Fast equilibrium reconstruction for tokamak discharge control based on GPU

Plasma Phys. Control. Fusion **55** (2013) 085016 (9pp) doi:10.1088/0741-3335/55/8/085016

Fast equilibrium reconstruction for tokamak discharge control based on GPU

X N Yue[1,2], B J Xiao[1,2], Z P Luo[2,3] and Y Guo[2]

[1] School of Nuclear Science and Technology, University of Science and Technology of China, Hefei 230026, People's Republic of China
[2] Institute of Plasma Physics, Chinese Academy of Science, Hefei 230031, People's Republic of China

E-mail: zhpluo@ipp.ac.cn

Received 16 November 2012, in final form 25 March 2013
Published 13 June 2013
Online at stacks.iop.org/PPCF/55/085016

Abstract
A parallel code named P-EFIT which could complete an equilibrium reconstruction iteration in 220 μs is described. It is build with the CUDA™ architecture. Some optimization for middle-scale matrix multiplication on graphics processing unit and an algorithm which could solve block tri-diagonal linear system efficiently in parallel is described. Benchmark test is conducted. Static test proves the correctness of the P-EFIT and simulation-test proves the feasibility of using P-EFIT for real-time reconstruction with a 65 × 65 grid.

(Some figures may appear in colour only in the online journal)

1. Introduction

1.1. Background

A high-quality tokamak operation needs an accurate feedback control of the plasma shape. However, it is very difficult to obtain the shape information of plasma directly. The most frequent used method is called equilibrium reconstruction which reconstructs the equilibrium using the diagnostic information. Various equilibrium reconstruction codes had been developed and implemented on tokamak devices [1, 2]. One of the most widely used codes is EFIT [3]. However, the numerical algorithms used by EFIT are generally too central processing unit (CPU) intensive to be used directly for real-time discharge control. In [4], a real-time version of EFIT is described through modifying some algorithms and it could provide a converged equilibrium result in about 30 ms with a 33 × 33 grid. The real-time-EFIT (RT-EFIT) [5] is another real-time version of EFIT which is currently used by DIII-D, KSTAR, NSTX, MAST and EAST [5–9]. The RT-EFIT code made many simplifications and the spatial resolution is relatively low compared with the off-line EFIT. For example, the RT-EFIT used on EAST scales the 2.4 m × 1.4 m rectangle plasma area by 33 × 33 grids, but the off-line equilibrium reconstruction code (EFIT) typically scales the same area by 129 × 129 grids.

Current real-time codes derived from EFIT could meet the real-time demand only if the grid number is substantially downsized, but this would reduce the precision of reconstruction result. The real-time reconstruction algorithm must be accelerated in order to use higher computation grids. In [10], a parallel Grad–Shafranov solver named GPEC (Garching parallel equilibrium code) is described. Through using the discrete sine transform (DST) method and parallelization with openMP, GPEC achieved an impressive acceleration rate over the serial code. However, this code currently just accelerated the Poisson solver of the reconstruction procedure, which counts about 50% computation time in its serial version and about 30% in the RT-EFIT.

This paper describes an equilibrium reconstruction code P-EFIT which accelerated the whole process of the reconstruction. Through efficiently making use of the massively parallel processing cores of graphic processing units (GPU), P-EFIT could obtain a converged equilibrium result in much less time. Compared with the off-line EFIT, P-EFIT is much faster. By cutting the iteration number into one or two, P-EFIT could directly been used for the real-time equilibrium reconstruction. Compared with RT-EFIT currently used by EAST, P-EFIT has a better spatial resolution. With a 65 × 65 grid number, only 220 μs is needed for the P-EFIT to complete one reconstruction iteration.

1.2. Overview of parallel computing with GPU

Attempts to parallelize the equilibrium reconstruction with MPI has been carried out by before [11]. However, the

[3] Author to whom any correspondence should be addressed.

accelerator is far from sufficient for real-time control. That is because during each iteration the amount of computing is relatively low and the algorithm itself does not have enough independence. With conventional parallel architectures, like MPI, the cost of communications between different computing threads would easily absorb the benefits gained from parallel [11]. OpenMP is a better choice as it uses shared memory architecture. The communication speed is much faster. However, under the openMP architecture, the maximum thread number that could run in parallel is restricted by the CPU core number, which is no more than 10 in most modern computers.

The GPU is mainly used to assist the CPU with the rendering of complex graphics. With this design principle, the GPU is made with highly parallel structure. Instead of having several powerful processing cores like CPU, one modern GPU could have several hundred simple processing cores. Coupled with its high memory bandwidth, GPUs could achieve a much stronger floating-point compute capability than CPU. Thread communication is relatively fast and there could be hundreds of threads running in parallel. So this architecture would be a very good choice for the acceleration of equilibrium reconstruction.

CUDA™ (compute unified device architecture) is a parallel computing architecture developed by NVIDIA. CUDA™ provides a software environment with which allows developers to use C style high-level language to write parallel code running on the GPU. A detailed description of CUDA™ can be found in [12].

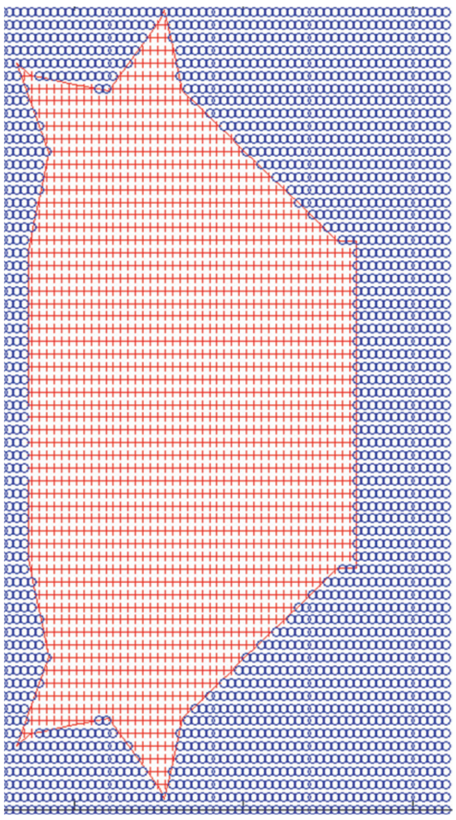


Figure 1. A schematic diagram of the meshed vacuum chamber, in which the red points are grid points inside the limiter while the light blue ones are those outside the limiter.

