



基于GPU的等离子体快速 平衡重建研究

答辩人：岳小宁

导师：肖炳甲 研究员

专业：核能科学与工程

2013年5月27日



目录

- 实时平衡重建简介
- 基于GPU的并行重建研究
- 测试结果与讨论
- 总结与展望



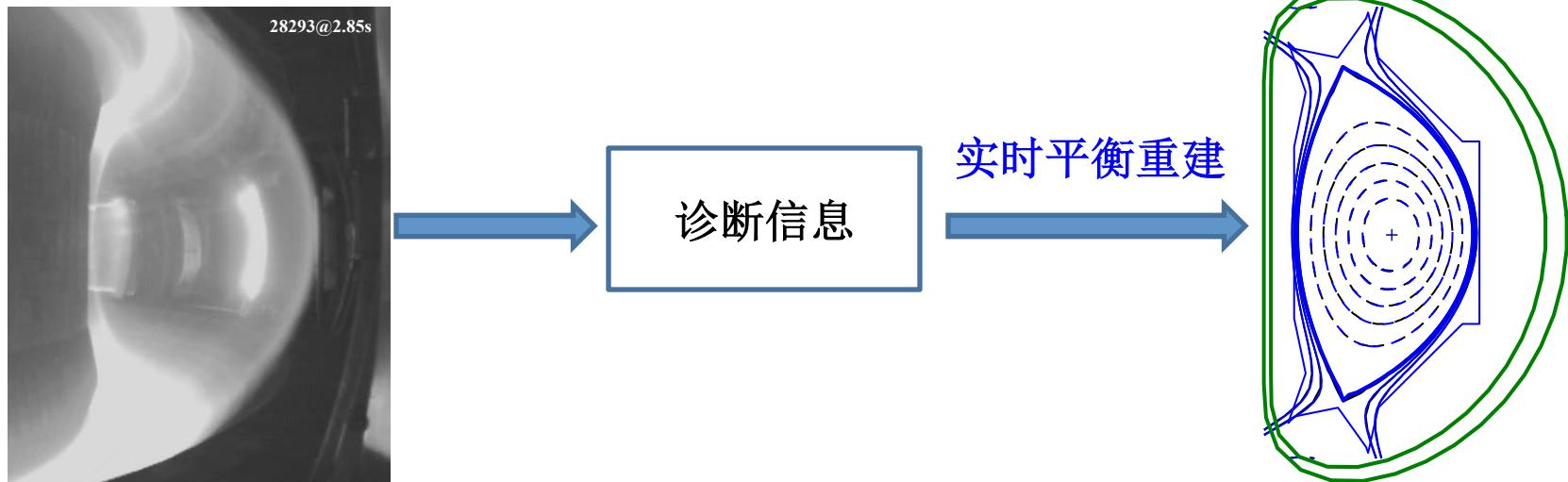
目录

- 实时平衡重建简介
- 基于GPU的并行重建研究
- 测试结果与讨论
- 总结与展望



实时平衡重建：概念

- 托卡马克中等离子体位形的精确反馈控制需要实时获得准确的等离子体位置和形状
- 等离子体的准确位置和形状是无法直接测量的。
- 但是可以通过外部诊断来重建。





实时平衡重建：平衡重建的原理

- Grad-Shafranov方程描述了环向对称的柱形等离子体的平衡方程。
$$R \frac{\partial}{\partial R} \frac{1}{R} \frac{\partial \psi}{\partial R} + \frac{\partial^2 \psi}{\partial z^2} = -\mu_0 R (R p'(\psi) - \frac{\mu_0}{R} F(\psi) F'(\psi))$$
- 将左侧的椭圆微分算子写为 Δ^* ，右侧的等离子体环向电流项写作 $\frac{\mu_0}{R} FF'$ ，将G-S方程简写为：
$$\Delta^* \psi = \mu_0 R^2 p' + \mu_0^2 FF'$$
- 平衡重建的过程可以认为是寻找满足G-S方程的一组极向磁通分布的过程。
- 右侧等离子体电流项的存在使得这一过程变得十分复杂。



实时平衡重建: 实现途径

- 边界重建方法: XLOC

在真空区域求解G-S方程，利用外部诊断信号拟合出真空区域的磁通分布，进而确定出边界位置。

- 间接方法: 神经网络

建立一个数据库，存储大量的已知平衡及其对应的诊断信息。在放电时根据实时诊断信息，建立于已知平衡之间的联系。估计出当前的平衡信息。

- 直接方法: EFIT

直接在整个真空室区域求解G-S方程来获得平衡信息。



实时平衡重建: 直接方法

$$\Delta^* \psi = \mu_0 R^2 p'(\psi) + \mu_0^2 F F'(\psi)$$

- 选取合适的电流模型描述右侧的电流项
- EFIT采用的是多项式电流模型

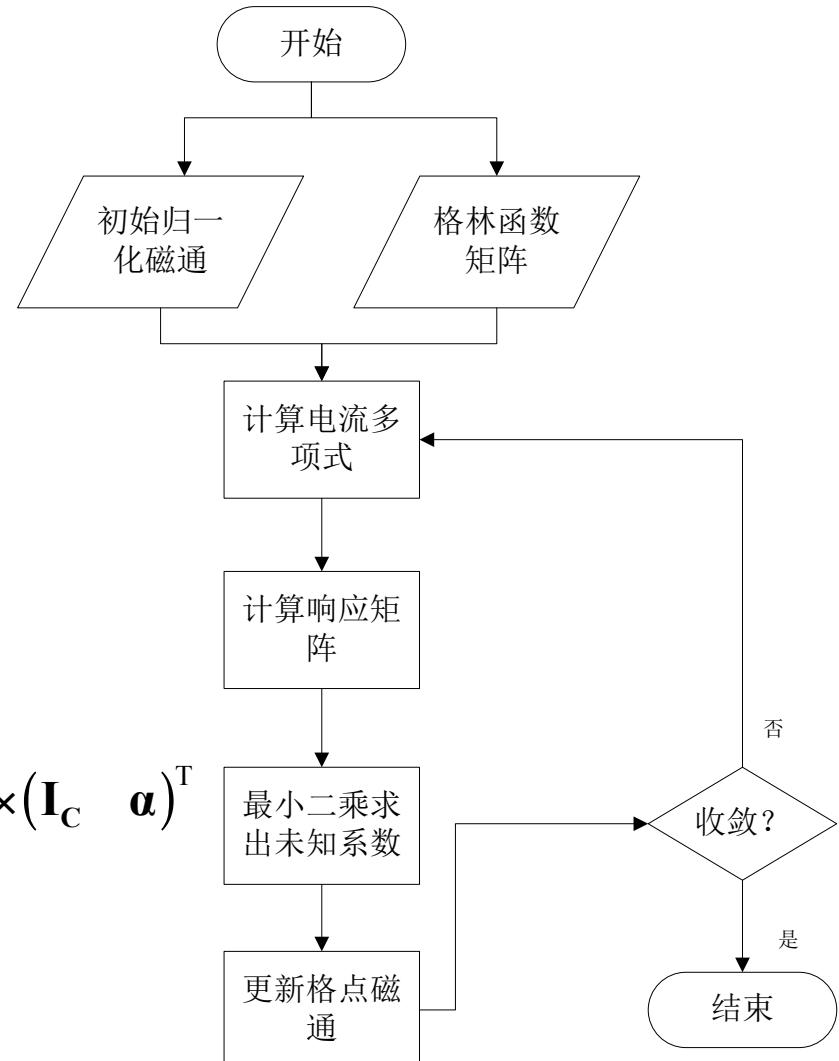
$$p'(\psi) = \sum_{n=0}^{n_p} \alpha_n \psi_N^n$$

$$F F'(\psi) = \sum_{n=0}^{n_F} \gamma_n \psi_N^n$$

- 测量点处的测量值可以表示为 $(\mathbf{G}_c \quad \mathbf{G}_p \times \mathbf{H}) \times (\mathbf{I}_c \quad \mathbf{a})^T$

$$\mathbf{D} = \Gamma \times \mathbf{U}$$

$$\Delta^* \psi^{(N+1)} = \mu_0 R j_\phi(\psi^{(N)})$$





实时平衡重建：直接方法的特点

- 直接方法可以得到等离子体的内部剖面信息。但是，这种方法的计算量很大。
- 采用这一方法的平衡重建程序，如RT-EFIT，均采用了较少的离散网格（ 33×33 ）。在这种情况下一次Picard迭代仍需要2~3ms左右。
- 当前的研究目标，增加离散网格数量（ 65×65 ），增加新的诊断。同时要保证足够的实时性。
- 串行下， 65×65 的反演一次迭代需要10ms以上的时间。目前主要是通过并行计算的方法来提高运行速度。
- 国际上目前采用的主要加速手段为OpenMP。目前只实现了对其中部分算法的加速。



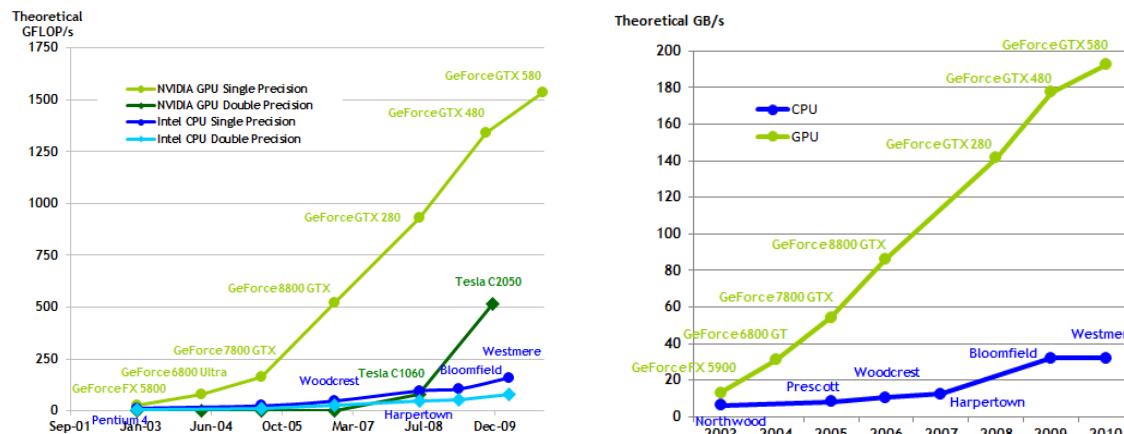
目录

- 实时平衡重建简介
- 基于GPU的并行重建研究
- 测试结果与讨论
- 总结与展望



基于GPU的并行重建研究：GPU并行概念

- GPU (Graphic Processing Unit)是上世纪九十年代末出现的一种专门用来执行图象渲染任务的协处理器。



英伟达公司在2006年底提出CUDA（Compute Unified Device Architecture）架构



基于GPU的并行重建研究：Why GPU?

- 反演算法中存在一些可以并发执行的部分，GPU上有数量庞大的并行执行单元，可以高效的执行这些部分。
- 反演算法中的不同计算线程需要频繁的通信和同步。而GPU的轻量级线程同步、通信以及调度的速度为CPU线程的1000倍左右。
- 反演算法过程中会产生大量的数据吞吐，而GPU的带宽远大于传统的CPU。
- 并行重建并不是一个单一的算法，由并行计算的阿姆达尔定律可知，必须保证每一个模块都有较高的加速比，才能获得总得理想的加速比。

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}}$$



基于GPU的并行重建研究：磁通的更新

- 在获得等离子体环向电流之后，需要更新整个等离子体区域的磁通分布

$$R \frac{\partial}{\partial R} \frac{1}{R} \frac{\partial \psi}{\partial R} + \frac{\partial^2 \psi}{\partial z^2} = -\mu_0 R j_\phi$$

- 利用格林函数法进行格点磁通更新的时间复杂度为O(m⁴)。

$$\frac{\psi_{i-1,j} - 2\psi_{i,j} + \psi_{i+1,j}}{(\Delta R)^2} + \frac{1}{R_i} \frac{\psi_{i-1,j} - \psi_{i+1,j}}{2\Delta R} + \frac{\psi_{i,j-1} - 2\psi_{i,j} + \psi_{i,j+1}}{(\Delta Z)^2} = -\mu_0 R_i j_{i,j}$$

$$\begin{bmatrix} \mathbf{A} & -d_1 \mathbf{I} & & \\ -b_2 \mathbf{I} & \mathbf{A} & -d_2 \mathbf{I} & \\ & -b_3 \mathbf{I} & \mathbf{A} & -d_3 \mathbf{I} \\ & & 0 & 0 & 0 \\ & & & 0 & 0 & -d_{M-1} \mathbf{I} \\ & & & & -b_M \mathbf{I} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \\ M \\ \vdots \\ \Psi_M \end{bmatrix} = \begin{bmatrix} S_2 + b_1 \Psi_0 \\ S_3 \\ S_4 \\ M \\ \vdots \\ S_M + d_M \Psi_{M+1} \end{bmatrix}$$



