*Hans Journal of Wireless Communications* 无线通信, 2013, 3, 71-76
http://dx.doi.org/10.12677/hjwc.2013.33011 Published Online June 2013 (http://www.hanspub.org/journal/hjwc.html)

# One-Side Jacobi Matrix Inversion Algorithm and DSP Realization

**Xi Yang, Zheng Li, Shuai Fang, Tian Zhou, Bin Jiang, Qin Guo**

Department of Information Science and Engineering, Southeast University, Nanjing
Email: seuyangxi@gmail.com, lizhengjustin@gmail.com, seufangshuai@gmail.com, seu.zhoutian@gmail.com, guoqinzju@gmail.com, binjiang@seu.edu.cn

Received: May 9th, 2013; revised: May 11th, 2013; accepted: May 20th, 2013



**Abstract:** Link adaptive transmission and advanced receiver are two key technologies in broadband wireless communication system. The design and realization of the system both involve a large number of matrix decomposition and inversion. The basic way to improve transmission efficiency of broadband wireless communication system is to enhance the efficiency of matrix decompose and inverse computations. For this purpose, this paper develops a kind of one-sided Jacobi algorithm based on classic Jacobi. Since this algorithm has the characteristic of parallelism, it can increase the efficiency at least twice in terms of the instruction execution cycle numbers. This article will first focus on the improved one-sided Jacobi algorithm as well as internal architecture and characteristics of DSP TMS320C6474. It then elaborates on how to implement this algorithm in parallel using TI's real-time multi-tasking operating system kernel (DSP/BIOS). Finally, this paper will compare instruction execution cycle numbers between parallel and serial algorithm under the same accuracy, proving the high efficiency of the improved one-sided Jacobi algorithm.

**Keywords:** One-Side Jacobi Algorithm; Matrix Inversion; TMS320C6474; DSP/BIOS; Parallel Realization

# 单侧 Jacobi 矩阵求逆算法及其 DSP 实现

**阳 析，李 峥，房 帅，周 天，江 彬，郭 骏**

东南大学信息科学与工程学院，南京
Email: seuyangxi@gmail.com, lizhengjustin@gmail.com, seufangshuai@gmail.com, seu.zhoutian@gmail.com, guoqinzju@gmail.com, binjiang@seu.edu.cn

收稿日期：2013 年 5 月 9 日；修回日期：2013 年 5 月 11 日；录用日期：2013 年 5 月 20 日

**摘 要：**链路自适应与先进接收机是宽带无线通信系统的核心技术，其设计与实现均涉及大量的矩阵分解以及矩阵求逆运算，提高矩阵分解和矩阵求逆运算的效率是提高宽带无线通信系统传输效能的基本途径。针对此目的，本文提出一种在经典 Jacobi 算法上改进的单侧 Jacobi 算法。由于该算法具有并行的特性，相比于串行（单核）实现在指令执行周期数上可提高至少两倍的运行效率。本文首先重点介绍改进的单侧 Jacobi 算法和 TMS320C6474 DSP 的内部架构与特性，然后重点阐述结合 TI 的实时多任务操作系统内核(DSP/BIOS)并行实现此算法，最后在同样精度的计算结果下比较并行算法与串行算法指令执行周期数，由此验证改进的单侧 Jacobi 算法在并行实现上的高效性。

**关键词：**单侧 Jacobi 算法；矩阵求逆；TMS320C6474；DSP/BIOS；并行

## 1. 引言

自 21 世纪以来，移动通信飞速发展，尤其是最

近几年，移动通信作为一种便捷、快速的通信方式融入人们的生活而备受青睐。然而人们对实时数据业务需求的增长给移动通信的数据速率提出了更高的要求，高数据速率需求的增加促使 3GPP 在 2004 年底启动了长期演进计划(Long Term Evolution, LTE)。3GPP LTE 的主要目标包括显著提高终端用户的吞吐量和扇区容量、降低用户延迟、显著提高用户移动性、降低网络功耗和用户设备复杂度等。为了达到技术指标，LTE 引入了许多关键技术，MIMO 技术便为其中之一。MIMO，即多输入多输出，它是利用多个天线实现多发多收，在不需要增加频谱资源和天线发送功率的情况下，成倍地提高信道容量，从而有效提升无线通信系统的频谱效率，提高传输速率并改善通信质量。但是如何从收集到的信号中滤除噪声，还原出信号源发射出的真实信号，则是一个关键问题，MIMO 的信道可以简单地抽象成一个矩阵，因此要解决这个关键问题就必然涉及到大量的矩阵求逆。由于 MIMO 技术能提供的信道容量和天线数是成正比的，所以矩阵的维度也随天线数的增加而增加，此时传统的算法逐渐表现出极低的效率、极差的数值稳定性以及极高的实现成本，所以能否开发出更为高效的矩阵求逆算法，成为整套系统中迫切需要解决的核心问题。