2. Algorithm details

2.1. *Basic procedure of equilibrium reconstruction algorithm*

P-EFIT takes the algorithm described in [3] as its basic framework. Its principle is to compute the poloidal flux(ψ) distribution in the R, Z plane, then obtain the toroidal current density distribution in the R, Z plane, which could provide a least-square best fit to the diagnostic data under the model given by the G–S equation

$$\Delta^* \psi_\mathrm{P} = -\mu_0 R J_\mathrm{t}(R, \psi), \tag{1}$$

ψ_P represents the flux created by plasma current. If ψ_coil is the flux created by coils, ψ could be obtained with expression $\psi = \psi_\mathrm{P} + \psi_\mathrm{coil}$.

First of all, the plasma area will be meshed into a set of rectangular elements. In the origin EFIT codes, considering the efficient of Buneman's method, the mesh density is generally set to $(2^n+1) \times (2^n+1)$. P-EFIT does not have this limitation. However, in this paper, we still use this regulation for easier comparison. Figure 1 shows a schematic diagram of the meshed vacuum chamber, where we could find that many elements are out of the limiter. Like RT-EFIT, P-EFIT set plasma currents of these elements to zero before each iteration which could reduce some computing time.

Elements will be initialized with a typical poloidal flux distribution. Having the ψ in each element, plasma current density will be represented with several free parameters through the following functions [5]:

$$J_\mathrm{t}(R, \psi; \alpha_J, \gamma_J, \delta_Z) = R\left(P'(\psi; \alpha_J, \delta_Z) + \frac{\mu_0 F F'(\psi; \gamma_J, \delta_Z)}{4\pi^2 R^2}\right), \tag{2}$$

$$P' = \sum_{n=0}^{n_P} \alpha_n \left(\psi_\mathrm{N} + \frac{\partial \psi_\mathrm{N}}{\partial Z}\delta_Z\right)^n, \tag{3}$$

$$FF' = \sum_{n=0}^{n_F} \gamma_n \left(\psi_\mathrm{N} + \frac{\partial \psi_\mathrm{N}}{\partial Z}\delta_Z\right)^n. \tag{4}$$

In these equations, P is the plasma pressure. F is related to the poloidal current. $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_J, \boldsymbol{\gamma}_J, \delta_Z)$ is the set of free parameters in the plasma current model. Expressions (3) and (4) show the polynomial approximations of P and F. Here $\psi_\mathrm{N} = (\psi - \psi_\mathrm{axis})/(\psi_\mathrm{bdy} - \psi_\mathrm{axis})$ is the flux normalized to the flux difference from the centre to the edge of the plasma. δ_Z allows the current profile model to have the freedom to adjust the vertical position during the calculation. Apart from these, in P-EFIT, currents in external poloidal field coils which represented with vector $\boldsymbol{I}_\mathrm{C}$ are also free parameters in the fitting problem.

Now the plasma current could be represented as

$$\boldsymbol{I}_\mathrm{p} = \boldsymbol{H} \times \boldsymbol{\alpha}, \tag{5}$$

$$\boldsymbol{H} = \left(R, R\psi_\mathrm{N} \cdots R\psi_\mathrm{N}^{n_\mathrm{p}}, \frac{1}{R}, \frac{\psi_\mathrm{N}}{R} \cdots \frac{\psi_\mathrm{N}^{n_F}}{R}, C_{\delta_Z}\right). \tag{6}$$

The operation '$\times$' between matrix or vectors in this paper indicates general matrix product. To compute the plasma current value on each grid point, we must obtain the free parameters $\boldsymbol{\alpha}$ by fitting it with the external diagnostics.

The relation between plasma currents and the external diagnostics could be pre-computed and represent as $\boldsymbol{G}_{\mathrm{P}}$. Then the diagnostic data resulting from the plasma current are $\boldsymbol{G}_{\mathrm{P}} \times \boldsymbol{I}_{\mathrm{P}} = \boldsymbol{G}_{\mathrm{P}} \times \boldsymbol{H} \times \boldsymbol{\alpha}$, and the contribution by external coils is $\boldsymbol{G}_{\mathrm{C}} \times \boldsymbol{I}_{\mathrm{C}}$, where $\boldsymbol{G}_{\mathrm{C}}$ is the pre-computed matrix showing the relation between external coils and the external diagnostics. If $\boldsymbol{D}$ represents the measured diagnostic data vector and $\boldsymbol{\Gamma}$ represents the response matrix, which is defined as $(\boldsymbol{G}_{\mathrm{C}} \quad \boldsymbol{G}_{\mathrm{P}} \times \boldsymbol{H})$. The unknown vector $\boldsymbol{U}$ is defined as $(\boldsymbol{I}_{\mathrm{C}} \quad \boldsymbol{\alpha})^{\mathrm{T}}$, and $\boldsymbol{F}$ is the weight vector whose element is defined as $1/\sqrt{(\sigma_{\mathrm{r}} \times d)^2 + \sigma_d^2}$, where $\sigma_{\mathrm{r}} \times d$ is the random uncertainty of the diagnostic measurement. d stands for the diagnostic value and σ_{r} is the maximum relative error of that diagnostic value. On EAST, σ_{r} was usually set to 0.03 or 0.05. σ_d is the systematic uncertainties of that diagnostic system. Then the relationship between the diagnostic and the unknown vector could be written as

$$\boldsymbol{F} \cdot \boldsymbol{D} = (\boldsymbol{F} \cdot \boldsymbol{\Gamma}) \times \boldsymbol{U}, \tag{7}$$

where the operator '$\cdot$' indicates multiplication of each column of the matrix by the vector. This is an over-determined system which could be solved through least-square best fit. Once $\boldsymbol{\alpha}$ is obtained, the plasma current profile could be computed using expression (5). After this, the poloidal flux ψ should be updated, and a new ψ_{N} distribution will be presented which will be used to compute a new set of $\boldsymbol{\Gamma}$ and $\boldsymbol{U}$. Through this iteration, a converged equilibrium result would be obtained.

The updating of ψ is a very important part of the reconstruction code, especially the flux induced by plasma current which is represented as ψ_{P}. The easiest way to obtain this is using Green's function to pre-compute a matrix $\boldsymbol{K}_{\mathrm{P}}$, and multiplying this matrix with the plasma current vector $\boldsymbol{I}_{\mathrm{P}}$, through which the ψ_{P} on each grid point would be obtained as follows:

$$\boldsymbol{\Psi}_{\mathrm{P}} = \boldsymbol{K}_{\mathrm{P}} \times \boldsymbol{I}_{\mathrm{P}}. \tag{8}$$

However, if the grid number is $k \times k$, then $\boldsymbol{K}_{\mathrm{P}}$ will be a $k^2 \times k^2$ matrix, which means the calculation would be very time consuming. So the equilibrium reconstruction codes use different approaches. The difference discrete of the G–S equation would create a block tri-diagonal linear system with a Dirichlet boundary condition. There are several algorithms that could solve this problem including Buneman's method adopted by EFIT and single cyclic reduction used by RT-EFIT.