基于GPU的并行重建研究：磁通的更新

$$\begin{bmatrix} \mathbf{A} & -d_1\mathbf{I} & & \\ -b_2\mathbf{I} & \mathbf{A} & -d_2\mathbf{I} & \\ & -b_3\mathbf{I} & \mathbf{A} & -d_3\mathbf{I} \\ & & 0 & 0 & 0 \\ & & & 0 & 0 & -d_{M-1}\mathbf{I} \\ & & & & -b_M\mathbf{I} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \\ M \\ \vdots \\ \Psi_M \end{bmatrix} = \begin{bmatrix} S_2 + b_1\Psi_0 \\ S_3 \\ S_4 \\ M \\ \vdots \\ S_M + d_M\Psi_{M+1} \end{bmatrix}$$

$$\Psi_i = \begin{bmatrix} \psi_{i,1} \\ \psi_{i,1} \\ M \\ \vdots \\ \psi_{i,N} \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 2(1+c) & -c & & 0 \\ -c & 2(1+c) & -c & \\ & & 0 & \\ 0 & 0 & -c & 2(1+c) \end{bmatrix} \quad \mathbf{S}_i = \begin{bmatrix} (\Delta R)^2 \mu_0 R_i j_{i,1} + c\psi_{i,0} \\ (\Delta Z)^2 \mu_0 R_i j_{i,2} \\ M \\ (\Delta Z)^2 \mu_0 R_i j_{i,N} + c\psi_{i,N+1} \end{bmatrix}$$

- 利用DST算法可以高效的在GPU上求解这一块三对角方程组。



DST算法求解块三对角方程：原理

$$\begin{bmatrix} \mathbf{A} & -d_1\mathbf{I} & & & \\ -b_2\mathbf{I} & \mathbf{A} & -d_2\mathbf{I} & & \\ & -b_3\mathbf{I} & \mathbf{A} & -d_3\mathbf{I} & \\ & & 0 & 0 & 0 \\ & & & 0 & 0 & -d_{M-1}\mathbf{I} \\ & & & & -b_M\mathbf{I} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \\ M \\ \vdots \\ \Psi_M \end{bmatrix} = \begin{bmatrix} S_2 + b_1\Psi_0 \\ S_3 \\ S_4 \\ M \\ \vdots \\ S_M + d_M\Psi_{M+1} \end{bmatrix}$$

- 对系数矩阵中的 \mathbf{A} 做特征值分解，其中 \mathbf{Q} 为一个正交矩阵

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$$

- 于是上述块三对角方程中的每一项可以表示为

$$-b_i\Psi_{i-1} + \mathbf{Q}\Lambda\mathbf{Q}^T\Psi_i - d_i\Psi_{i+1} = \mathbf{g}_i$$

- 左右同时乘以 \mathbf{Q}^T ，可以将上式转化为：

$$-b_i\mathbf{Q}^T\Psi_{i-1} + \Lambda\mathbf{Q}^T\Psi_i - d_i\mathbf{Q}^T\Psi_{i+1} = \mathbf{Q}^T\mathbf{g}_i$$



DST算法求解块三对角方程：原理

$$-b_i \mathbf{Q}^T \Psi_{i-1} + \Lambda \mathbf{Q}^T \Psi_i - d_i \mathbf{Q}^T \Psi_{i+1} = \mathbf{Q}^T \mathbf{g}_i$$

- 令 $\mathbf{Q}^T \Psi_i = \hat{\Psi}_i \quad \mathbf{Q}^T \mathbf{g}_i = \hat{\mathbf{g}}_i$

$$-b_i \hat{\Psi}_{i-1} + \Lambda \hat{\Psi}_i - d_i \hat{\Psi}_{i+1} = \hat{\mathbf{g}}_i$$

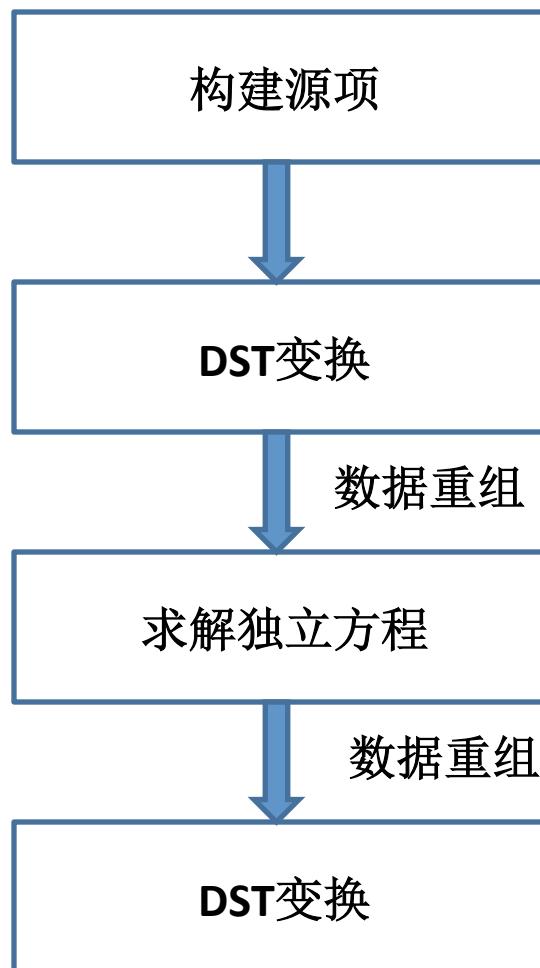
- 抽出其中每一项的第 j 个子方程，进行重组，可以得到新的线性系统

$$\begin{pmatrix} \mathbf{T}_1 & & & \\ & \mathbf{T}_2 & \ddots & \\ & \ddots & \ddots & \\ & & & \mathbf{T}_N \end{pmatrix} \begin{pmatrix} \tilde{\Phi}_1 \\ \tilde{\Phi}_2 \\ \vdots \\ \tilde{\Phi}_N \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{g}}_1 \\ \tilde{\mathbf{g}}_2 \\ \vdots \\ \tilde{\mathbf{g}}_N \end{pmatrix}$$

- 可以看出，原来的线性系统被转化为了 N 个完全独立的线性系统，在 GPU 上可以实现完全并发的求解。



DST算法求解块三对角方程：流程





DST算法求解块三对角方程：独立方程的求解

$$\begin{pmatrix} b_1 & c_1 & & \\ a_1 & b_2 & 0 & \\ 0 & 0 & c_{n-1} & \\ & a_{n-1} & b_n & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ M \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ M \\ d_n \end{pmatrix}$$

- 这类三对角方程一般采用一种称作追赶法的特殊高斯消元方法来进行求解
- 向前消元:
$$b'_i = b_i - (a_{i-1} / b'_{i-1})c_{i-1}$$
$$d'_i = d_i - (a_{i-1} / b'_{i-1})d'_{i-1}$$
- 向后回代:
$$x_n = d'_n / b'_n$$
$$x_i = (d'_i - c_i x_{i+1}) / b'_i$$



DST算法求解块三对角方程：独立方程的求解

- 向前消元：
$$b'_i = b_i - (a_{i-1} / b'_{i-1})c_{i-1}$$
$$d'_i = d_i - (a_{i-1} / b'_{i-1})d'_{i-1}$$
- 将已知量提前计算存入内存中，可以将这一阶段简化为

$$d'_i = d_i - k_{i-1}d'_{i-1}$$

$$d'_1 = d_1$$

$$d'_2 = d_2 - k_1 d'_1$$

$$d'_3 = d_3 - k_2 d'_2 = d_3 + (-k_2) d_2 + (-k_2)(-k_1) d_1$$

$$d'_4 = d_4 + (-k_3) d_3 + (-k_3)(-k_2) d_2 + (-k_3)(-k_2)(-k_1) d_1$$

M

$$d'_n = d_n + (-k_{n-1}) d_{n-1} + (-k_{n-1})(-k_{n-2}) d_{n-2} + \dots + \prod_{i=1}^{n-1} (-k_i) d_1$$



DST算法求解块三对角方程：独立方程的求解

$$d'_1 = d_1$$

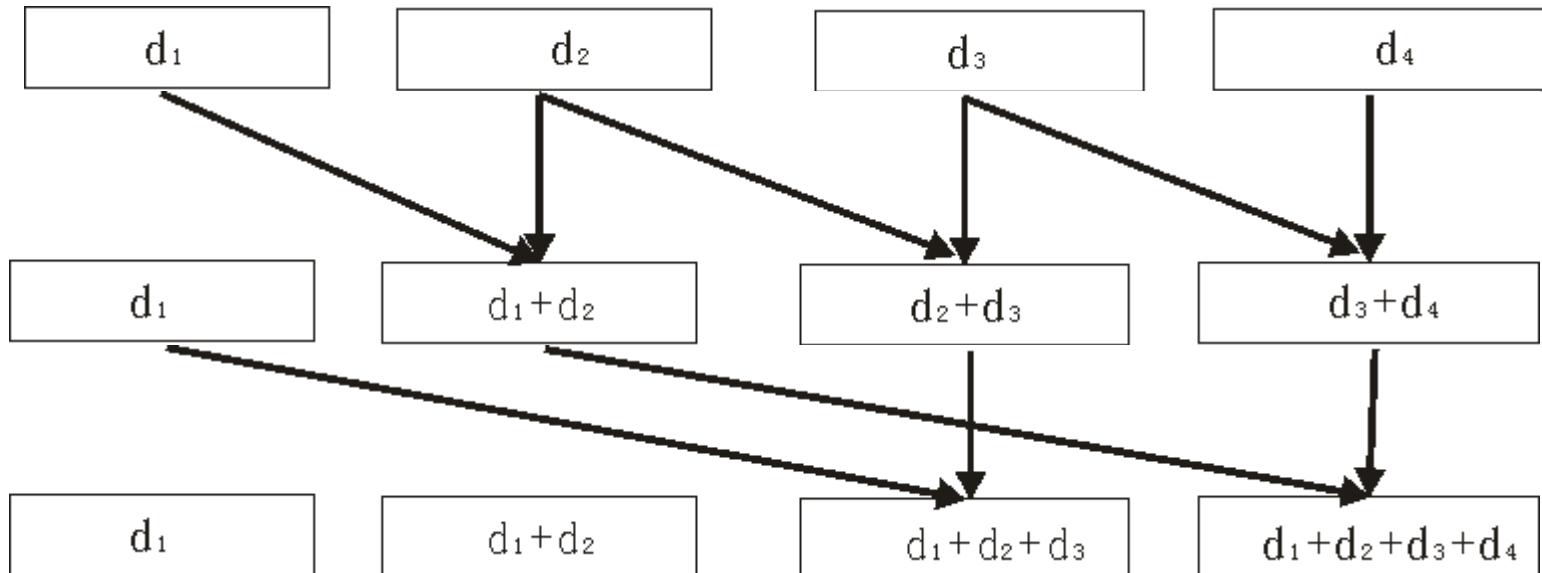
$$d'_2 = d_2 - k_1 d'_1$$

$$d'_3 = d_3 - k_2 d'_2 = d_3 + (-k_2)d_2 + (-k_2)(-k_1)d_1$$

$$d'_4 = d_4 + (-k_3)d_3 + (-k_3)(-k_2)d_2 + (-k_3)(-k_2)(-k_1)d_1$$

M

$$d'_n = d_n + (-k_{n-1})d_{n-1} + (-k_{n-1})(-k_{n-2})d_{n-2} + \dots + \prod_{i=1}^{n-1} (-k_i)d_1$$





