boldsymbol{s}}$, $\omega_j = (\boldsymbol{s}, \boldsymbol{t}) / (\boldsymbol{t}, \boldsymbol{t})$, $\boldsymbol{x}_j = \boldsymbol{x}_{j-1} + \alpha_j \boldsymbol{p} + \omega_j \hat{\boldsymbol{s}}$.
- (h) If $\boldsymbol{x}_j$ satisfies accuracy requirement, exit iterations, otherwise, let $\boldsymbol{r}_j = \boldsymbol{s} - \omega_j \boldsymbol{t}$, $j = j + 1$. Go to Step (c).

Obviously, the main forms of computation of the iterative method proposed above include matrix-vector multiplication, inner product operation, etc., which all have natural parallelism. Parallel processing with GPU can improve the efficiency of the iterative method.

IV. Paraellel Formation of Jacobi Matrix

The formation of Jacobi matrix is another time-consuming step, so that GPU parallel acceleration of Jacobi matrix formation is considered.

A. Parallelism Analysis

In the power flow calculation, in polar coordinates, the Jacobi matrix can be expressed as:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{H} & \boldsymbol{N} \\ \boldsymbol{F} & \boldsymbol{L} \end{bmatrix} \tag{9}$$

Considering the similarity among these four sub-matrices, the matrix H can be taken as an example:

$$H_{ij} = \begin{cases} -U_i U_j (G_{ij}\sin\delta_{ij} - B_{ij}\cos\delta_{ij}) & i \ne j \\ Q_i + B_{ii}U_i^2 & i = j \end{cases} \tag{10}$$

Q_i and U_i respectively represents passive power injection and magnitude of voltage of node i. δ_{ij} represents phase difference between node i and node j. G_{ij} and B_{ij} represents transconductance and transsusceptance of node i and node j. B_{ii} represents self-susceptance of node i.

Expression (7) shows that, every element in the Jacobi matrix is only related to the node admittance, phase difference between two nodes, node voltage and node power injection. There is no dependency between any two elements, so that the formation of Jacobi matrix is naturally parallelizable.

On the other hand, the power flow problems under different contingencies are independent, which indicates that formation of corresponding Jacobi matrices can also be formed in parallel.

B. Concatenation Method in Parellel

1) Sparsity Technology and Storage of the Jacobi Matrix

Considering that the concatenated Jacobi matrix is a sparse matrix, sparsity storage technology can be used to store the Jacobi matrix to save storage space. Take IEEE standard CASE4 as an example, then the position of non-zero elements of original Jacobi matrix is shown on the left side in Fig. 1. If using CSR (Compressed Sparse Row) format [9], then concatenated Jacobi matrix will be stored in the way shown on the right side in Fig. 1.