The basic flow path of the equilibrium reconstruction code could be written as:

(1) Input an initial ψ_{N} distribution.
(2) Compute the response matrix $\boldsymbol{\Gamma}$.
(3) Compute the least-square best fit result of the unknown current arguments.
(4) Refresh the plasma current profile and then compute the new ψ_{N} distribution.
(5) Check the error. If not converged, go back to step 2 and continue, else output the result.

2.2. *Parallelization of the equilibrium reconstruction algorithm*

Steps 2, 3, and 4 of the equilibrium reconstruction algorithm take a comparable amount of time, so high parallelism degree must be achieved for each step.

In this paper, without specification, the grid number will be 65×65, the polynomial coefficient number will be 3 ($n_P = 2$, $n_F = 1$), the external magnetic diagnostic used for reconstruction is 74 and tests are conducted on a workstation running Ubuntu 10.10 with a four-core Intel Xeon® E3-1200 and a NVIDIA Tesla C2050 GPU.

2.2.1. Parallelization of response matrix calculation. The most time-consuming part of step 2 includes the calculation of $\boldsymbol{G}_{\mathrm{P}} \times \boldsymbol{H}$ and the determination of boundary flux. The exact size of $\boldsymbol{G}_{\mathrm{P}}$ and $\boldsymbol{H}$ will be changing with the movement of limiter and the number of free parameters in equation (2). On EAST, typically, $\boldsymbol{G}_{\mathrm{P}}$ would be a 74×2000 matrix and $\boldsymbol{H}$ would be a 2000×4 matrix. Serial computation of this matrix multiplication using basic linear algebra subprograms (BLAS) on CPU takes about 900 μs. While using the GPU-accelerated version of BLAS (CUBLAS), same operation could be done in 250 μs, which is still too slow. It is because CUBLAS is mainly designed for large-scale computation, and $\boldsymbol{G}_{\mathrm{P}} \times \boldsymbol{H}$ is only a middle-scale task. For this reason, a specific algorithm is designed for this operation based on the optimization principles of CUDA™ [13]. Through carefully assigning the tasks among every multiprocessor, communication and synchronization between threads is reduced to quite a low level. The idea of this strategy is described in the appendix. Using this algorithm, this matrix multiplication could be complete in 40 μs.

The strategy to determine the boundary flux is similar to the one used by RT-EFIT. First, the poloidal flux at selected limiter points and the possible X points will be calculated. Then we compare the maximum flux value of the selected limiter points with the flux of possible X points and choose the greater one as the boundary flux.

The strategy to compute the flux at possible X points is as follows:

(1) Define two computing boxes, i.e. the black boxes in figure 2, which covers those areas where the X point could appear. Compute B_R and B_Z of the grid points inside the boxes using expression $B_R = -(1/R)(\partial\psi/\partial z)$ and $B_z = -(1/R)(\partial\psi/\partial R)$. The B_R and B_Z are just rough values since the gradient of ψ is computed with a finite difference using the flux value of adjunct grid points. However, this level of precision is acceptable as they were just used to determine the rough position of the X points to prepare for the final calculation of the accurate X point positions.
(2) Find the grid point which has the minimal $B_R^2 + B_z^2$ in both boxes. Then the accurate poloidal field B_R^a and B_z^a on these two points and their adjunct grid points (6 points altogether) are computed using the pre-computed Green function matrix multiply by the current vector, this multiplication would take around 10 μs.

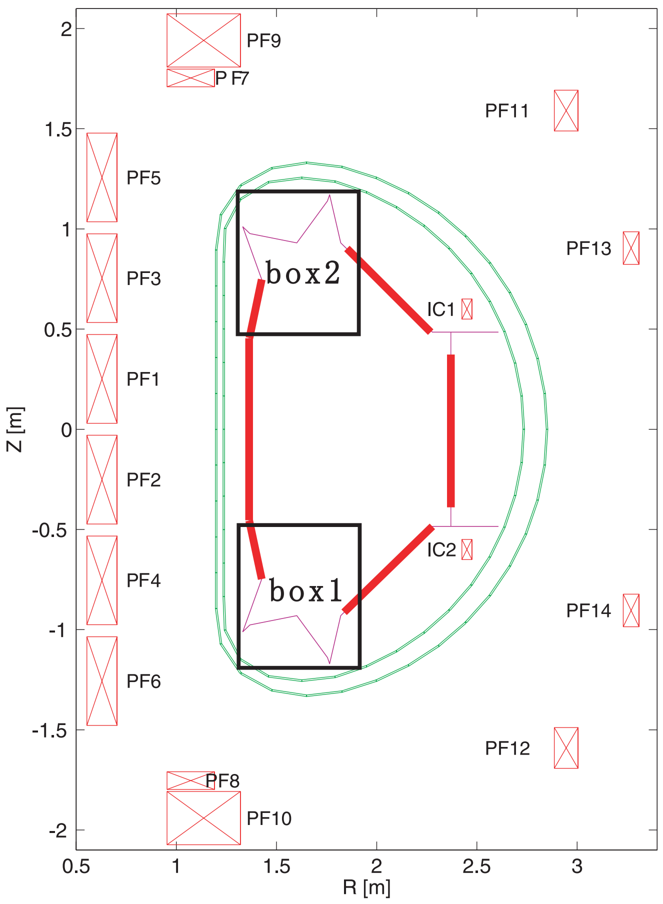


Figure 2. A schematic diagram of the EAST geometry. The two black boxes are the scope of X point locating and their sizes are changeable.

(3) Perform first-order Taylor expansion of B_R^a and B_z^a around the points located in previous step. A linear equation group is solved to locate the precise position of X points. If the X point is too far from the expansion point (>10 cm) or the X point is located outside the boxes, then the X point would not be considered. After locating the X point position, poloidal flux on the X point would be calculated through interpolation.

This strategy is based on the assumption that poloidal magnetic field density is derivable near the point located in step 2. The process of this algorithm is quite parallel. Using the GPU, the X point could be located in 20 μs. The X point found in this way differs from the one computed with EFIT by less than 2 mm.

After locating the X point, the maximum poloidal flux on the X points would be compared with the maximum poloidal flux of selected limiter points. If the latter one is bigger, then it means the X points are outside the last closed flux surface. So it was a limiter discharge, and the maximum flux on that limiter point would be taken as the boundary flux. Otherwise, it was a X point discharge and the poloidal flux at that X point would be taken as the boundary flux.