DST算法求解块三对角方程：独立方程的求解

$$d'_1 = d_1$$

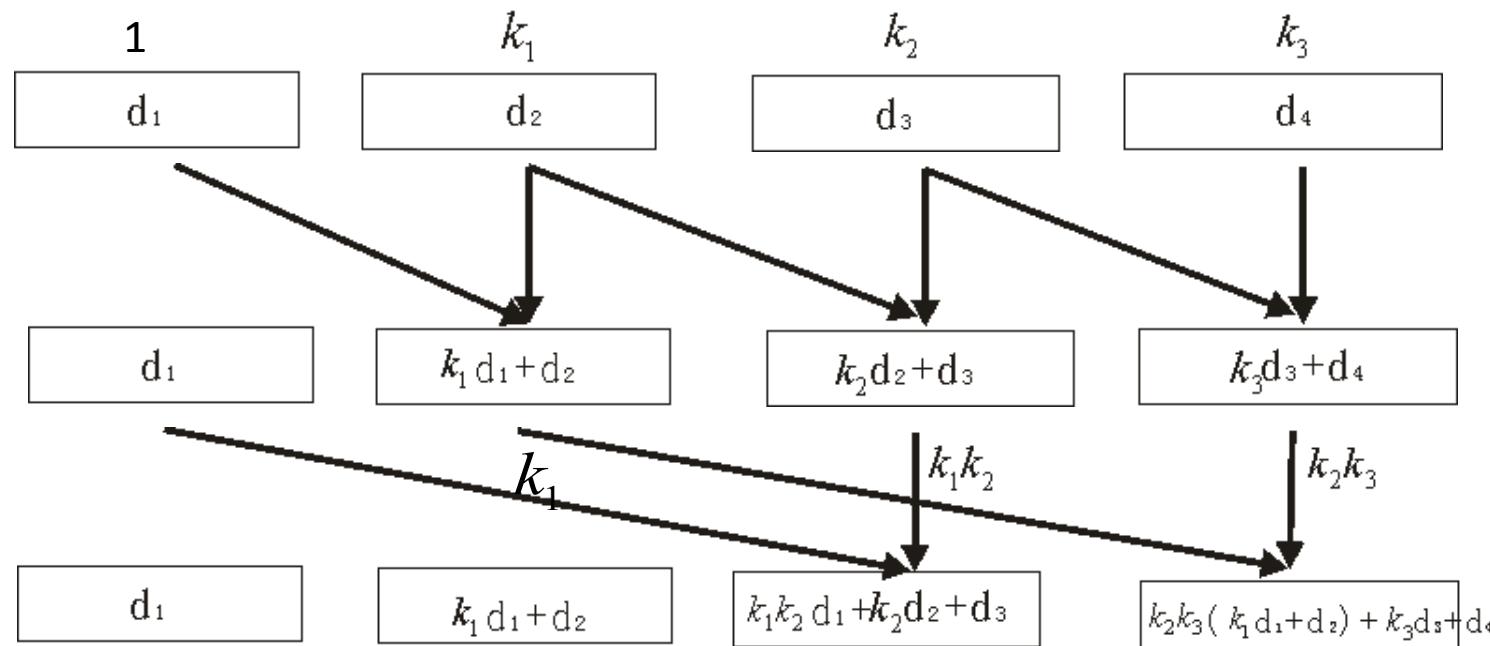
$$d'_2 = d_2 - k_1 d'_1$$

$$d'_3 = d_3 - k_2 d'_2 = d_3 + (-k_2) d_2 + (-k_2)(-k_1) d_1$$

$$d'_4 = d_4 + (-k_3) d_3 + (-k_3)(-k_2) d_2 + (-k_3)(-k_2)(-k_1) d_1$$

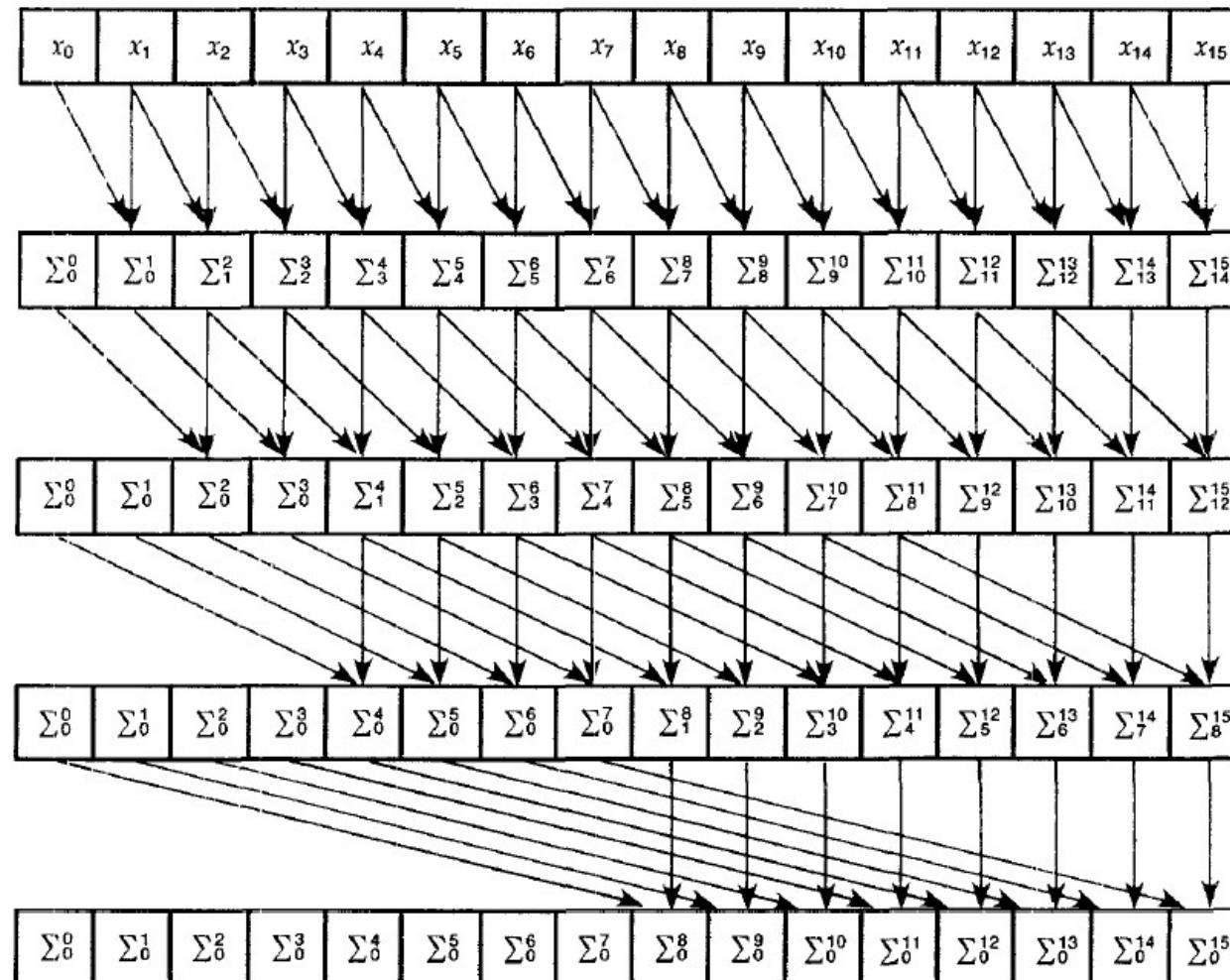
M

$$d'_n = d_n + (-k_{n-1}) d_{n-1} + (-k_{n-1})(-k_{n-2}) d_{n-2} + \dots + \prod_{i=1}^{n-1} (-k_i) d_1$$





DST算法求解块三对角方程：独立方程的求解





DST算法求解块三对角方程：独立方程的求解

for $j = 1$ **to** $\log_2 n$ **do**

for all m **in parallel do**

if $m \geq 2^j$ **then**

$$d[m] = k[m-1] \times d[m - 2^{j-1}] + d[m]$$

$$k[m-1] = k[m-1] \times k[m-1 - 2^{j-1}]$$

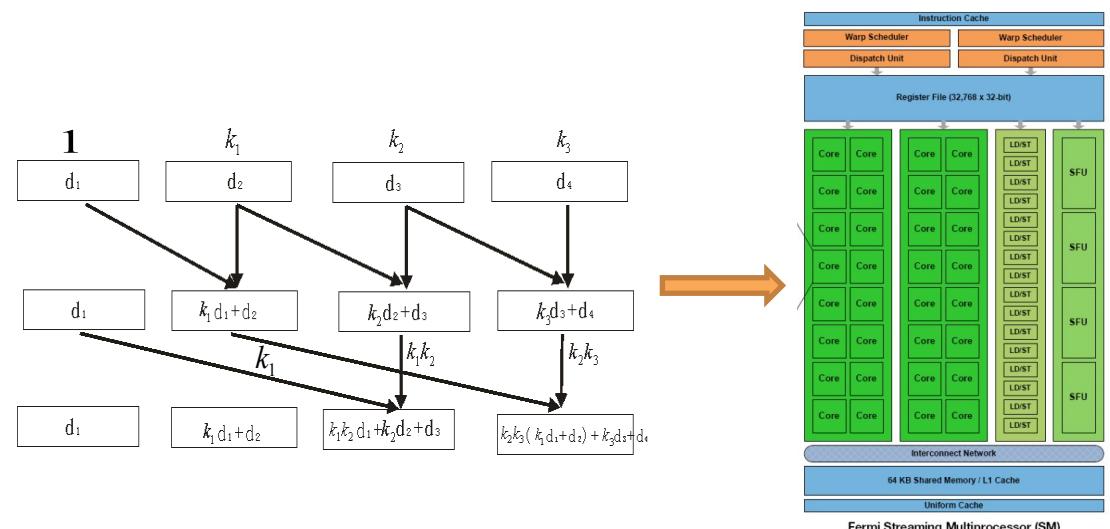
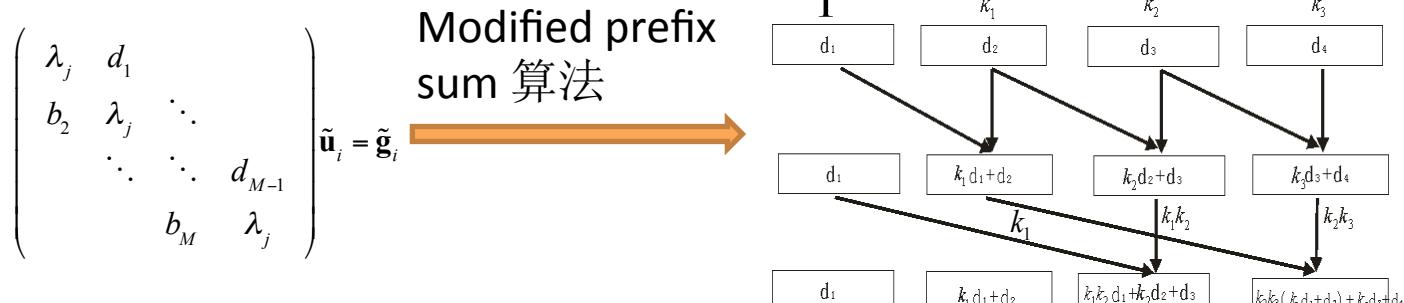
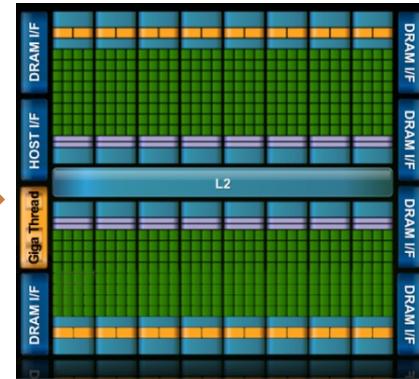
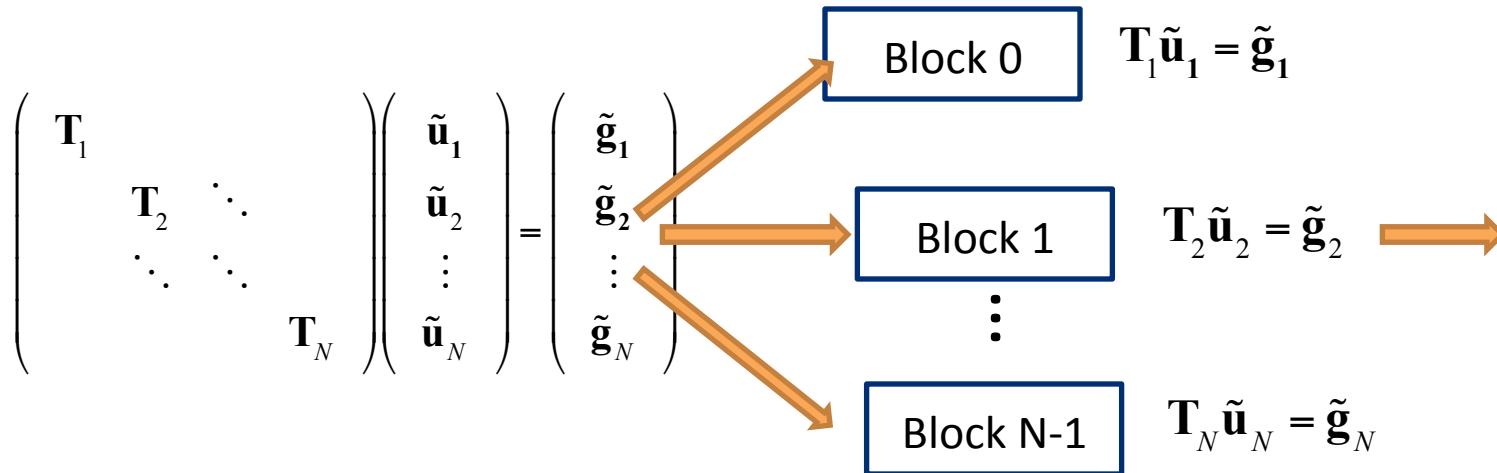
end if