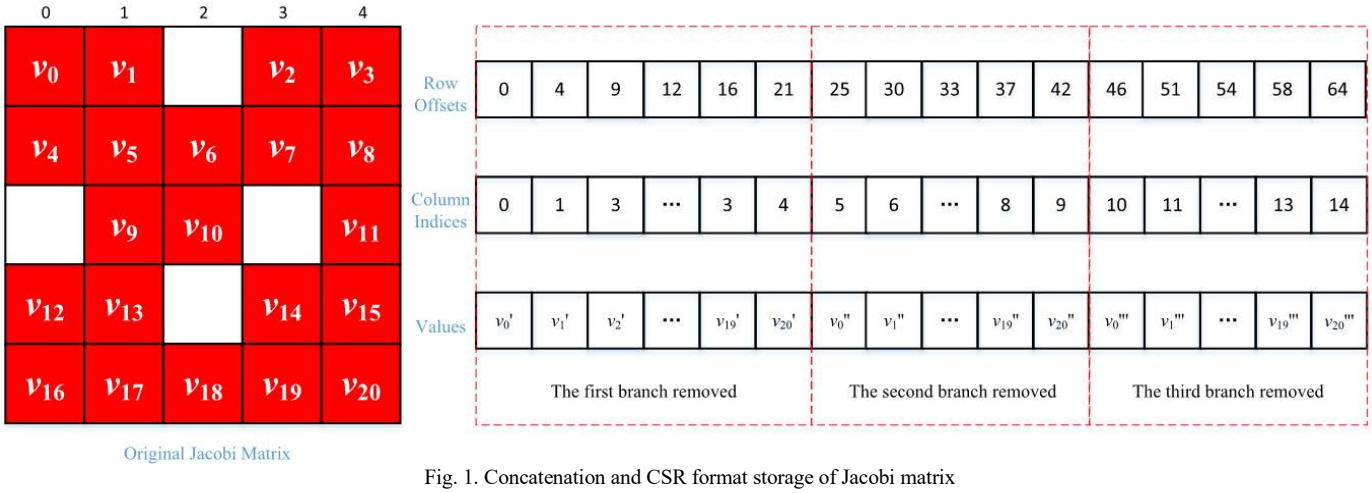

Fig. 1. Concatenation and CSR format storage of Jacobi matrix

2) Parellel Formation Method

According to the analysis in Section III-A, the row offset array, column index array and value array can be generated respectively in parallel by GPU.

a) Parellel Formation of Row Offset Array

The threads with consecutive number in one thread block are arranged to calculate the row offsets, in the concatenated Jacobi matrix, of elements in the same position belonging to different independent power flow problems. Set the number of enabled blocks equal to the order of the Jacobi matrix of the original power flow problem, and set the number of enabled threads in each block equal to the number of elements in the check set.

b) Parellel Formation of Column Index and Value Array

Column index array shares the same length with value array. The threads with consecutive number in one thread block are arranged to calculate the column indices and values, in the concatenated Jacobi matrix, of elements in the same position belonging to different independent power flow problems. Set the number of enabled blocks equal to the number of non-zero elements of the Jacobi matrix of the original power flow problem, and set the number of enabled threads in each block equal to the number of elements in the check set.

In the process of calculation, according to the fact that the Jacobi matrix in the case of one branch removed shares the identical structure with the Jacobi matrix of original power flow problem, the recorded positions of non-zero elements during the formation of original Jacobi matrix can be used to calculate the column index conveniently. On the other hand, after updating elements related to the removed branch in the node admittance matrix, value array can be obtained by using the formula for calculating a Jacobi matrix.

V. Steps of Real-time N-1 ACPF Algorithm

The algorithm applies GPU-CPU computing framework, divided into GPU processing part and CPU processing part. CPU processes set of iterative initial value, formation of node admittance matrix, formation of check set, correction of iterative value, convergence judgement, etc., while GPU processes concatenation of Jacobi matrix. Modified equations are solved by CPU or GPU according to scale of equations, in order to achieve the purpose of fast solving. The detailed steps of the algorithm are as follows:

- (a) CPU: Input data of a system and set the tolerance and maximum iteration times of N-R method.
- (b) CPU: Form the node admittance matrix and the Jacobi matrix in the first iteration of the power flow problem in the normal operation. Record the position of nonzero elements in the Jacobi matrix.
- (c) CPU: Check the connectivity when each branch removed, and form the check set *S*, of which number of set elements is *n*.
- (d) CPU: Set the initial values of iteration variables, and set iteration times $i=0$.
- (e) GPU: According to the results from Step (b) and Step (c), using acceleration approach discussed in Section II-A and N-R method, to form concatenated Jacobi matrix and node power injection correction vector. Then, modified equations are obtained.
- (f) CPU or GPU: According to the scale of modified equations, use a suitable approach to solve.
- (g) CPU: Do converge judgement of N-R method using solutions in Step (f). If accuracy demand is satisfied, the algorithm is converged and exit the algorithm. Otherwise, correct iterative values.
- (h) CPU: Let iteration times $i=i+1$. If the number has reached the maximum iteration times preset, the algorithm is not converged and exit the algorithm. Otherwise, go to Step (e).