The limiter point selective strategy is as follows. Limiter points were chosen from the bold segments of the limiter as shown in figure 2 and the detailed limiter point chosen strategy

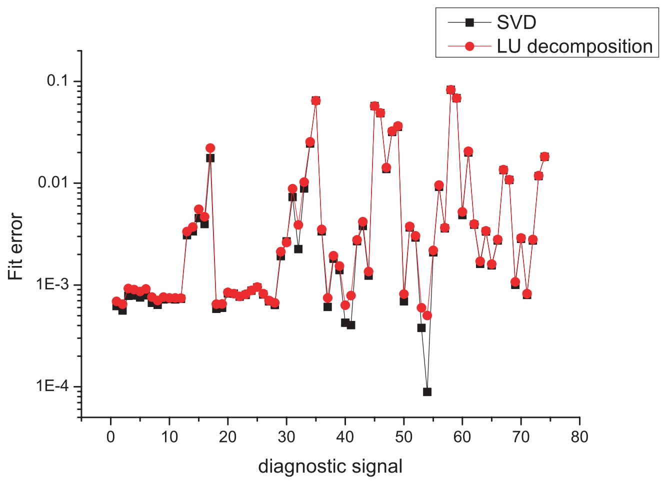


Figure 3. Comparison of the average fit error. Red points are based on reconstruction results using the pivoting L-U decomposition method to solve the normal equation. Black points are based on reconstruction results through direct inverse the response matrix using SVD.

would be determined based on the discharge setup. However, P-EFIT would do the following check in runtime to exclude those points inside the private flux region:

(1) If no null point (where $B_p = 0$) was located in box 1 and box 2, no point will be eliminated.
(2) If a null point was located in box 1, then those limiter points below that null point would not be considered, and if a null point was located in box 2, those limiter points above that null point would not be considered.

The poloidal flux on these selected points would be computed through a pre-computed Green function matrix multiplied by the current vector. Then the parallel is compared to find the largest flux value. It takes about 15 μs to obtain the limiter flux.

The P-EFIT method used to determine the boundary flux is theoretically less accurate compared with EFIT; however, the error is small enough for real-time shape control.

The shape control system of EAST uses the 'isoflux' method [14], which does not need the detailed boundary information. The real-time reconstruction code only needs to pass the relative error of poloidal flux on the control points with the reference flux (boundary flux) and the position of X points to the control system. So the boundary tracing routine used by off-line EFIT is not necessary in P-EFIT. However, the plasma shape display system needs a boundary shape in real-time. So, an existed contouring code was transplanted to the GPU. Since the display system only requires one shape information every 100 ms, this algorithm had not been optimized at all, which would take around 1.5 ms to complete the routine. However, during discharge, this algorithm only needs be triggered every 100 ms, so it has little impact to the speed of P-EFIT. After discharge, this algorithm would also be used to draw the flux contour of arbitrary time-slice.

2.2.2. Parallelization of the least-square best fit. $\boldsymbol{F} \cdot \boldsymbol{D} = (\boldsymbol{F} \cdot \boldsymbol{\Gamma}) \times \boldsymbol{U}$ is an over-determined equation. U could be obtained

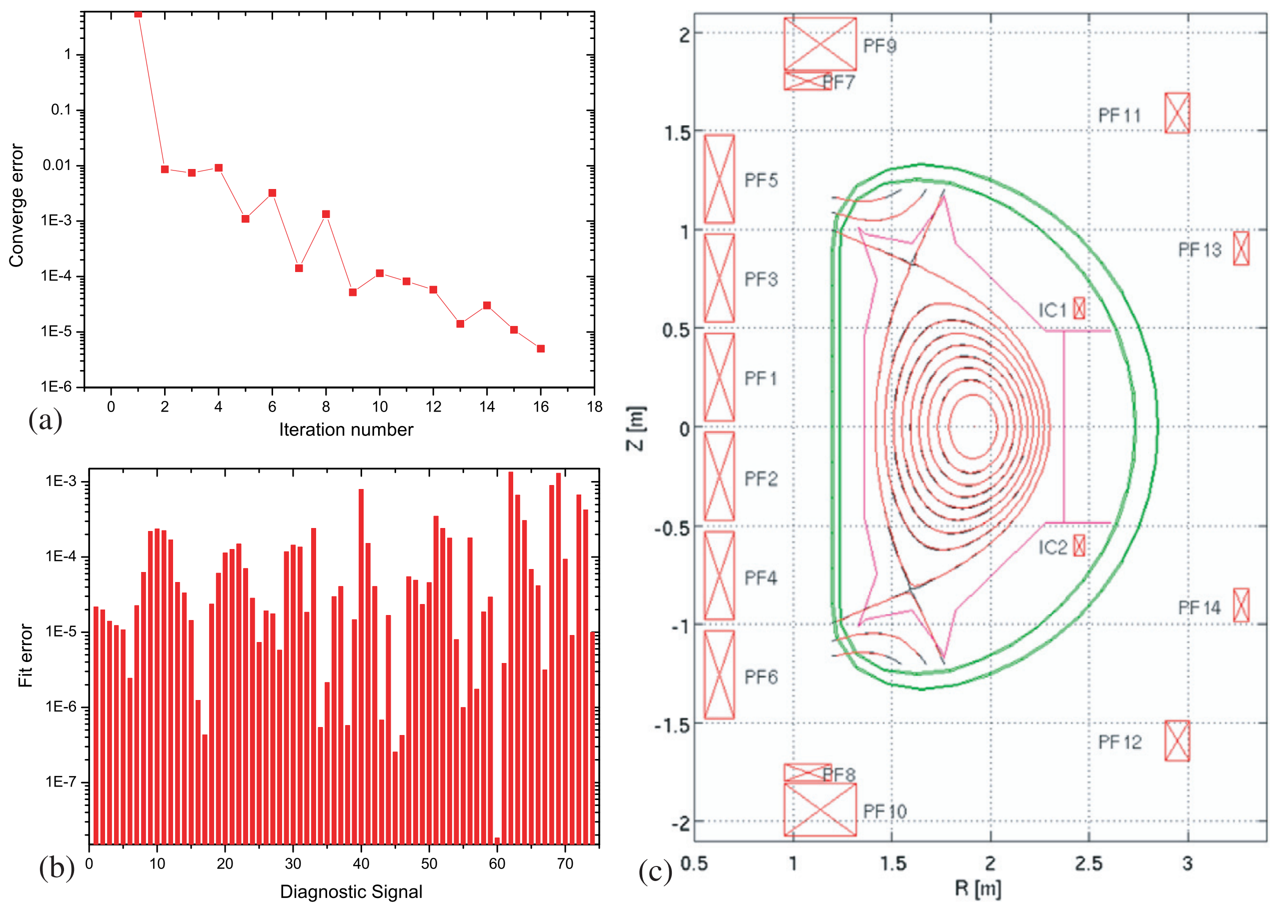
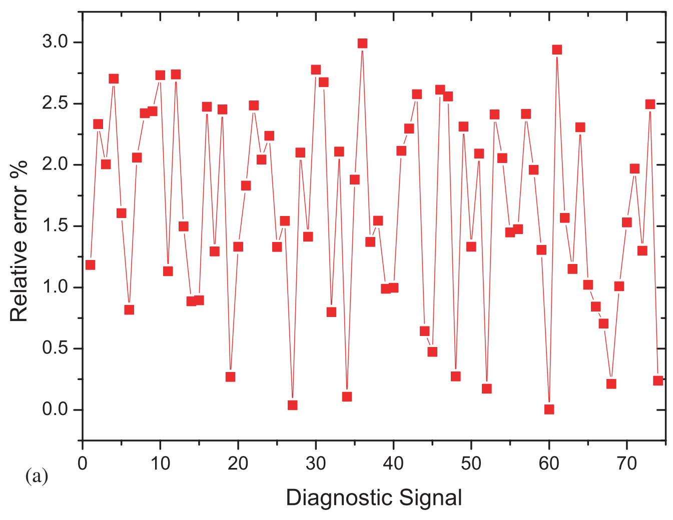