end do

end do



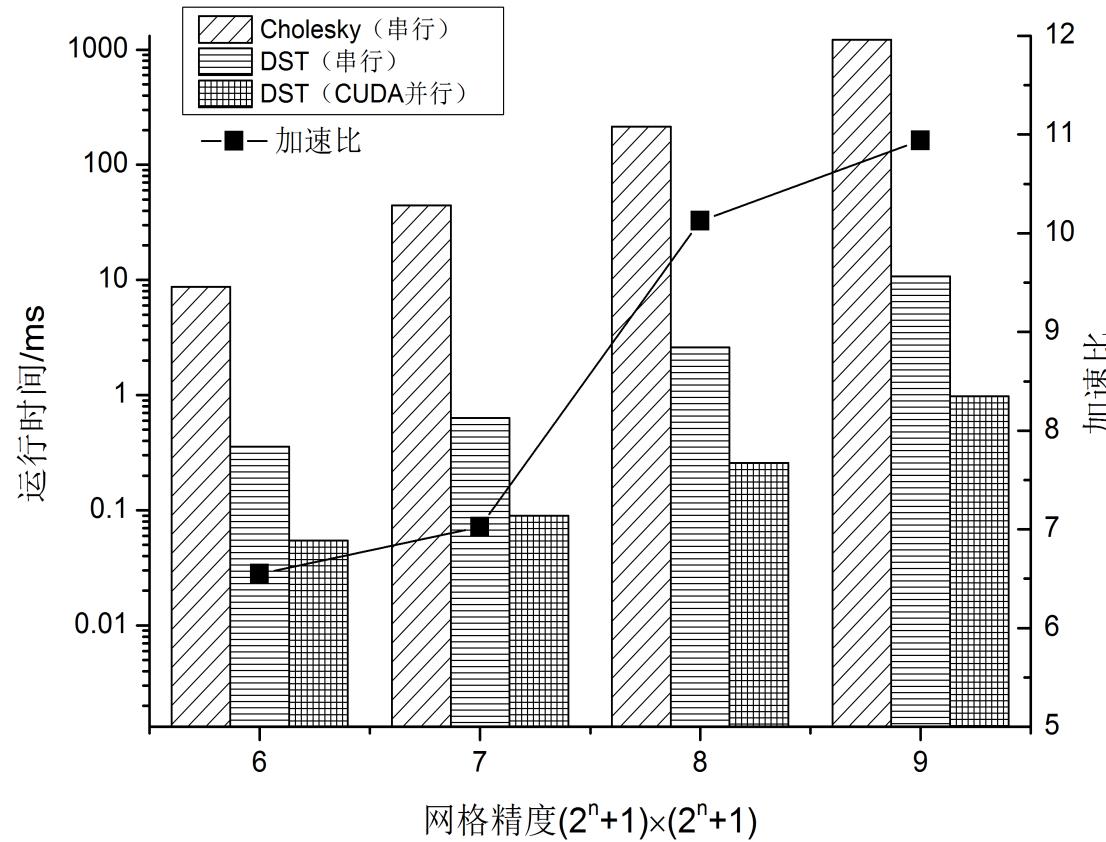
DST算法求解块三对角方程：独立方程的求解





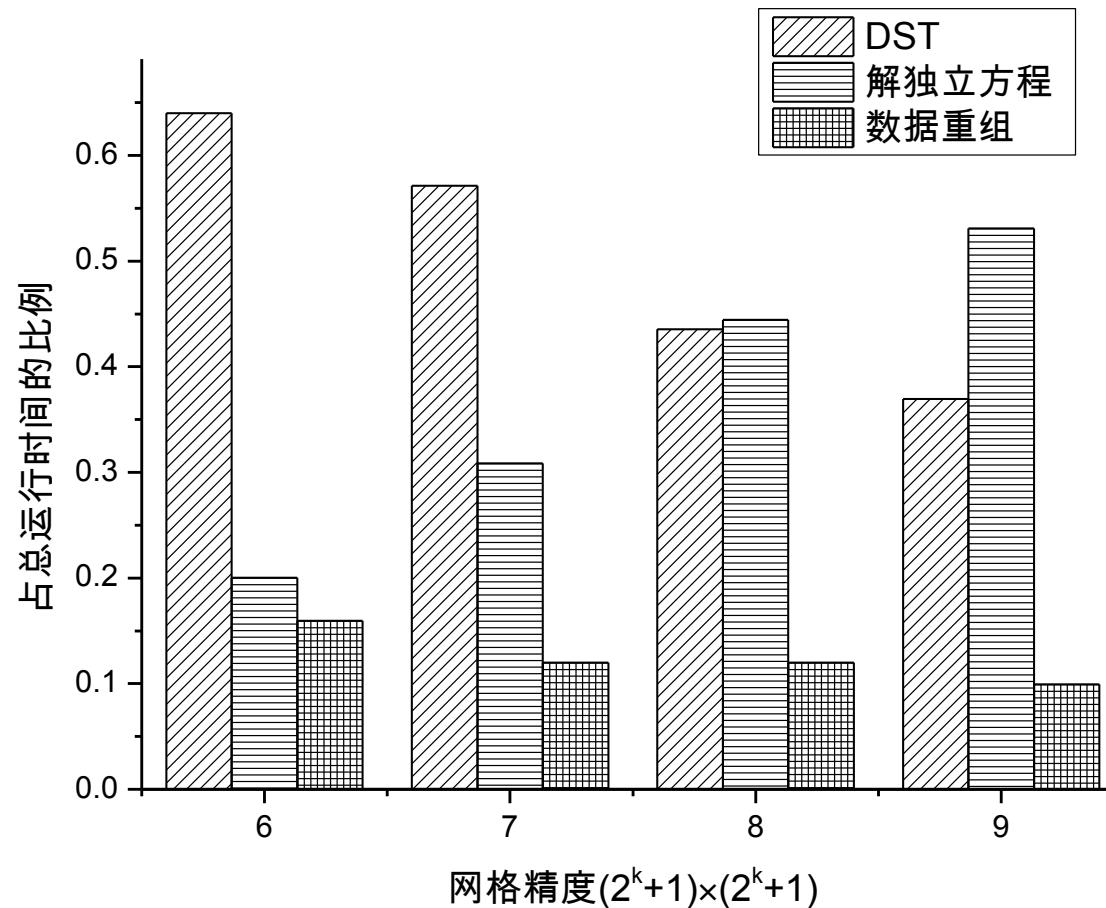
DST算法求解块三对角方程：测试结果

测试环境：GPU: Tesla c2050 CPU: Intel Xeon E3-1230 3.2GHz



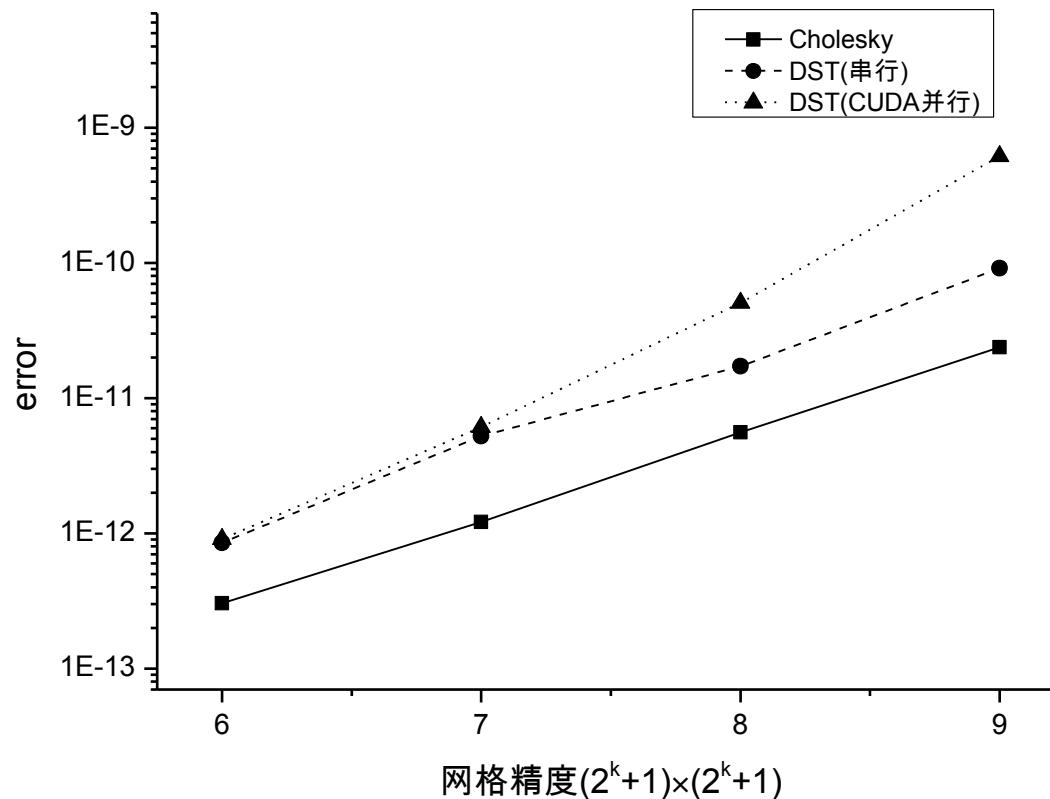


DST算法求解块三对角方程：测试结果





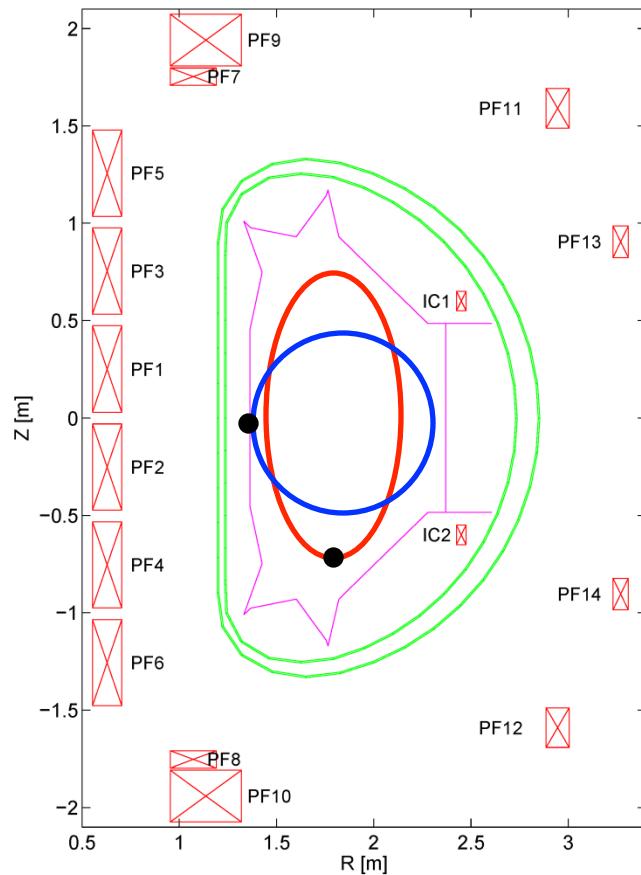
DST算法求解块三对角方程：测试结果





基于GPU的并行重建研究：边界磁通的确定

- 等离子体边界的定义：不与限制器相交的最外层闭合磁面。
- EFIT通过折半查找的来试探出满足这一条件的磁面以及磁面上的磁通
- 用作实时控制的程序，如XLOC, RT-EFIT一般采用另一种寻找方法。



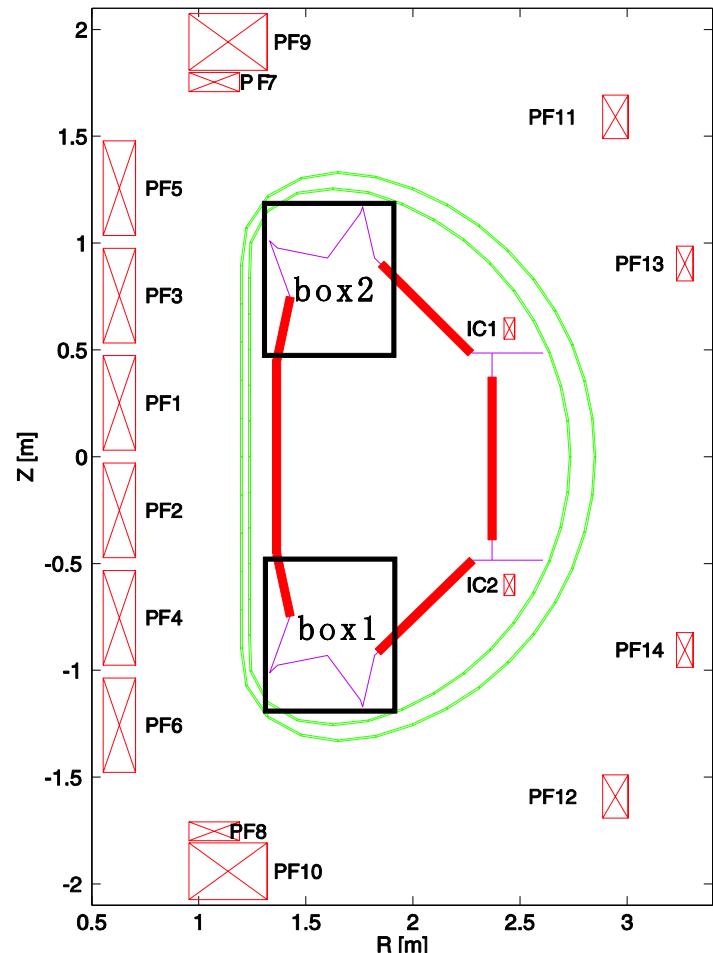


边界磁通的确定：X点寻找算法

- 利用 $B_R = -\frac{1}{R} \frac{\partial \psi}{\partial z}$ $B_z = -\frac{1}{R} \frac{\partial \psi}{\partial R}$ ，求出两个box中每个格点上的极向磁场值。
- 求解每个格点的 $B_R^2 + B_z^2$ ，利用并行锦标赛算法找出这一值最小的点。
- 以该点作为展开原点，做一阶泰勒展开，利用以下关系找出X点位置

$$\begin{cases} B_r^0 + \frac{dB_z}{dR} \delta r + \frac{dB_z}{dZ} \delta z = 0 \\ B_z^0 + \frac{dB_R}{dR} \delta r + \frac{dB_R}{dZ} \delta z = 0 \end{cases}$$

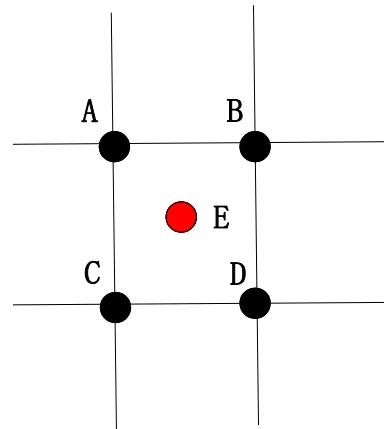
$$\begin{cases} \delta r = \frac{\left(B_r^0 \frac{dB_z}{dZ} - B_z^0 \frac{dB_R}{dZ} \right)}{\left(\frac{dB_z}{dR} \frac{dB_R}{dZ} - \frac{dB_R}{dR} \frac{dB_z}{dZ} \right)} \\ \delta z = \frac{\left(B_z^0 \frac{dB_R}{dR} - \frac{dB_z}{dR} B_r^0 \right)}{\left(\frac{dB_z}{dR} \frac{dB_R}{dZ} - \frac{dB_R}{dR} \frac{dB_z}{dZ} \right)} \end{cases}$$