Jacobi 算法因其良好的并行性[1-3]，一直是矩阵求逆算法研究的热点。Jacobi 方法的思想就是通过不断的正交相似变换，使矩阵的非对角线元素逐步趋于 0，从而得到对角阵，此时对角上的元素就是矩阵的特征值，由每一次的旋转矩阵 J 累乘可以得到特征向量矩阵。然而，传统的双边 Jacobi 算法，每次迭代都同时涉及行和列的更新，行和列之间存在很大的数据相关性，这种数据相关性在算法的并行实现时将带来很大的核间通信代价，降低算法效率。

本文首先提出一种改进的单侧 Jacobi 矩阵求逆算法。紧接着，在简要介绍 TMS320C6474 DSP 的内部架构与特性后将重点阐述如何基于 DSP/BIOS 并行实现此算法。最后，本文将分别给出在同样计算结果精度的情况下，并行算法与串行算法的指令执行周期数，以证明此改进的单侧 Jacobi 算法在并行实现上的高效。

## 2. 改进的单侧 Jacobi 算法

单侧 Jacobi 算法由于每次迭代只涉及列的数据更新，数据相关性大为降低，只需较少的核间通信便能并行实现，因此各处理器之间可以更加独立的工作，算法效率也将更高。本小节将主要介绍单侧 Jacobi 算法以及对其进行的改进。

以下为改进的单侧 Jacobi 算法的具体步骤：

1) 旋转矩阵的求取：

设实对称矩阵 $\boldsymbol{A}=\boldsymbol{A}_0$，逐次选择修正矩阵 $\boldsymbol{R}_k$，使 $\boldsymbol{A}_k$ 的 $i,j$ 两列正交化[4,5]。为减少乘法次数，改进的单侧 Jacobi 算法的修正矩阵 $\boldsymbol{R}_k$ 将采用快速 Givens 矩阵的形式：

$$\boldsymbol{R}_k(i,j,\alpha)=\begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & \alpha & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\alpha & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix} \quad (1)$$

$$\begin{matrix} & & i & & j & & \end{matrix}$$

式(1)为快速 Givens 矩阵的形式，主要作用为将矩阵中元素的值集中到对角线元素上，以利于后续矩阵求逆的计算，映射到以列为矢量元素的矢量图中相当于对矩阵中的列矢量进行旋转。其中，参数 $\alpha$ 计算如下：

设

$$I=\left\langle a_i^{(k)},a_i^{(k)}\right\rangle, J=\left\langle a_j^{(k)},a_j^{(k)}\right\rangle, e=\left\langle a_i^{(k)},a_j^{(k)}\right\rangle,$$

这里 $\langle x,y\rangle$ 表示向量 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 的内积，若 $e=0$，则说明两列已经正交，取 $\alpha=0$，否则记

$$t=\frac{J-I}{2e},\quad \alpha=-t+\operatorname{sgn}(t)\sqrt{t^2+1} \quad (2)$$

$\alpha$ 这样取值可以保证相应的旋转角满足 $|\theta|\le\pi/4$，以提高算法的数值稳定性。注意到参数 $\alpha=-t+\operatorname{sgn}(t)\sqrt{t^2+1}$，这里涉及到开方运算，复杂度较高，考虑进行一定的近似，从而降低复杂度。注意到 $\alpha$ 关于 $t$ 是奇对称的，所以本文仅讨论 $t>0$ 的情形即

$$\alpha=-t+\sqrt{t^2+1} \qquad (t>0) \quad (3)$$

利用微积分的知识以及 Taylor 展开式，易得：

$$\lim_{t\to\infty}\sqrt{t^2+1}-t=\lim_{t\to\infty}\frac{1}{\sqrt{t^2+1}+t}=\lim_{t\to\infty}\frac{1}{2t} \quad (4)$$

$$\lim_{t\to 0}\sqrt{t^2+1}-t \approx \lim_{t\to 0}\left(1+\frac{t^2}{2}\right)-t \tag{5}$$