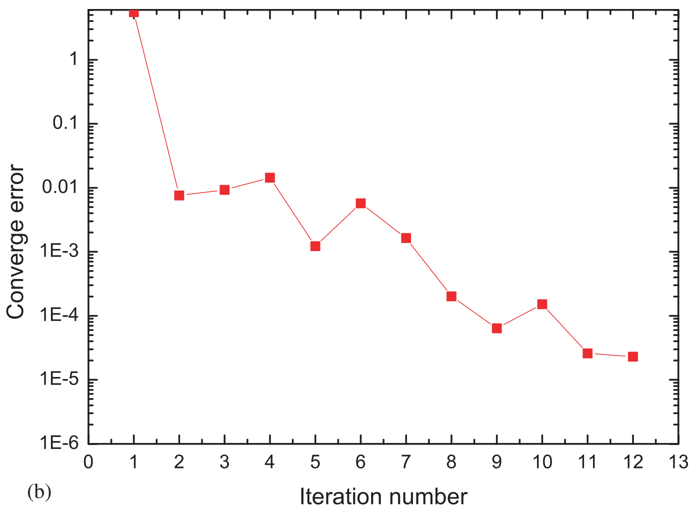
Figure 4. Result of the steady-state test. (*a*) shows the converge error, which is defined as $\max|(\psi^N - \psi^{N-1})|/|\psi_{\mathrm{axis}} - \psi_{\mathrm{bdy}}|$, in which ψ^N means the poloidal flux value of a grid point in the Nth iteration. (*b*) shows the fit quality of the reconstruction. (*c*) shows the comparison between the poloidal flux profiles computed with off-line EFIT (black line) and P-EFIT (red line).

by evaluating the pseudo-inverse matrix of $(\boldsymbol{F}\cdot\boldsymbol{\Gamma})$, which could be represented as $\boldsymbol{U} = (\boldsymbol{F}\cdot\boldsymbol{\Gamma})^{-1} \times (\boldsymbol{F}\cdot\boldsymbol{D})$. However, the computation of $(\boldsymbol{F}\cdot\boldsymbol{\Gamma})^{-1}$ using SVD decomposition is very inefficient on the GPU. So, a different approach is made by solving the norm equation directly. First, multiply both sides of equation (7) by $(\boldsymbol{F}\cdot\boldsymbol{\Gamma})^{\mathrm{T}}$, i.e. $(\boldsymbol{F}\cdot\boldsymbol{\Gamma})^{\mathrm{T}} \times \boldsymbol{F}\cdot\boldsymbol{D} = ((\boldsymbol{F}\cdot\boldsymbol{\Gamma})^{\mathrm{T}} \times (\boldsymbol{F}\cdot\boldsymbol{\Gamma})) \times \boldsymbol{U}$, the initial over-determined system is transformed into a full rank system. Although it is still too slow for the GPU to solve this equation set, we could deliver this task to the CPU, because the time-consuming parts, e.g. matrix multiplications, have been completed on the GPU, and the full rank system is relatively small. For example, on EAST, if external coil current is reconstructed, $(\boldsymbol{F}\cdot\boldsymbol{\Gamma})^{\mathrm{T}} \times (\boldsymbol{F}\cdot\boldsymbol{\Gamma})$ will generate a 17×17 matrix, solving this equation set on the CPU using the LAPACK library with pivot L-U decomposition method needs about $5\,\mu$s and the whole least-square best fit computation, including the overhead of data transfers between the GPU memory and main memory, could be completed in $30\,\mu$s. This method is quick, however, since $(\boldsymbol{F}\cdot\boldsymbol{\Gamma})^{\mathrm{T}} \times (\boldsymbol{F}\cdot\boldsymbol{\Gamma})$ was an ill-conditioned matrix (condition number $\sim 10^{11}$). To verify the feasibility of solving it directly using pivoting L-U decomposition, we choose 1000 EAST experiment equilibriums and do reconstruction using the pivoting L-U decomposition and SVD method. Figure 3 presents the average fit error of both methods, which shows that both methods are accurate. The fit error vector is defined as $(\boldsymbol{\Gamma} \times \boldsymbol{U} - \boldsymbol{D})^2 \cdot \boldsymbol{F}^2$. In the work of Krzysztof Bosak [15], a norm equation of similar condition number is obtained after regularization. He also proves that pivoting L-U decomposition is good enough for such problem.

2.2.3. Parallelization of the poloidal flux refreshing. Having the plasma current value, poloidal flux could be calculated by solving the G–S equation with the finite-difference method. To create a Dirichlet boundary condition, flux values at the boundary of the meshed area must be calculated through a pre-computed Green function matrix multiplied by the plasma current vector. This calculation amount is relatively heavy. It takes the CPU about $800\,\mu$s CPU using BLAS. But with the algorithm described in the appendix, this operation only takes the GPU $30\,\mu$s.

P-EFIT uses similar algorithm with the one described in [10] to solves the block tri-diagonal linear system. The basic principle of this algorithm is using DST to decouple the original problem so that independent part could be solved in parallel [16–18]. After a DST and a transpose, the initial linear equation group is transformed into k independent triangular system with size k, which could be solved in parallel on the GPU using the data parallel method described in [19]. After this, another transpose and DST would be performed in parallel on the flux matrix to obtain the final result.