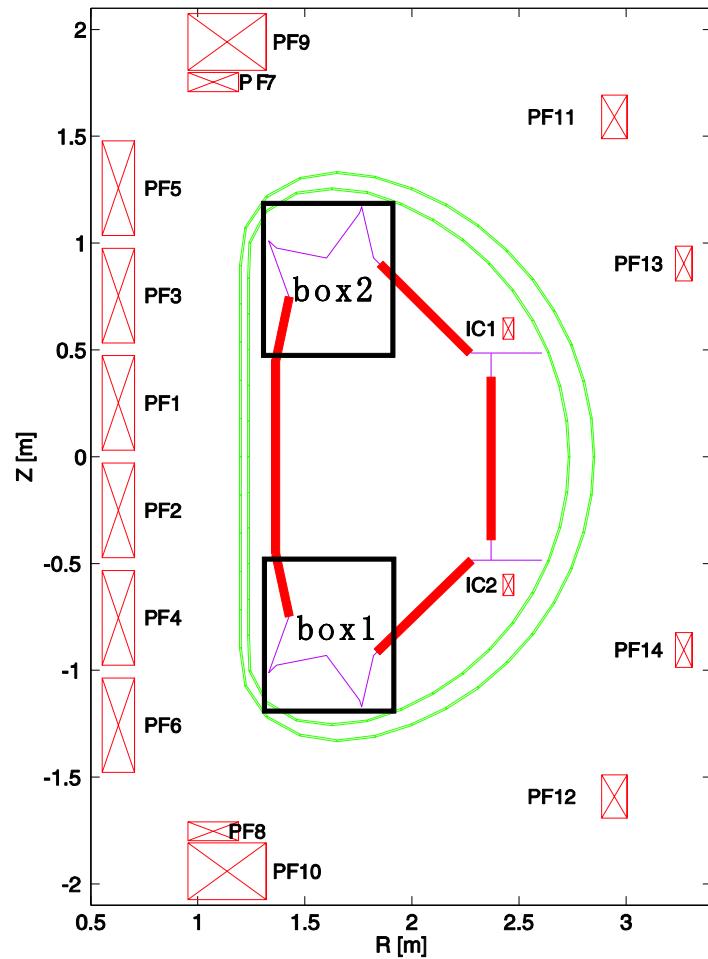


边界磁通的确定：限制器上的磁通值求解

- 放电之前，在限制器上选择若干个点。
- 在放点前预先计算好有限差分的系数。
- 在运行时根据周围四个点实时的磁通值插值得到限制器点上的磁通值。利用并行锦标赛算法找出磁通值最大的点。
- 注意Private flux region的影响。



$$\psi_E = a\psi_A + b\psi_B + c\psi_C + d\psi_D$$



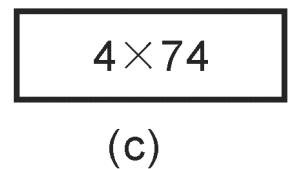
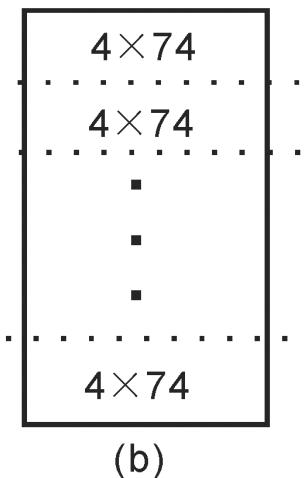
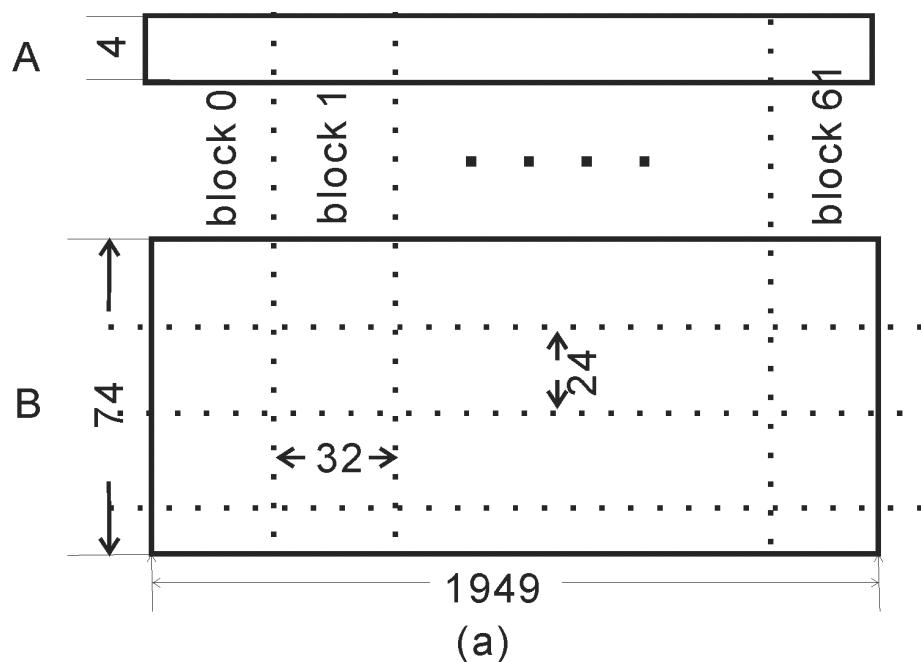


基于GPU的并行重建研究：响应矩阵的求解

$$\mathbf{G}_P \times \mathbf{H}$$

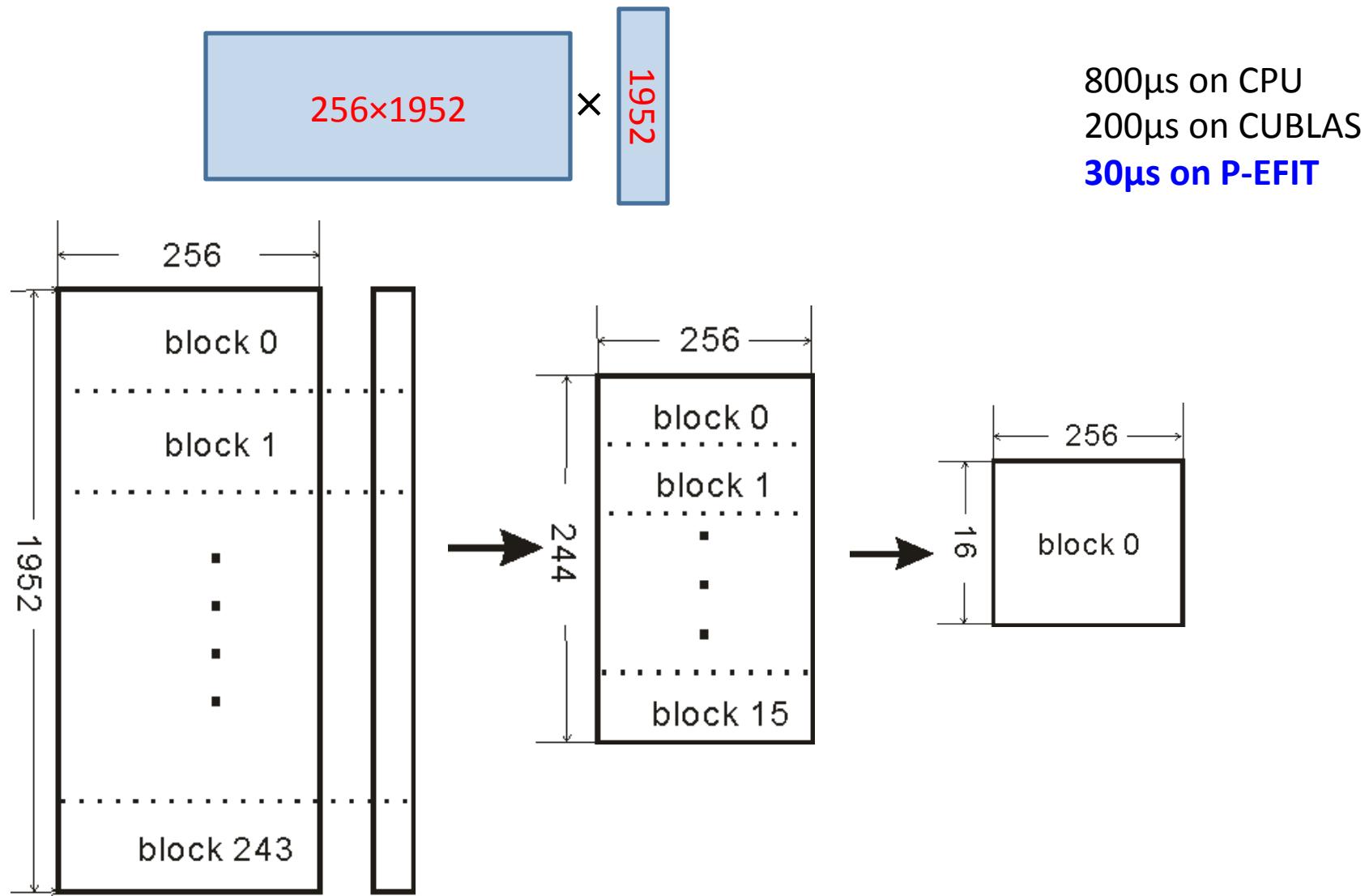
$$\begin{matrix} 74 \times 1952 \\ \times \\ 4 \times 1952 \end{matrix}$$

900μs on CPU
250μs on CUBLAS
40μs on P-EFIT





基于GPU的并行重建研究：边界磁通的求解





基于GPU的并行重建研究：最小二乘

- 未知量与诊断测量之间的关系为 $\mathbf{D} = \boldsymbol{\Gamma} \times \mathbf{U}$
- 为了增加最小二乘的精度，要增加合适的权重。

$$\mathbf{F} \cdot \mathbf{D} = (\mathbf{F} \cdot \boldsymbol{\Gamma}) \times \mathbf{U}$$

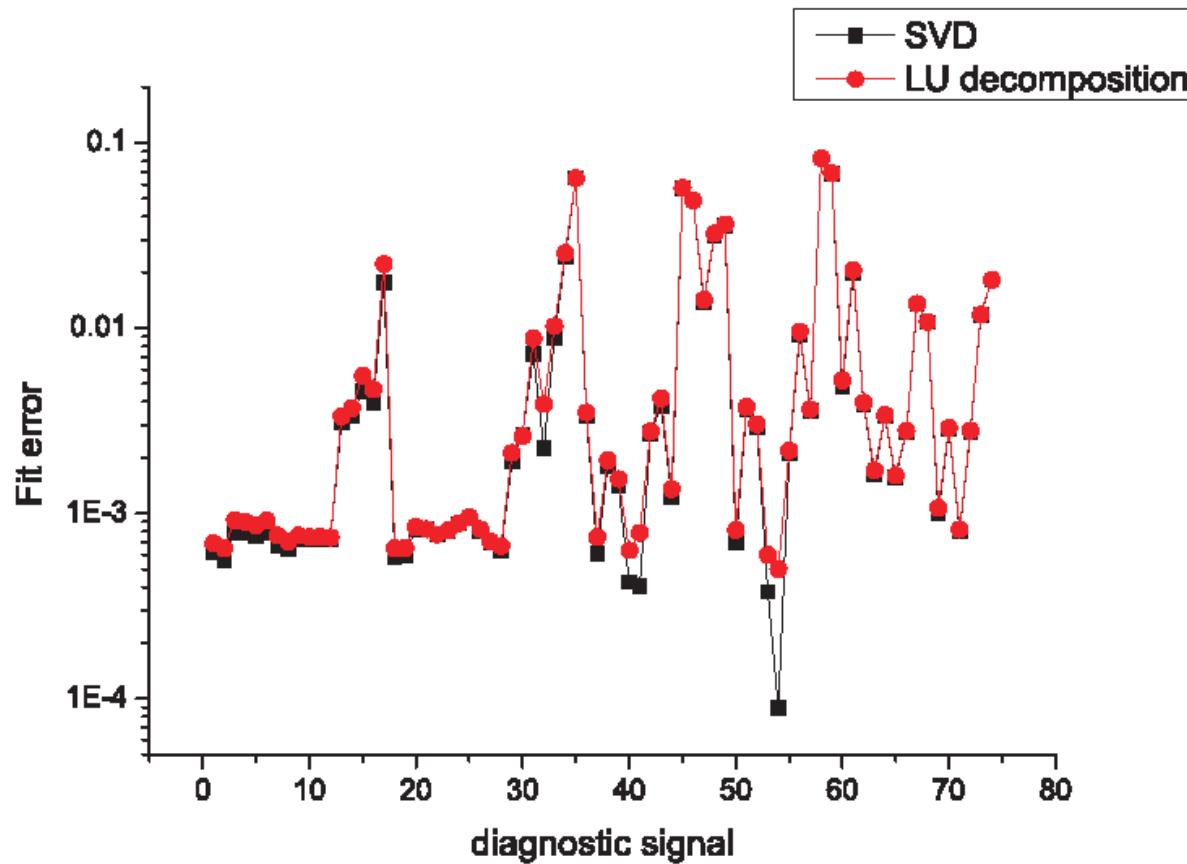
- 求解这一超定方程的方法包括SVD分解，QR分解以及直接法。
- P-EFIT采用的是Tikhonov 正则化的直接解法

$$(\boldsymbol{\Gamma}_w^T \boldsymbol{\Gamma}_w + \alpha^2 \mathbf{I}) \mathbf{U} = \boldsymbol{\Gamma}_w^T \mathbf{D}_w$$