所以，在实际计算时可以作出如下近似(分段点的选取是随本文要求的精度变化而变化的，此处本文假设控制误差在$10^{-4}$数量级内，得出如下的近似结果)：

$$\tilde{\alpha}=\begin{cases}1-t+\dfrac{t^2}{2} & 0<t\le 0.25\\ -t+\sqrt{t^2+1} & 0.25<t\le 8\\ \dfrac{1}{2t} & 8<t\le 1024\\ 0 & 1024<t\end{cases} \tag{6}$$

即在实际计算时，采用式(6)的近似式代替式(3)，可以避免开方运算，有效降低运算复杂度，提高运算速度。

2) 得到快速 Givens 矩阵 $\boldsymbol{R}_k$ 后，即可对目的矩阵 $\boldsymbol{A}_k$ 进行旋转变换：

$$A_{k+1}=\boldsymbol{A}_k\boldsymbol{R}_k\ (k=0,1,\cdots) \tag{7}$$

注意到在每次旋转变换时都需要计算第 $i,j$ 列的模值，即 I，J 的值，常规的单侧 Jacobi 算法采用直接求取列向量模值的方法，共需要 4n 个 flop，n 为矩阵维数。受到文献[6,7]的启发，本文提出的改进的单侧 Jacobi 算法将采取一种效率更高的列向量模值计算的方法，如下：

从表达式中易知$\begin{cases}a_i^{(k+1)}=a_i^{(k)}-\alpha a_j^{(k)}\\ a_j^{(k+1)}=\alpha a_i^{(k)}+a_j^{(k)}\end{cases}$，所以

$$\begin{cases}I^{(k+1)}=\left\langle a_i^{(k+1)},a_i^{(k+1)}\right\rangle=\left(a_i^{(k)}-\alpha a_j^{(k)}\right)^T\left(a_i^{(k)}-\alpha a_j^{(k)}\right)\\ \quad=I^{(k)}-2\alpha e^{(k)}+\alpha^2 J^{(k)}\\ J^{(k+1)}=\left\langle a_j^{(k+1)},a_j^{(k+1)}\right\rangle=\left(\alpha a_i^{(k)}+a_j^{(k)}\right)^T\left(\alpha a_i^{(k)}+a_j^{(k)}\right)\\ \quad=\alpha^2 I^{(k)}+2\alpha e^{(k)}+J^{(k)}\end{cases} \tag{8}$$

由此，可以在每次旋转变换结束前对列的模值按照式(1-8)进行更新，这样只需要 12 个 flop，仅为原来的 3/$n$，在 $n$ 较大的情况下，效率可以得到比较大的提升。

3) 反复进行上述两步，选择著名的 Robin Ring Ordering 作为列的正交化顺序，则 $\boldsymbol{A}_k$ 的各列趋向于两两正交，即 $\boldsymbol{A}_k$ 趋向于 $\boldsymbol{Q}$，而$\boldsymbol{V}=\boldsymbol{R}_0\boldsymbol{R}_1\boldsymbol{R}_2\,\boldsymbol{L}\,\boldsymbol{R}_k$，即有 $\boldsymbol{AV}=\boldsymbol{Q}$，即 $A=QV^{-1}$。

矩阵求逆：将 $\boldsymbol{Q}$ 矩阵各列化为单位向量，得 $\boldsymbol{Q}=\boldsymbol{Q}_0\boldsymbol{\Lambda}$，$\boldsymbol{Q}_0$ 为单位正交阵，$\boldsymbol{\Lambda}$ 为对角阵，所以 $\boldsymbol{A}=\boldsymbol{Q}_0\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$，则

$$\boldsymbol{A}^{-1}=\boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{Q}_0^{-1}=\boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{Q}_0^{T} \tag{9}$$

此即完成了矩阵求逆。综上所述，改进的单侧 Jacobi 算法主要在三个方面提高了算法效率：第一，采用快速 Givens 旋转矩阵作为修正矩阵，节省了一半的乘法运算量；第二，采用极限和泰勒展开式近似，减少复杂的开方运算；第三，在计算列向量的模值时采用迭代的方法，对列向量模值进行更新。