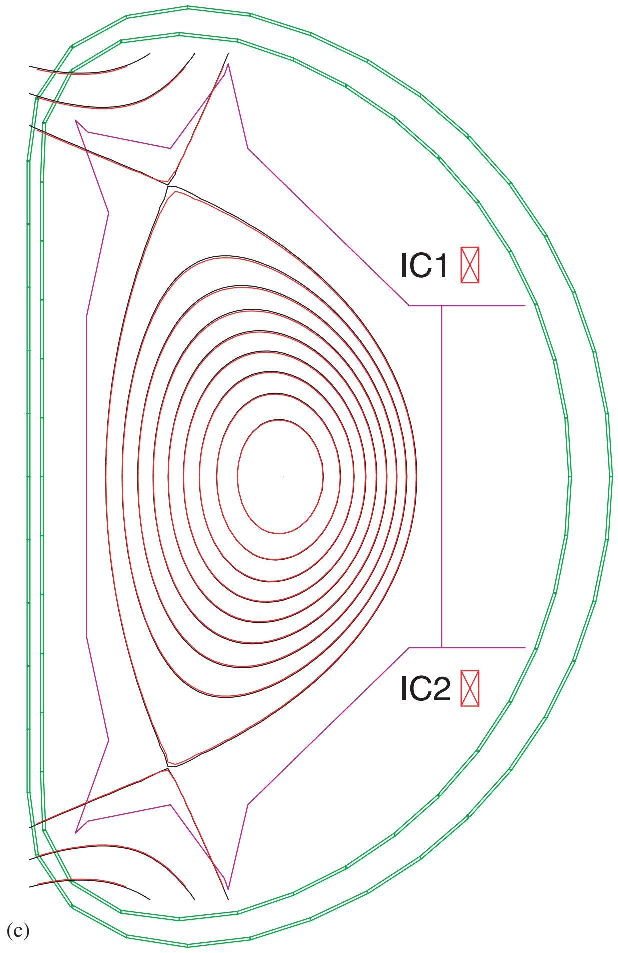


Figure 5. Results of the steady-state test with random noise added to the diagnostic signal. (*a*) shows the relative error we added to the diagnostic signal. (*b*) shows the converge error of each iteration step. (*c*) shows the comparison between the poloidal flux profiles computed by P-EFIT (red line) and the original error free result (black line).

This algorithm achieved impressive speedup ratio on the GPU, it takes only 40 μs to solve this equation set, while on the CPU, it takes 1 ms to complete the same task.

2.2.4. Other parts. In addition to those three parts discussed above, there are some other calculations that is necessary for the equilibrium reconstruction, such as calculating the fit error and converge error, calculating the flux generated by poloidal coils. These parts were also ported to the GPU and achieved high acceleration.

The data transfer between the host memory and the GPU memory also takes some time. Currently, all parts of the reconstruction, except solving the normal equation, are done by the GPU. So there are four data transfers needed for a one iteration reconstruction, including the transfer of diagnostic measurements and parameters needed by shape control system. They will last around 35 μs.

In addition, the GPU kernel launch would also take some time. So the number of kernels in the program should be minimized.

Currently, the total time needed for P-EFIT to complete one complete reconstruction iteration is around 220 μs. Including 80 μs for refreshing the flux, 60 μs for computing the response matrix, 30 μs for least-square best fit and 50 μs for the other parts.

3. Benchmark test

Two benchmark tests are carried out to prove the correctness of P-EFIT. The first one is an one slice equilibrium test with offline-EFIT, and the other one is a discharge pulse simulated with TSC [20].

3.1. One slice equilibrium reconstruction test

In the one slice equilibrium reconstruction test, convergence error and equilibrium quality of the P-EFIT is checked through compare the reconstruction result with the offline-EFIT results.

In this test, set of measurement data on EAST magnetic diagnostics is created by off-line EFIT using a fixed boundary equilibrium calculation mode. These measurement data will be used as the input measured data to P-EFIT. As figure 4(*a*) shows, the computation converged quickly. After about nine iterations, the converge error drops to less than 10^{-4}. At this point, the maximum fit error of P-EFIT which could be figured out from figure 4(*b*) is around 10^{-3}, which indicates a high-quality equilibrium reconstruction. Finally, the poloidal flux computed with P-EFIT is plotted and compared with benchmark result in figure 4(*c*). The shape error is less than 0.2 mm.

In reality, the diagnostic value always has some uncertainty. On EAST, the relative uncertainty is normally less than 2% [21]. To verify the correctness of reconstruction result under such condition, a random relative uncertainty up to 3% is added to the original diagnostic measurements, as figure 5(*a*) shows. The reconstruction result is presented in figures 5(*b*) and (*c*), which demonstrated that the 3% random error did not bring significant impact on the final reconstruction result.

3.2. Simulation test

Through the one slice equilibrium reconstruction test, it is proved that P-EFIT could obtain a well-converged and accurate enough equilibrium result in 8–9 iterations. This process will take around 2 ms. Although P-EFIT is much faster than EFIT, it is still not sufficient for real-time control. For this reason, a strategy similar to RT-EFIT is adopted, whose basic premise is to use the equilibrium result of last time-slice as the initial input for next computation, and perform least-square best fit using the most recent diagnostic data. For each computation, only one iteration is conducted. It is expected that the P-EFIT could provide an accurate enough equilibrium result in each 250 μs using this method.

To verify the feasibility of this approach, another benchmark test is conducted with a set of ideal equilibrium results derived from an ideal discharge shot simulated with TSC. To simulate the real situation, 3% relative error is added to the measurements. Figure 6 illustrated the flux contour map at three time points of this shot. The plasma shape has some obvious changes between during these time points. Figure 7 compares the plasma boundary reconstructed by P-EFIT and the output from TSC, which showed that the boundary reconstructed by P-EFIT using the one iteration strategy was accurate even with 3% noise existed on the measurements. In figure 8, average fit error of 4000 time slices in this shot was computed and compared. Although the presence of noise obviously decreased the fit quality, the fit error was still on a quite low level.

4. Conclusion and future developments

In this paper, we have proved the feasibility of accelerating the equilibrium reconstruction algorithm with CUDA™ and described some important implementation details. The GPU-accelerated reconstruction code P-EFIT could complete a full iteration in around 220 μs, with 65 × 65 grid number, 3 polynomial coefficients and 74 magnetic diagnostics. With this code, an accurate equilibrium result could be obtained in real-time.