- 目前的程序中正则化参数为 0。
- 原超定问题被转化为了一个很小维度的一般线性方程求解问题。
该问题可以在CPU上利用选pivoting-L-U分解算法完成。



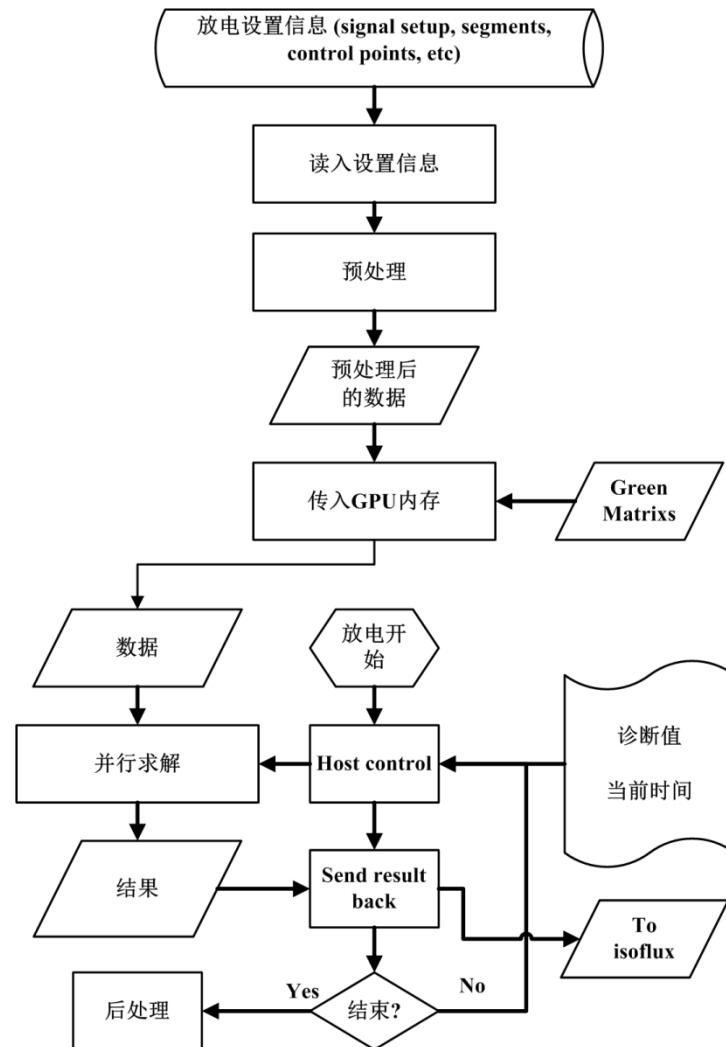
基于GPU的并行重建研究：最小二乘





基于GPU的并行重建研究：与PCS系统的整合

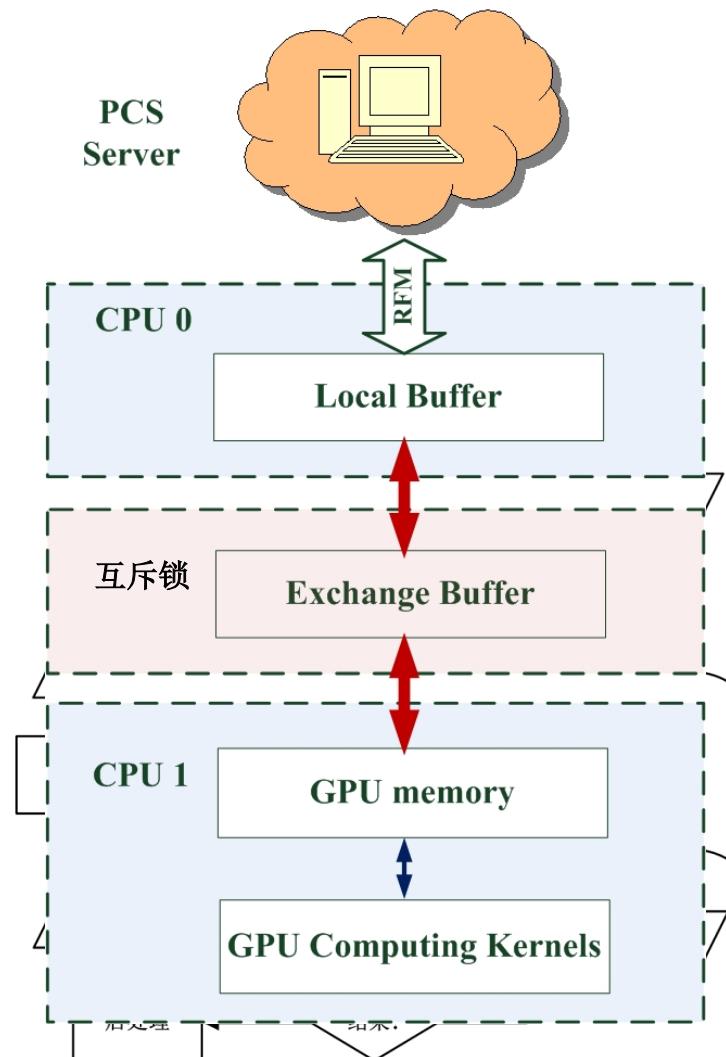
- 为了在将P-EFIT应用到实验中去，必须将P-EFIT与PCS系统整合起来。
- 这一整合需要PCS端和P-EFIT端的一系列工作。PCS端的工作由袁旗平老师和裴筱芳同学完成。这里主要介绍的是P-EFIT端的设计。
- P-EFIT端的主要工作流程为：





与PCS系统的整合：对P-EFIT性能的影响

- 与PCS系统整合后，CPU端增加了很多计算。主要包括对共享内存卡的读写，以及一些逻辑控制。
- 为此，P-EFIT定义了两个线程，通过硬亲和性绑定到两个CPU上。
- 其中一个是负责读取反射内存卡以及一些逻辑计算。另外一个负责控制GPU进行计算。
- 两个线程之间通过一个互斥锁控制的共享区域实现快速数据交换。



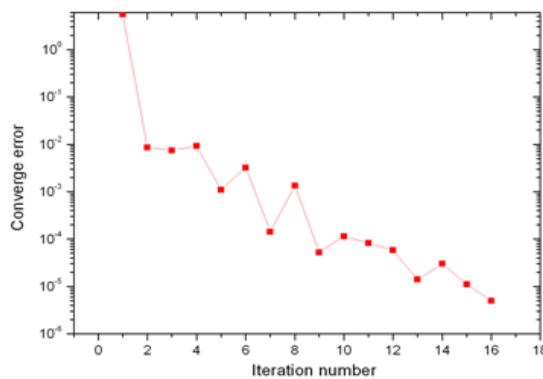


目录

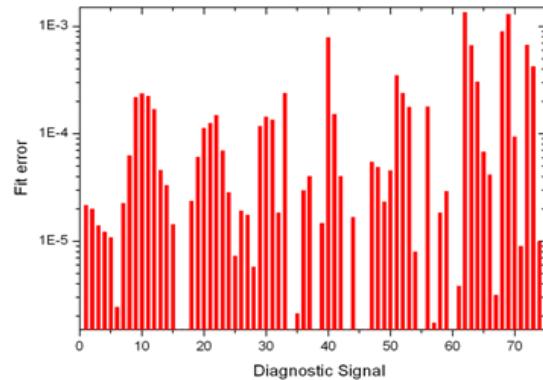
- 实时平衡重建简介
- 基于GPU的并行重建研究
- 测试结果与讨论
- 总结与展望



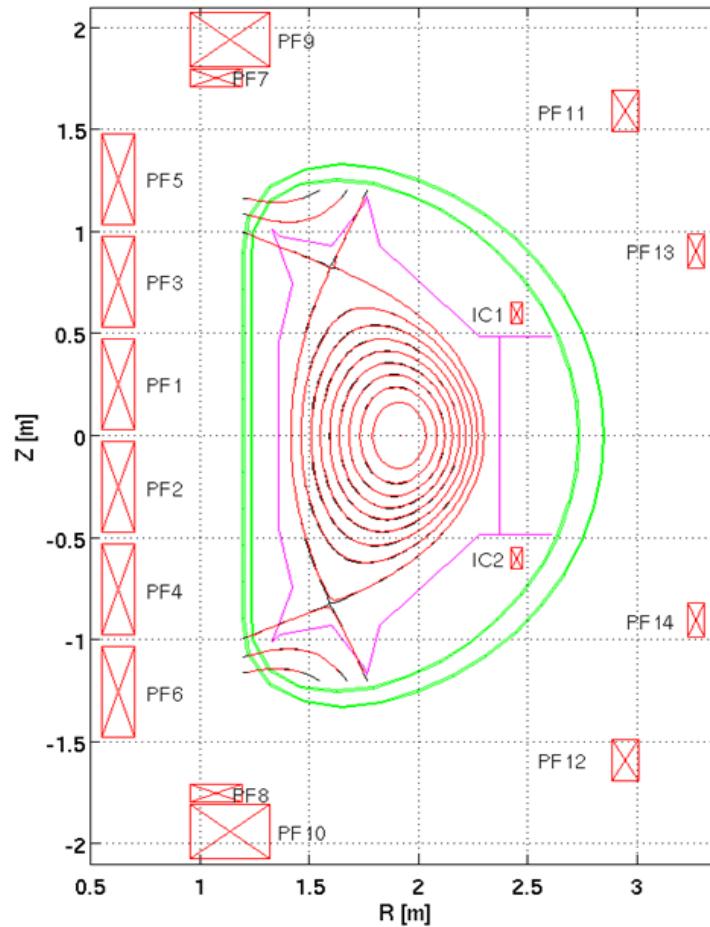
测试结果与讨论：正确性的验证



(a)



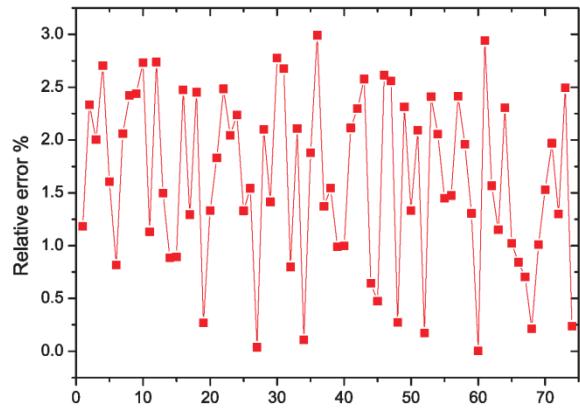
(b)



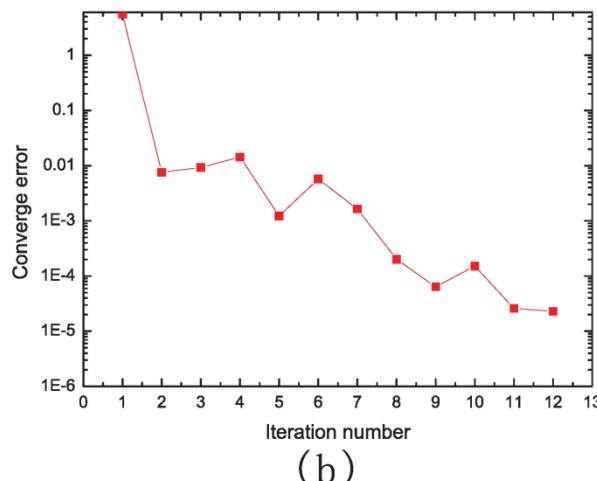
(c)



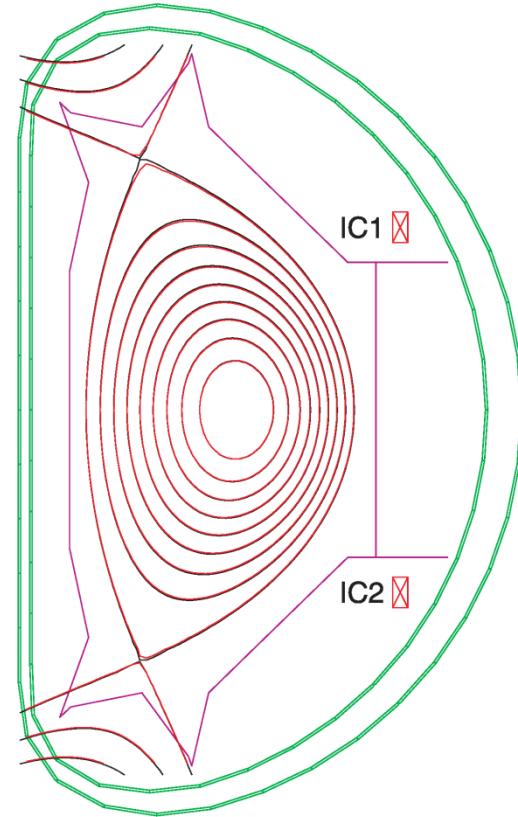
测试结果与讨论：正确性的验证（带误差的）



(a)