## 3. TMS320C6474 的特性和 DSP/BIOS

本文将主要采用多核 DSP TMS320C6474 来并行实现上一节所提出的改进的单侧 Jacobi 算法，因此本小节将根据算法具体实现时所涉及到的 DSP 具体部件简要介绍 TMS320C6474 DSP 的内部架构与特性。

由上一节算法的具体实施步骤可知，对于高维矩阵的求逆，单侧 Jacobi 算法的计算量主要集中于对矩阵的列的排序和每两列之间的数据处理，这就要求 DSP 有较高的执行能力、足够的存储空间以及较快的存储器访问速度。如图 1(C6474 内部结构简略图，主要示出算法的 DSP 实现中所需模块)所示，TMS320C6474 包括三个 TMS320C64x + TM DSP 核，总主频可达 3.6 GHz，性能高达 28800MIPS (Million Instructions

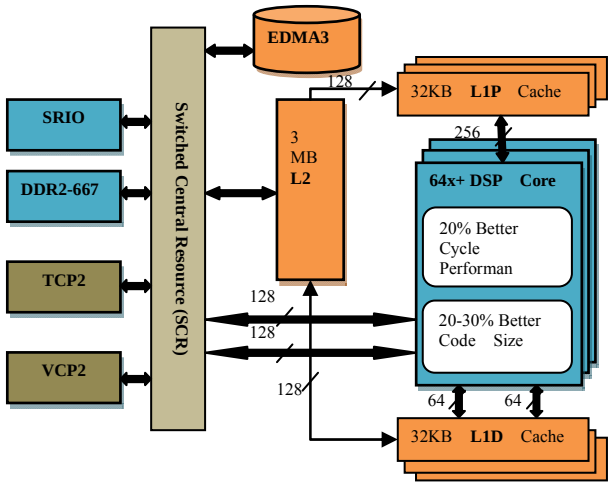



**Figure 1. Internal architecture of TMS320C6474**
**图 1. TMS320C6474 的内部架构**

Per Second)且 3 个核之间相对独立，每个核可以执行不同的程序。片上共有 8 个功能单元，每个时钟周期执行一条指令，最高情况下可同时执行八条指令；此外 C6474 DSP 还集成了大量片上存储器，称之为三级存储系统：第一级为 Cache，包括 32KB 的一级程序存储器(L1P)和 32KB 的一级数据存储器(L1D)；第二级(L2)存储器中三个核共享 3MB 的存储空间，有两种不同的配置方式，速度比第一级慢，但是容量大；第三级存储器为通过 DDR 接口扩展外部 DDR 存储器。这些特性为单侧 Jacobi 算法的并行实现提供了必要的物理基础。

由于算法是并行实现，这就要求同时建立多个线程，并对多线程之间的执行顺序进行正确的调度，此外还需对有限的资源进行合理分配。DSP/BIOS 是 TI 公司为其 TMS320C6000TM 系列 DSP 平台所设计开发的一个尺寸可裁剪的实时多任务操作系统内核，它主要由三部分组成：多线程实时内核(抢占式多线程)，实时分析工具和芯片支持库。它以模块化的方式提供给用户对线程(TSK)、中断(HWI，SWI)、定时器(CLK)、内存资源(MEM)、所有外设资源的管理能力，这些资源都可以根据需要剪裁。

## 4. 单侧 Jacobi 算法的 DSP 实现

论文在前面两小节详细讲述单侧 Jacobi 算法以及 DSP TMS320C6474 的内部架构与主要特性后，本小节将重点阐述如何结合 TI 的实时多任务操作系统内核(DSP/BIOS)在 C6474 上并行实现此改进的单侧 Jacobi 算法。

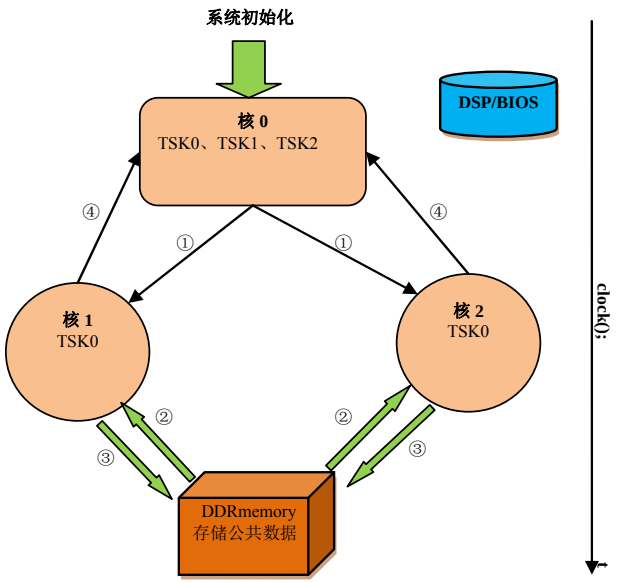
在 DSP 中主要有两种并行处理模式：Master/Slave 模式和 Data Flow 模式。