CUDA™ is a new technology in parallel computing area and it is developing fast. The GPU we used for test belongs

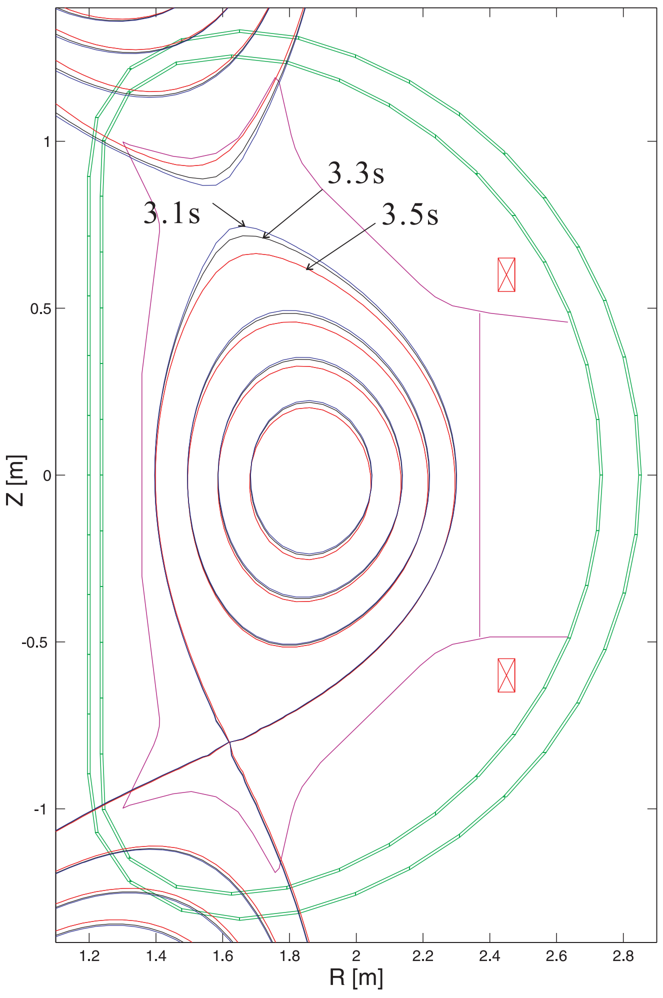


Figure 6. The flux contour maps at three time points of the TSC shot, during which the plasma shape is changing significantly.

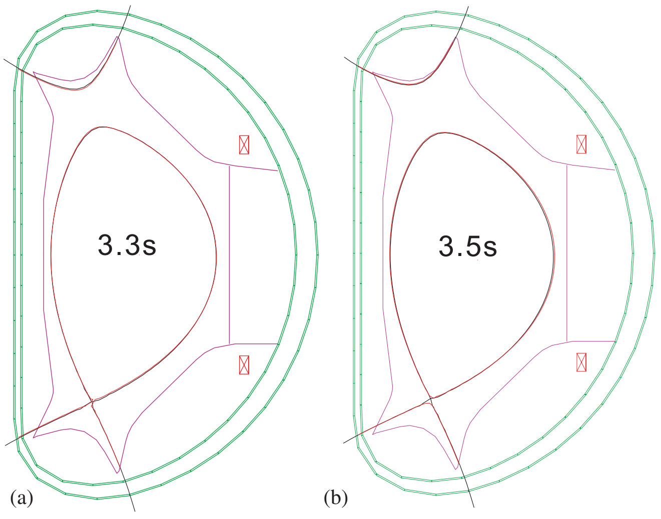


Figure 7. The comparison of P-EFIT reconstruct result (red line) at two time points with the TSC output (black line).

to Fermi architecture. The more powerful GPU architecture ‘Kepler’ had already been released. The running speed of the P-EFIT is expected to be faster on those more powerful GPU.

The study of Q Ren *et al* shows that fine spatial resolution grids are needed to accurately reconstruct the detailed features

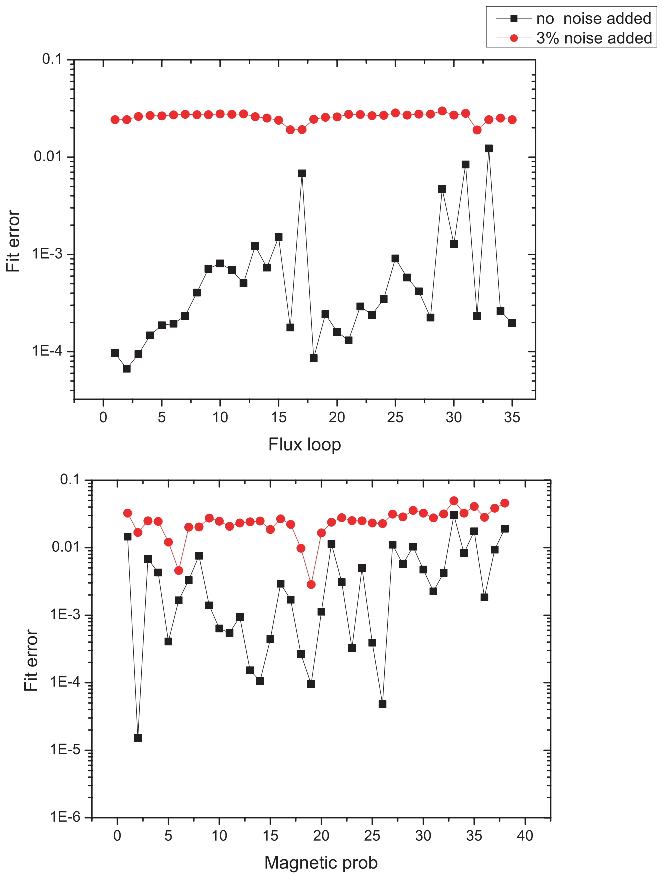


Figure 8. The comparison of average fit error with (red points) and without (black points) random noise added to the measurement.

often present in H-mode pedestal current and pressure profiles and finer spatial grids generally have smaller average force-balance error than coarser grids and those interpolated from coarser grids [22]; however, the computation time increases exponentially when the spatial resolution getting better. The reconstruction of one time-slice will take almost 100 s using 513 × 513 grid on a 4-CPU Linux box with a CPU speed of 2.66 GHz each. This will bring big challenges to the off-line reconstruction during a discharge interval [23]. So in the next step, we are going to use the GPU to accelerate the off-line EFIT in order to perform high spatial resolution reconstruction.

Acknowledgments

This work was supported by the Major State Basic Research Development Program of China (973 program, No 2012GB105000), the National Nature Science Foundation of China (No 10835009) and the Key Project of Knowledge Innovation Program of Chinese Academy of Sciences, project number: KJCX3.SYW.N4.