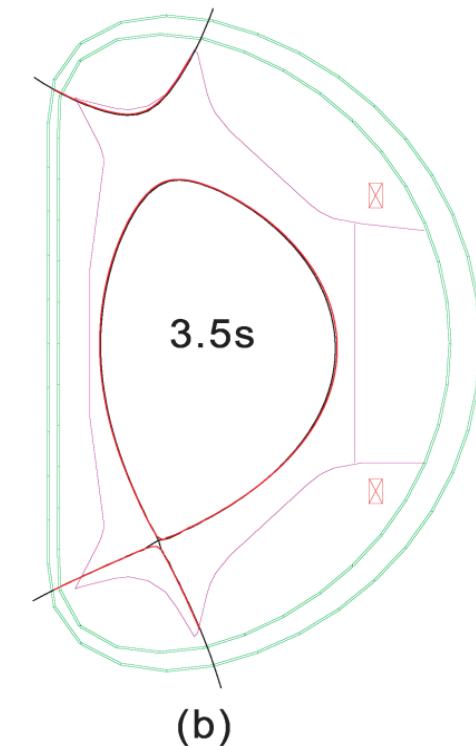
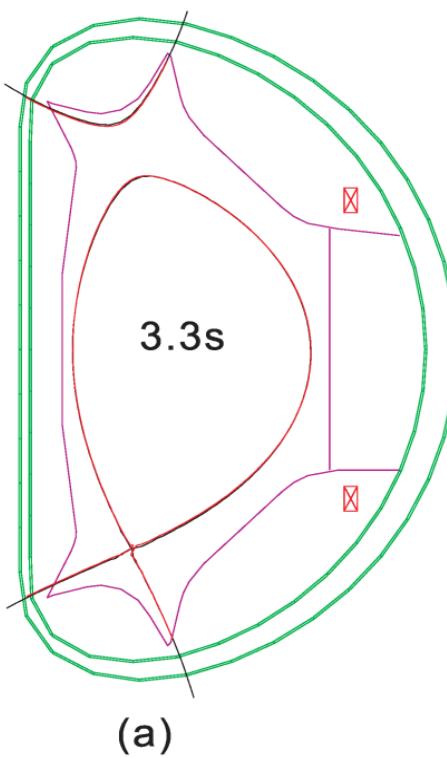
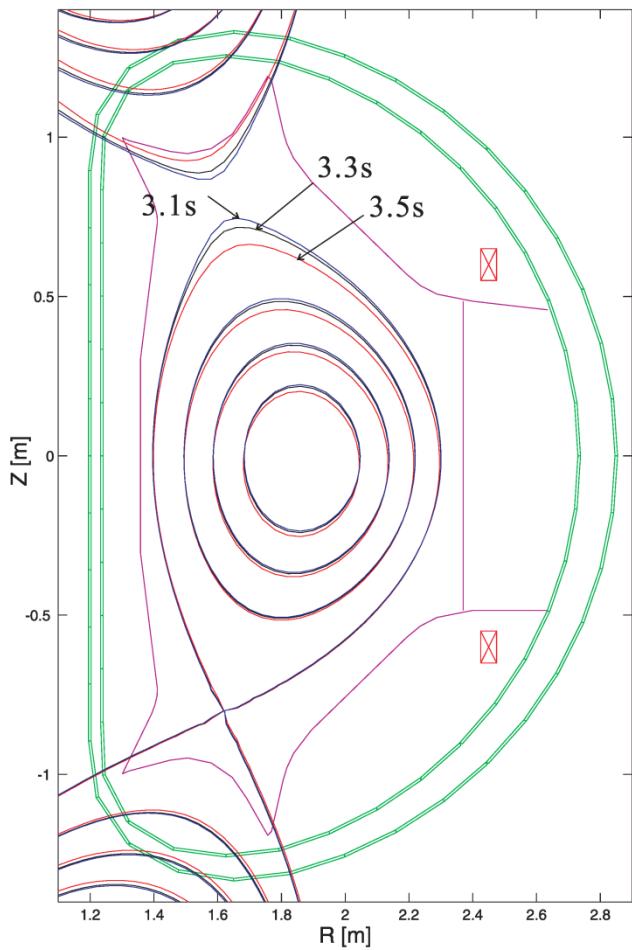
(b)



(c)

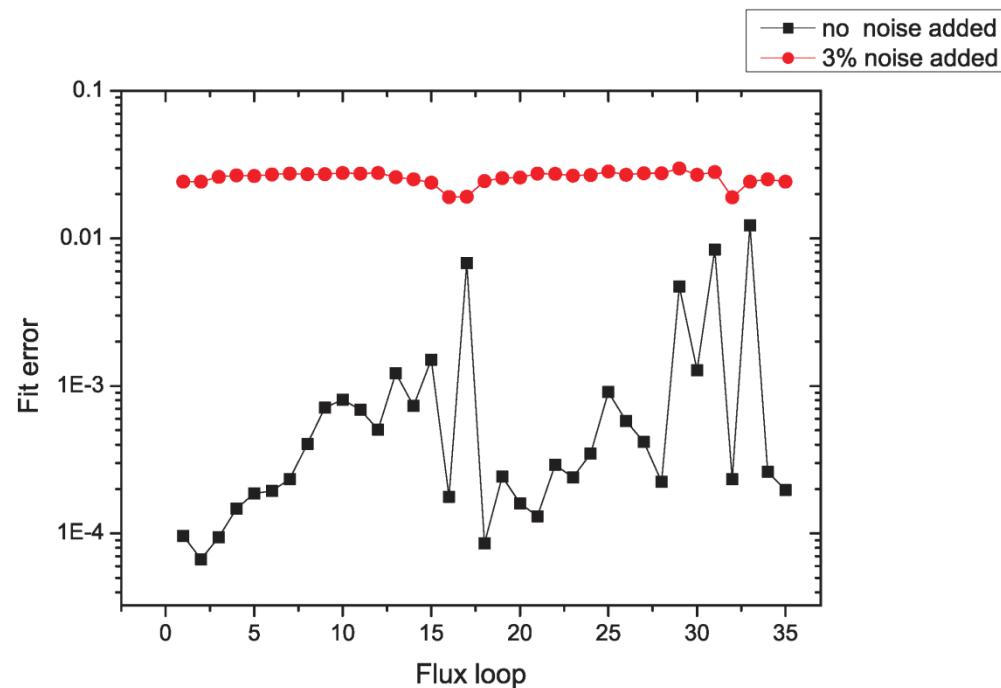
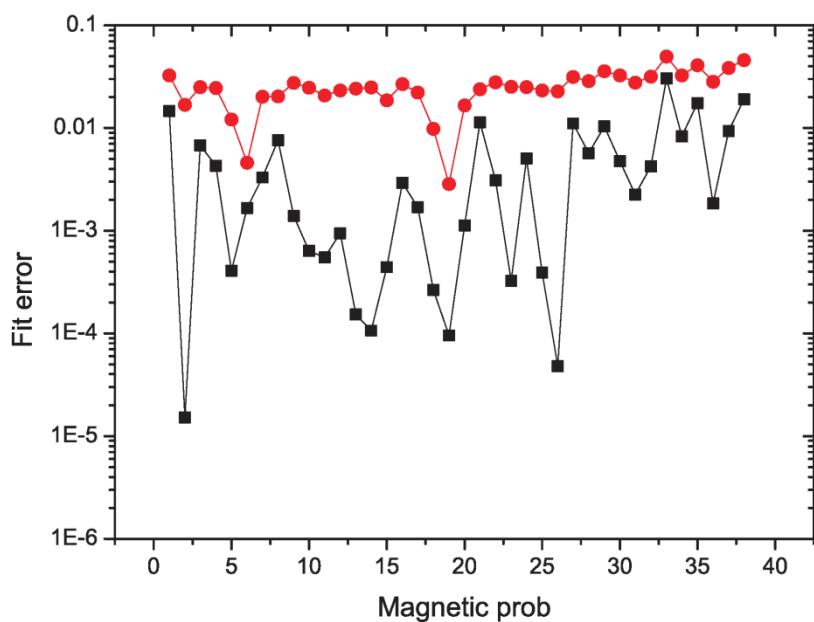