Master/Slave 模式：表现为由一个处理器中央控制多个处理器共同完成，中央控制处理器需要调度任务的执行线程和相关资源，并将线程分配给空闲的处理器来完成；此种模式主要适用于少量数据运算，大量任务调度的应用程序，一般和相应的操作系统相结合，它的缺点是控制处理器之间通信和调度的代码量较大，导致整个程序所需存储空间大，它难点则在于怎样保证在线程很多情况下的实时性。

Data Flow 模式：当一个处理器对数据完成相应的操作后，将数据传递给下一个处理器以便对数据进行进一步的处理；此模式适合于需要大量的复杂的数据运算的程序，它的难点在于怎样将程序分解成合适的处理流程以便实现流水线操作和较高的数据吞吐率。

考虑到算法的特点，本文最终采用结合了 Master/Slave 模式和 Data Flow 模式二者特点的混合模式。算法实现中，将要被处理的数据存放在 DSP 的 DDR2 memory 中，主核中建立有多个线程(Task)，由各自的信号量(semaphore)控制执行(running)和暂停(blocking)；主核(核 0)通过 IPC module 控制其它两个副核(核 1、核 2)对不同数据同时进行操作，两个副核完成相应操作后反馈信息给主核，主核则根据信息判断目前所处的状态并进行下一步的操作的部署直至算法完成后在 message_log 中输出所得结果。同时程序利用 clock()；函数对算法的执行时间进行了较精确的统计，以用于算法之间效率的比较。

如图 2 所示，首先在 DDR2 memory 中开辟出一块共属于三个核的公共数据的存储区域。然后由主核(核 0)控制系统的初始化工作，包括初始化 DDR2 memory，载入需要处理的相关数据，启动 clock()；函数等。一切准备工作就绪后，主核通过核间通信模块(IPC module)通知另外两个副核开始进行数据处理，并行的单侧 Jacobi 算法开始执行。




① 主核发送控制信号；
② 从 memory 中读取数据；
③ 将数据写到 memory 中；
④ 副核反馈信息给主核；

**Figure 2. DSP implementation schematic**
**图 2. DSP 具体实现示意图**

3 个核的详细功能为：

**核 0：**主核，控制整个算法的执行、计算整个算法执行所需的 cycle 数以及输出最后的结果。在 main(); 函数使能 IPC 中断后执行 TSK0，TSK0 的主要功能为提示整个算法开始执行，并抛出信号量给 TSK1 进而真正开启整个算法。TSK1 则是控制中心，在对所要处理的矩阵的列号进行排序后，通过一定的算法每次将特定的两列的序号分别发送给其它两个副核以进行进一步的操作，遍历完后，对矩阵的列号再一次排序并发送，以此类推直至下一次循环即完成一轮，经过几轮迭代后整个算法执行完毕，激活 TSK2，TSK2 负责对最后所得结果进行一定的处理以输出最后的结果。

**核 1 和核 2：**两个副核的操作除所处理的数据不同外，其它均相同。这两个核是算法执行的主体，算法的实现由这两个核共同完成。副核的主要操作均在 TSK0 中，由信号量 SEM0 进行控制。TSK0 的主要任务为：根据接收到的主核发来的序号对相应列进行 Givens 旋转变换，以使最后得到的矩阵各列两两正交，操作完成后反馈主核。

## 5. 串/并行效率比较

本小节将在上一节的基础上比较在同样计算结果精度的情况下，并行算法与串行算法所需的指令执行周期数，以证明此改进的单侧 Jacobi 算法在并行实现上的高效。

串行实现是指仅使用 C6474 的一个核执行该算法，并行实现则是指采用上一节所述的思想由 C6474 的三个核相互合作共同执行该算法。用 clock();函数分别对算法的完成所需的 cycle 数进行