Appendix

The reconstruction procedure of P-EFIT needs several matrix multiplications, which takes more than 1/3 of the whole computation time. Among these matrix multiplications, the computation of the boundary flux, which is a 256 × 1952 matrix

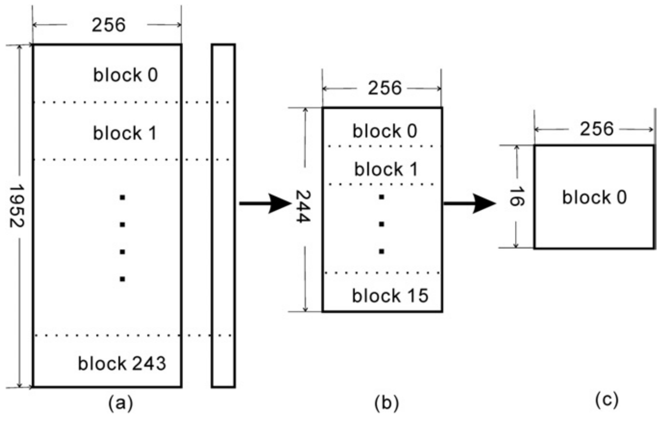


Figure 9. Schematic drawing of the basic idea of matrix multiplication in CUDA. (*a*) shows the original matrix. After computation in first kernel and one global synchronization, the problem was reduced to (*b*). After another kernel computation and global synchronization, a small matrix as (*c*) would be obtained, which could be solved quickly using one block.

multiply a vector with 1952 elements, is most CPU intensive. Following is some detail of the acceleration strategy of this process used by P-EFIT.

Figure 9 shows the basic flow path of this strategy, in figure 9(*a*), the original problem was divided into 244 parts, and each held 8 rows of the matrix. When implemented in CUDA, each part would correspond with a ‘block’, which is a thread union of the CUDA program model. 256 threads would be assigned to every block. Then every thread would multiply 8 elements in the matrix with the corresponding element in the vector. After this step, the original problem was transformed into calculating the sum of rows of a 244 × 256 matrix, as figure 9(*b*) shows. As one block could only run on one of the 14 stream multiprocessors of the GPU, to make full use of the computing ability of GPU, the rest problem was solved using two steps. First, the problem was reduced 16 times using 16 blocks, and then the small matrix was processed using one block to obtain the final result. As threads between different blocks could only communication through the global memory, the whole procedure would require two global synchronizations. The acceleration of other matrix multiplications of P-EFIT took similar strategy with some modification made according to the matrix dimensions. For example, if the column number is small, then more rows could be calculated by one block. The final result could be obtained with only one global synchronization.

References

[1] Beghi A and Cenedese A 2005 Advances in real-time plasma boundary reconstruction—From gaps to snakes *IEEE Control Syst. Mag.* **25** 44–64

[2] Blum J, Boulbe C and Faugeras B 2008 Real-time equilibrium reconstruction in a tokamak *J. Phys.: Conf. Ser.* **135** 012019

[3] Lao L L *et al* 1985 Reconstruction of current profile parameters and plasma shapes in tokamaks *Nucl. Fusion* **25** 1611–22

[4] Zwingmann W 2003 Equilibrium analysis of steady state tokamak discharges *Nucl. Fusion* **43** 842–50

[5] Ferron J R *et al* 1998 Real time equilibrium reconstruction for tokamak discharge control *Nucl. Fusion* **38** 1055–66

[6] Xiao B J *et al* 2012 Recent plasma control progress on EAST *Fusion Eng. Des.* **87** 1887–90
[7] Pangione L, McArdle G and Storrs J 2013 New magnetic real time shape control for MAST *Fusion Eng. Des.* doi:10.1016/j.fusengdes.2013.01.048
[8] Kwak J G *et al* 2012 Key features in the operation of KSTAR *IEEE Trans. Plasma Sci.* **40** 697–704
[9] Gates D A *et al* 2006 Plasma shape control on the National Spherical Torus Experiment (NSTX) using real-time equilibrium reconstruction *Nucl. Fusion* **46** 17–23
[10] Rampp M *et al* 2012 A parallel Grad–Shafranov solver for real-time control of tokamak plasmas *Fusion Sci. Technol.* **62** 409–18
[11] Luo Z 2011 Study of reliability and rapidity for east plasma equilibrium reconstruction *PhD Dissertation* Hefei, Chinese Academy of Science
[12] NVIDIA 2011 *CUDA C Programming guide* v. 4.0 (Santa Clara, CA: NVIDIA Corporation)
[13] Ryoo S *et al* 2008 optimization principles and application performance evaluation of a multithreaded GPU using CUDA *Ppopp'08: Proc. 2008 Acm Sigplan Symp. on Principles and Practice of Parallel Programming (Salt Lake City)* pp 73–82
[14] Hofmann F and Jardin S C 1990 Plasma shape and position control in highly elongated tokamaks *Nucl. Fusion* **30** 2013–22
[15] Bosak K 2001 Real-time numerical identification of plasma in tokamak fusion reactor *Master's Thesis* University of Wroclaw, Poland and http://www.diegm.uniud.it/elettrotecnica/Ricerca/Gruppo_ET/Fusion/BoundaryRec/Pubblications/bosak_equinox_MASTER.pdf
[16] Buzbee B 1992 Poissons-equation revisited—a citation-classic commentary on on direct methods for solving Poissons equations by Buzbee B L, Golub G H, and Nielson C W *Curr. Contents/Phys. Chem. Earth Sci.* **i** 36
[17] Buzbee B L, Golub G H and Nielson C W 1970 Direct methods for solving poissons equations *SIAM J. Numer. Anal.* **7** 627
[18] Hockney R W 1965 A fast direct solution of poissons equation using Fourier analysis *J. ACM* **12** 95
[19] Hillis W D and Steele G L 1986 Data parallel algorithms *Commun. ACM* **29** 1170–83
[20] Jardin S C, Pomphrey N and Delucia J 1986 Dynamic modeling of transport and positional control of tokamaks *J. Comput. Phys.* **66** 481–507
[21] Yang S K, Xiao B J, Luo Z P and Guo Y 2011 Correction of poloidal field coils and magnetic diagnostic system on EAST *Nucl. Fusion Plasma Phys.* **31** 62–7
[22] Ren Q *et al* 2011 High spatial resolution equilibrium reconstruction *Plasma Phys. Control. Fusion* **53** 095009
[23] Luo Z P *et al* 2010 Online plasma shape reconstruction for EAST tokamak *Plasma Sci. Technol.* **12** 412–5