VI. Case Analysis

IEEE standard cases as well as 'BENCH' (1648 nodes) and 'Coal 2005 NI TP' (610 nodes) from PSS\E are taken as examples for test. The compiler is Microsoft Visual Studio 2013 Update 3 and NVIDIA Nsight Visual Studio Edition, and the program runs on the Windows 10 of 64 bits. The CPU model in the test is Intel Core i7-7700K, with 4.20GHz master frequency and 32GB memory. The GPU model is NVIDIA GeForce GTX1080, supporting CUDA8.0. Set the maximum iteration times of N-R method to 10, and set the tolerance to 0.01. The iterative method requires accuracy of 1e-6.

A. Acceleration Effect Analysis of Parallel Formation of Jacobi Matrix

Cmpare the computing time of the parallel formation with that of serial formation of the Jacobi matrices for every independent power flow problem. Data in Fig.2 is the average computing time of 10 times.

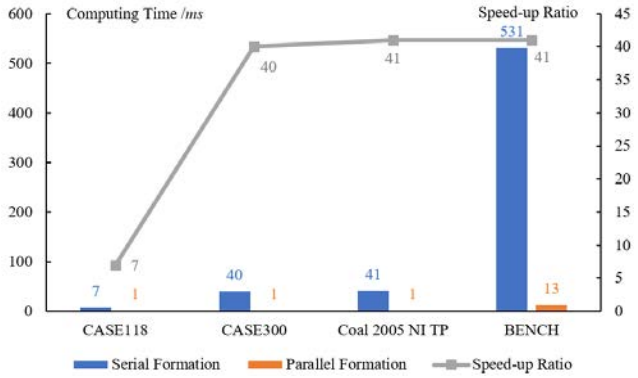

Fig. 2. Acceleration effect analysis of parallel formation of Jacobi matrix

As can be seen in Fig.2, the parallel formation method proposed in this paper has significant acceleration effect. Especially when the scale of system is large, such as Coal 2005 NI TP and BENCH, the speed-up ratio can reach around 40 times. In the N-1 security, it is necessary to form the Jacobi matrix for several times, and the acceleration effect of parallel formation can produce cumulative effect.

B. Comparison of Computing Time of Different Methods and the Choice of Method for Solving Linear Equations

The concatenated equations are solved by the direct method based on SuperLU library, or the iterative method based on the

Krylov subspace theory proposed in this paper, according to the scale of equations. Data in Fig.3 is the average computing time of 10 times.

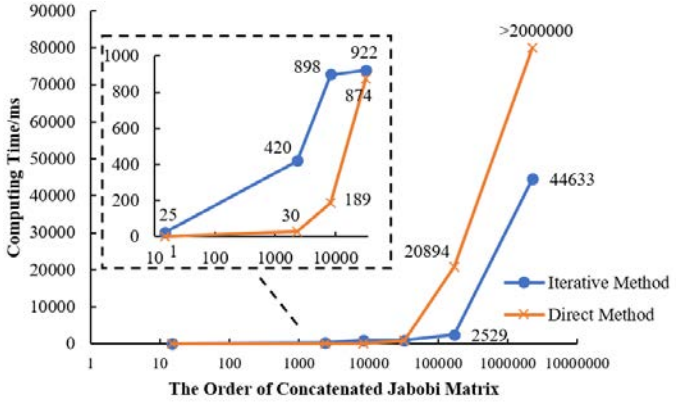

Fig. 3. Comparison of computing time of methods for solving linear equations

According to Fig. 3, 35,000 is the dividing point of the computing time of these two methods. Moreover, the larger the scale of Jacobi matrix, the more obvious the advantage in speed of the iterative method. Therefore, 35,000 can be chosen as the cut-off point of direct method and iterative method.

C. Overall Computational Efficiency Analysis of the Algorithm

In order to test the overall efficiecy of the proposed algorithm, make a comparison between the algorithm proposed in this paper and traditional N-1 AC power flow algorithm (Solve every independent power flow problem one by one).