测试结果与讨论：一次迭代方法测试（带误差）



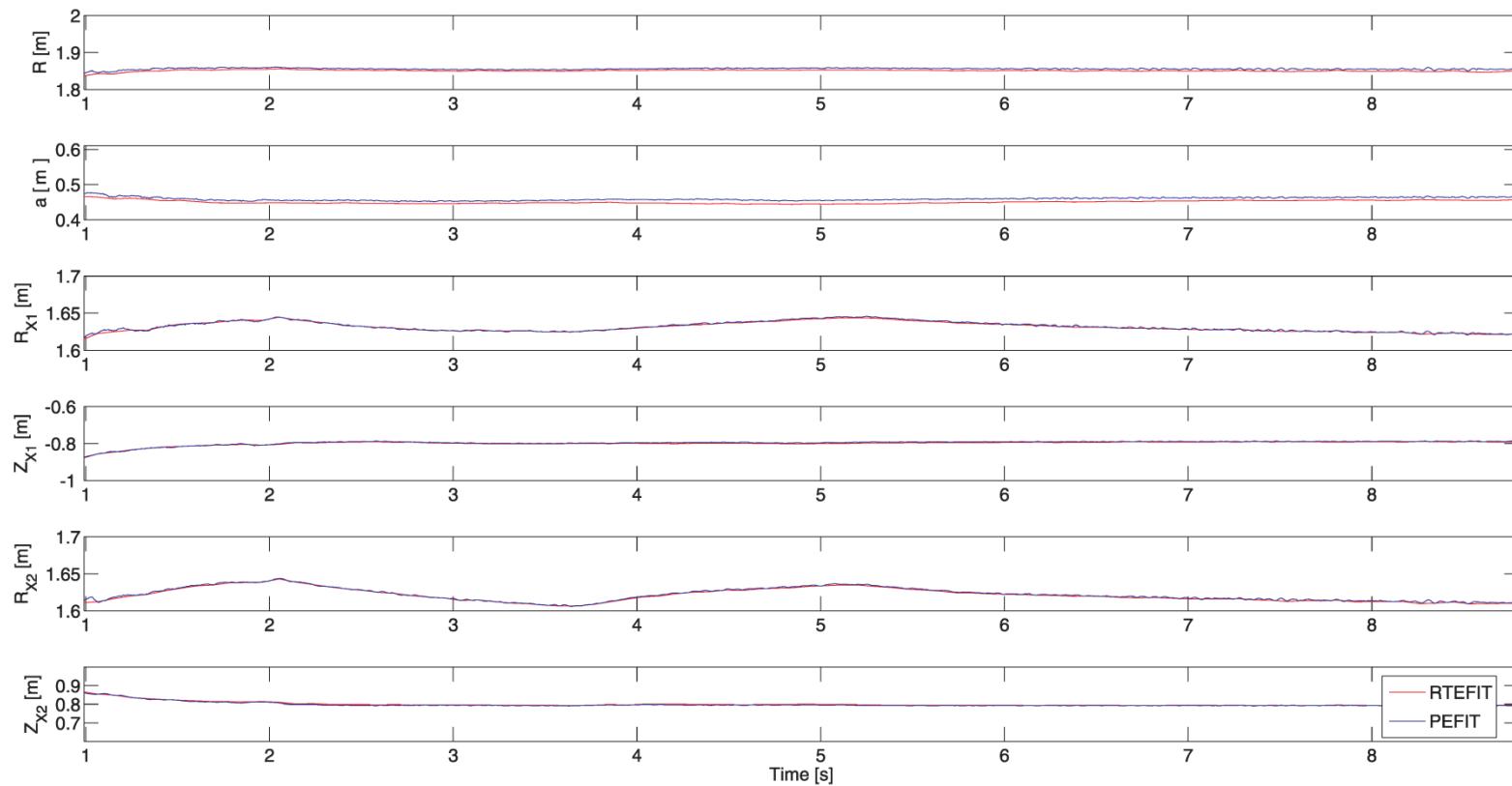


测试结果与讨论：一次迭代方法测试（带误差）





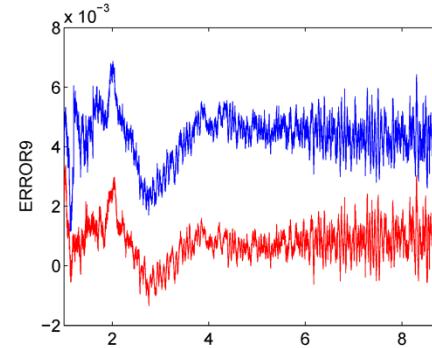
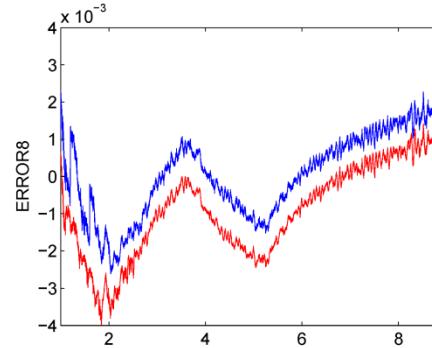
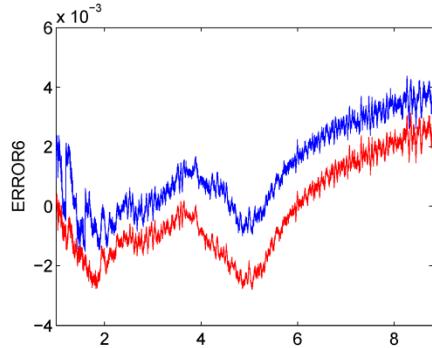
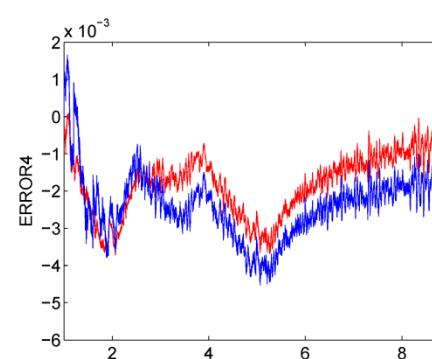
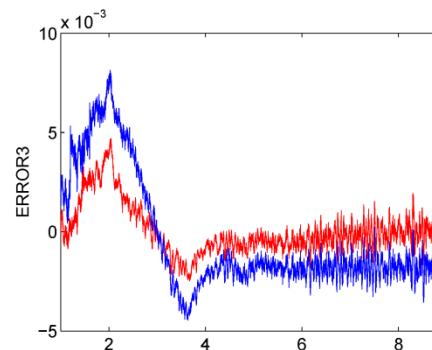
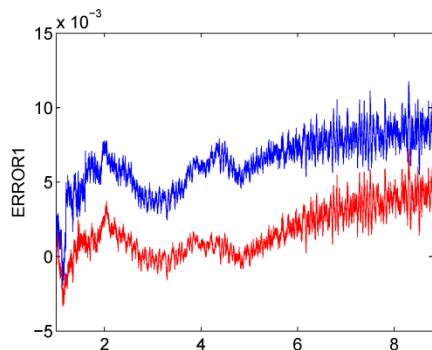
测试结果与讨论：实验环境下的模拟



EAST Shot 43362



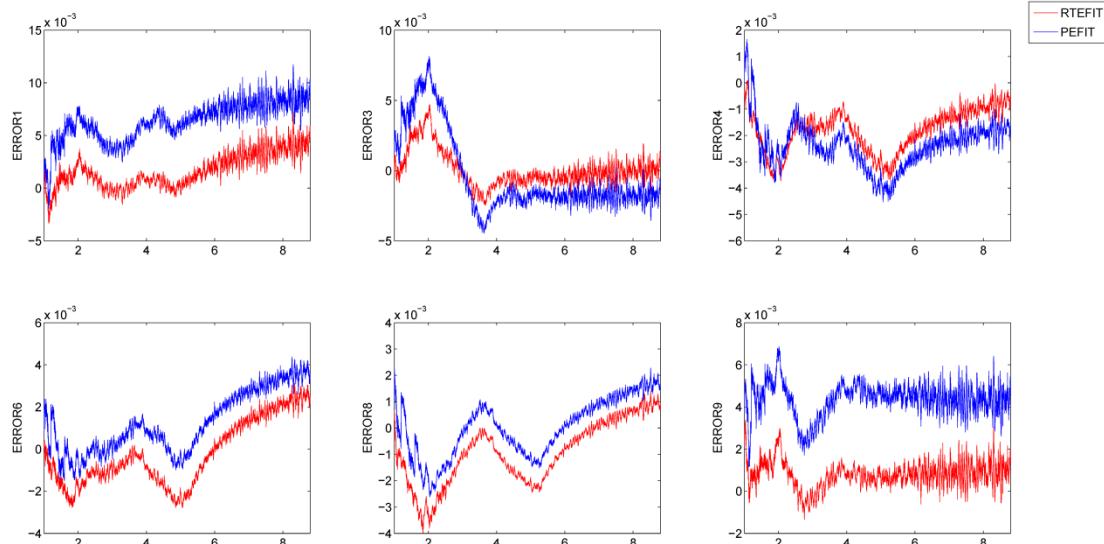
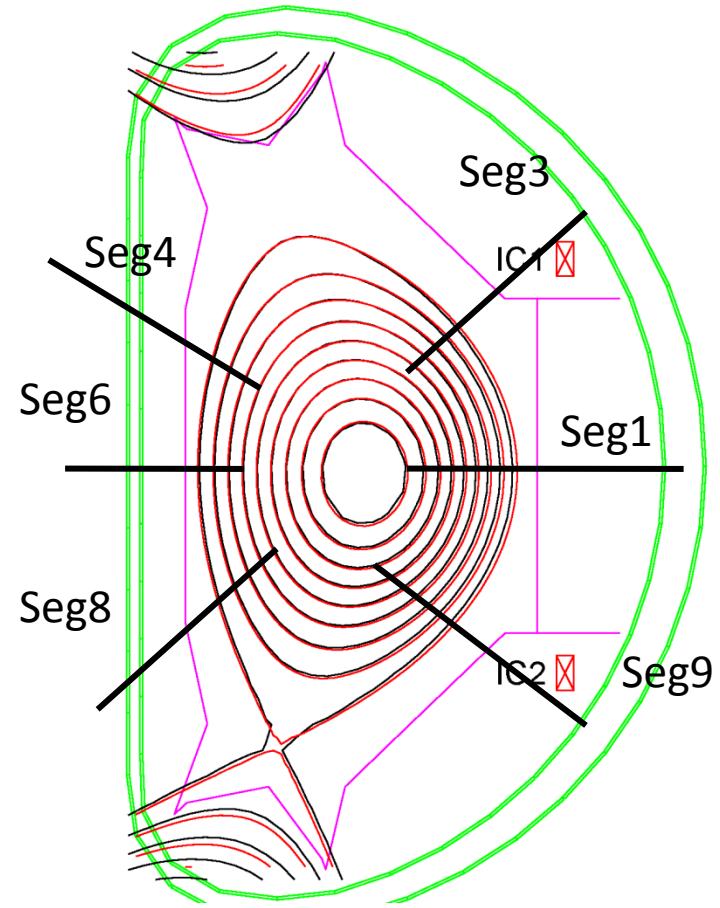
测试结果与讨论：实验环境下的模拟



RTEFIT
PEFIT



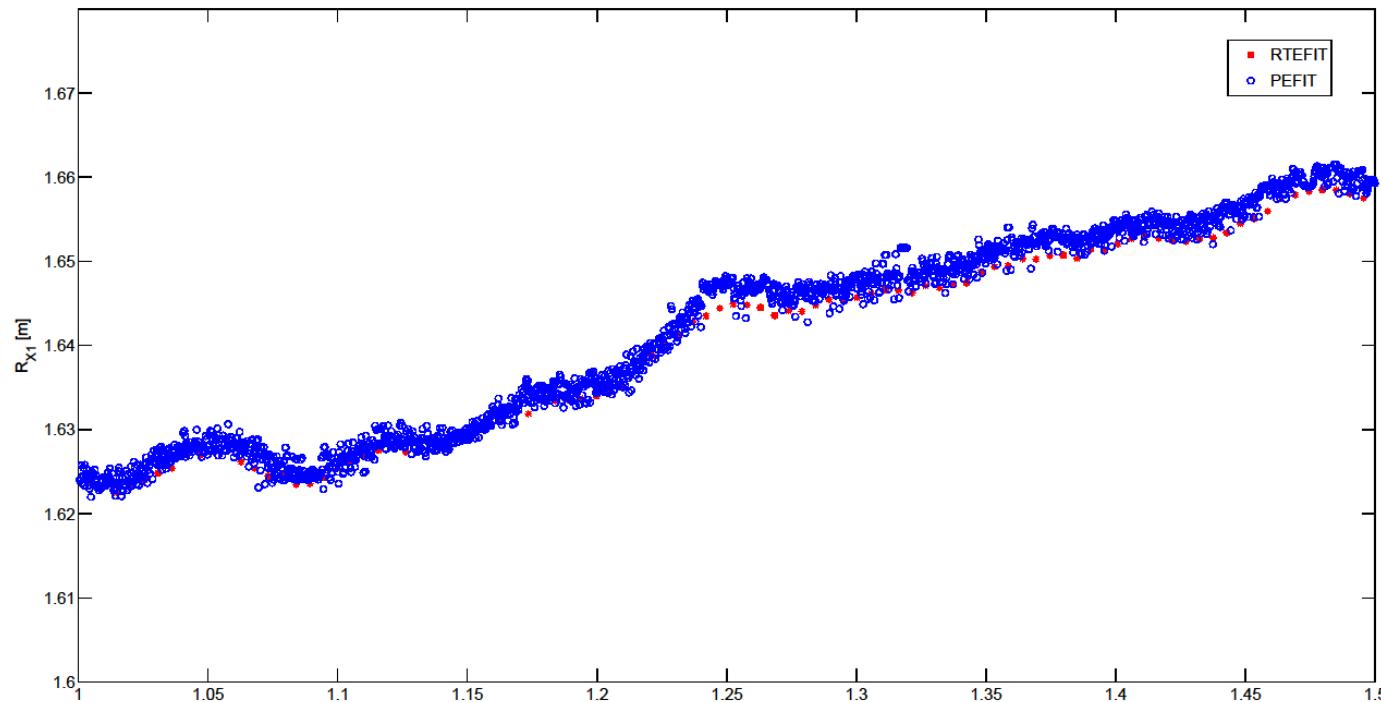
测试结果与讨论：实验环境下的模拟



沿着control segment的磁通梯度大约为 $4 \times 10^{-3}/\text{cm}$,
在seg1和seg9处两者位形存在大约1.3cm的误差,
从而使得此时输出的seg1和seg9的磁通error存在大
约 $5 \sim 6 \times 10^{-3}$ 的误差



测试结果与讨论：运行时间



EAST Shot 40403: 1.0s~1.5s,

RT-EFIT: 94 points 33×33 grid

PEFIT: 1985 points 65×65 grid



目录

- 实时平衡重建简介
- 基于GPU的并行重建研究
- 测试结果与讨论
- 总结与展望



总结与展望：总结

- 针对利用Picard迭代的直接平衡重建方法，分析了其算法的原理和特点。
- 开发了适用于Fermi架构GPU的块三对角方程求解算法。
- 针对 65×65 格点平衡反演过程中遇到的特定维数的矩阵相乘，开发了可以在GPU上有效加速的算法。
- 分析了绕开奇异值来进行最小二乘求解的可行性，开发了利用GPU和CPU协同的最小二乘算法。
- 开发了与PCS系统之间通信的模块，并尽量减少了其对算法运行速度的影响。
- 目前P-EFIT可以在 $220\mu\text{s}$ 左右完成一次 65×65 格点的重建迭代。
- 实验表明P-EFIT的精度满足实时平衡重建的要求，并且验证了P-EFIT应用在EAST实验中的可行性



总结与展望：展望

- P-EFIT的速度依然有提高的潜力。
 - 其中的一些算法依然可以继续优化。
 - 一些算法可以并发的执行。
 - 开发针对Kepler架构GPU的P-EFIT版本。
- 进一步提高P-EFIT的反演精度。
 - 改进X点找寻算法。
 - 进一步研究Tikhonov正则化参数对结果的影响。
- 增加P-EFIT的应用范围。
 - 增加P-EFIT的功能。
 - 开发 129×129 甚至更高格点精度的P-EFIT版本。
 - 增加P-EFIT的通用性。
 - 开发适用于超长脉冲放电的P-EFIT



硕士期间发表的学术论文

待发表论文：

- [1] Xiaoning Yue, Bingjia Xiao, Zhengping Luo Yong Guo, “Fast Plasma Equilibrium Reconstruction based on GPU parallel computing”, *Plasma Physics and Controlled Fusion* (录用)
- [2] 岳小宁, 肖炳甲, 罗正平; “基于CUDA的二维泊松方程快速直接求解”, 计算机科学 (录用)

会议论文：

- [1] Q.P. Yuan, X.N. Yue, X.F. Pei, Z.P. Luo, R.R. Zhang, B.J. Xiao, “The implementation of real time parallel equilibrium reconstruction in EAST PCS”, 9th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research, May, 2013.



致谢

感谢我的导师肖炳甲研究员，感谢您这三年来对我学习和工作上一丝不苟的指导，对我生活上无微不至的照顾。感谢您给我们创造出的这种自由的学术环境，使得我们每个人都可以毫无顾忌的探索自己感兴趣的方向。

感谢罗正平博士三年来对我的指导和帮助，感谢您耐心的传授我EFIT算法的知识，帮助我查找程序中的问题。您认真、努力的工作态度是我学习的榜样。

感谢袁旗平老师对我的帮助。感谢孙有文老师的指导。

感谢7室所有的老师和同学。



核科学技术学院

School of Nuclear Science & Technology

USTC

谢谢大家



核科学技术学院

School of Nuclear Science & Technology

USTC

欢迎提问



Algorithm Implementation in EAST PCS

- GPU will store those useful reconstruction results that are not needed during discharge in its own memory
- With 65×65 grid, if those results are recorded every 1ms. Then after about 50s, the GPU memory will be used out (Tesla c2050 have 3GB global memory).
- Have to transfer those data back during long pulse discharge
- So we are going to use the following method:

GPU 1



GPU 2