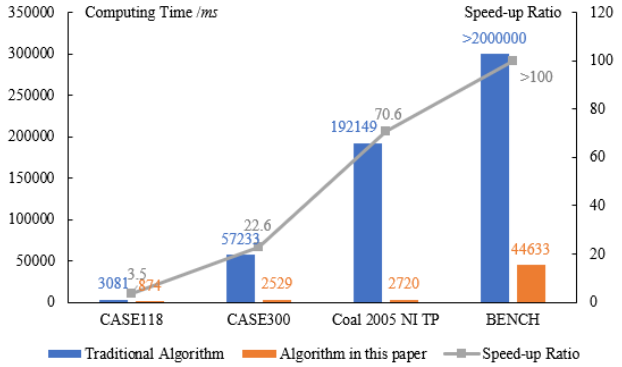

Fig. 4. Comparison of computing time of traditional method and the algorithm proposed in this paper

Fig.4 shows that the algorithm proposed in the paper can achieve significant acceleration. When the scale of system is large, the speed-up ratio is more than 100 times. Moreover, the larger the scale of system is, the more significant the acceleration effect of the algorithm proposed in the paper.

D. Memory Footprint of the Algorithm

TABLE I shows that, when the system has 1648 nodes, the peak value of RAM memory is only 1.69GB, and the peak value of GPU memory is 2.42GB, which a GPU with 4GB memory can support.

TABLE I. MEMORY FOOTPRINT OF THE ALGORITHM

Cases	Order of Concatenated Jacobi Matrix	Peak Value of RAM Memory /MB	Peak Value of GPU Memory /MB
CASE118	32037	390.0	1606.0
CASE300	169600	590.3	1606.0
Coal 2005	164889	422.3	1606.0
BENCH	2272284	1730.9	2478.0

VII. CONCLUSION

This paper proposed a real-time N-1 ACPF algorithm based on concatenation method and GPU-CPU computing framework. This algorithm has higher accuracy than DC power flow method and sensitivity analysis method. The concatenation method concatenates all the independent power flow problems, which means that N-R method will be used only once. GPU is used to parallelly process the formation of Jacobi matrix and iterative method for solving large scale linear equations, effectively improving the overall efficiency of the algorithm. Moreover, the algorithm needs small memory space, which can be supported by CPU and GPU equipped by a common PC. Therefore, this algorithm has engineering application value.

REFERENCES

[1] D. Van Hertem, J. Verboomen, K. Purchala, R. Belmans and W. L. Kling, "Usefulness of DC power flow for active power flow analysis with flow controlling devices," *The 8th IEE International Conference on AC and DC Power Transmission*, London, UK, 2006, pp. 58-62.

[2] DU Zhengwang, HA Hengxu, SONG Yang, DUAN Yujing and HU Xitong, "New algorithm based on the sensitivity and the compensation methods for line-outage problem of power network," *Power System Protection and Control*. China, vol.38, pp. 103-107, August 2010.

[3] ZHAO Jinquan, YE Junling and DENG Yong, "Comparative Analysis on DC Power Flow and AC Power Flow," *Power System Technology*. China, vol.36, pp. 147-152, October 2012.

[4] J. Singh and I. Aruni, "Accelerating Power Flow studies on Graphics Processing Unit," *2010 Annual IEEE India Conference (INDICON)*, Kolkata, 2010, pp. 1-5.

[5] Li, Xue, F. Li, and J. M. Clark, "Exploration of multifrontal method with GPU in power flow computation." *IEEE Power and Energy Society General Meeting*, Vancouver, Canada, 2013, pp. 1-5.

[6] P. Sao, X. Liu, R. Vuduc and X. Li, "A Sparse Direct Solver for Distributed Memory Xeon Phi-Accelerated Systems," *2015 IEEE International Parallel and Distributed Processing Symposium*, Hyderabad, India, 2015, pp. 71-81.

[7] R. Idema, D. J. P. Lahaye, C. Vuik and L. van der Sluis, "Scalable Newton-Krylov Solver for Very Large Power Flow Problems," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 390-396, Feb. 2012.

[8] B. Yang, H. Liu, Z. Chen and X. Tian, "GPU-Accelerated Preconditioned GMRES Solver," *IEEE International Conference on Big Data Security on Cloud*, New York, USA, 2016, pp.280-285.

[9] J. L. Greathouse and M. Daga, "Efficient Sparse Matrix-Vector Multiplication on GPUs Using the CSR Storage Format," *SC14: International Conference for High Performance Computing*, Networking, Storage and Analysis, New Orleans, LA, 2014, pp. 769-780.