统计，并以 cycle 数与结果的精度为主要指标对串行与并行的效率进行比较。

在此本文以 8 × 8 维的浮点数矩阵为例，所处理的浮点数矩阵如下图 3：

根据算法的特性，当增加迭代的次数时结果的精度会不断提高，直到迭代的次数大于某一界限时(该界限与所处理的矩阵的维数有关，矩阵维数越大，所需迭代次数越多)，采用控制变量法对效率进行比较，即在保持其他条件如矩阵维数、数据精度等不变的情况下，增加迭代次数，所得结果如下所示：

为更加清楚地体现并行实现的优势，将表 1 的数据以直方图的形式展示出来，见图 4。从图 4 中可以看出，当迭代次数较少时，串行实现与并行实现所需的 cycle 数差距不大，且并行实现的 cycle 数有时会出现比串行实现所需的 cycle 数多的情况；但当迭代次数增加时，并行实现的优势逐渐体现出来，当迭代次数为 20 次时，串行的效率就只是并行的 69.7%，随着迭代次数的加大，并行的效率更是不断上升。

```
[ 12.6351,   -1.9208,   -3.2268,    0.6072,   -2.0858,   -1.3467,   -0.5611,    4.4287,
  -1.9208,    8.0002,    1.9015,   -0.3800,    0.6324,   -1.2629,    1.6724,    0.1012,
  -3.2268,    1.9015,    9.6686,   -4.5581,    5.2569,   -5.2729,   -0.0747,   -0.0788,
   0.6072,   -0.3800,   -4.5581,    7.7524,   -2.5850,    2.2863,   -0.7494,    0.6535,
  -2.0858,    0.6324,    5.2569,   -2.5850,    4.1810,   -2.1095,   -1.7881,    2.0431,
  -1.3467,   -1.2629,   -5.2729,    2.2863,   -2.1095,    5.2536,   -0.5986,   -0.3366,
  -0.5611,    1.6724,   -0.0747,   -0.7494,   -1.7881,   -0.5986,    6.6779,   -4.2090,
   4.4287,    0.1012,   -0.0788,    0.6535,    2.0431,   -0.3366,   -4.2090,    7.2049,]
```

**Figure 3. Experimental floating-point**
**图 3. 实验浮点矩阵**

**Table 1. Serial and parallel implementations with cycle number**
**表 1. 串并行实现所用 cycle 数**

| 迭代次数 | 串行实现所用 cycle 数 | 并行实现所用 cycle 数 |
|---|---|---|
| 8 | 3278268 | 4247376 |
| 20 | 8104920 | 5653764 |
| 50 | 19944208 | 9191054 |
| 100 | 39415158 | 15100850 |
| 200 | 78025414 | 26880504 |

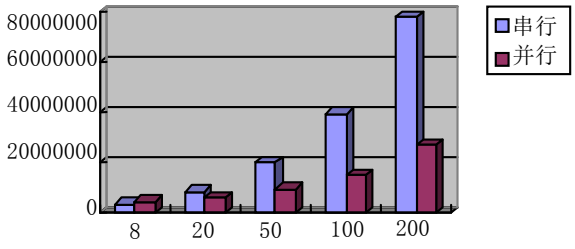



**Figure 4. Serial and parallel implementations with cycle number comparison chart**
**图 4. 串并行实现所用 cycle 数对比图**

## 6. 总结

综上所述，本文在提出一种改进的单侧 Jacobi 矩阵求逆算法后，利用 TI 的 TMS320C6474 DSP 和 DSP/BIOS 并行实现了此算法。最后为证明此改进的单侧 Jacobi 算法在并行实现上的高效，本文比较了在同样计算结果精度的情况下，并行算法与串行算法的指令执行周期数。由本文可见，在对单侧 Jacobi 算法进行改进并采用并行实现后，算法的执行效率有了大幅度的提高，矩阵的求逆也更为高效和快速，这将为 MIMO 通信中高维数矩阵的求逆提供了一种可能的解决方法。

但由于改进的算法采用了近似，因此当矩阵维数增加至百维以上时，迭代次数急剧增加，近似所带来的误差也会增加，由此误差带来的逆矩阵的非正交性将是本文的下一步研究目标。此外，采用汇编语言来实现并行算法，进一步提高运行效率也是本文的努力方向。

## 7. 致谢

本论文是基于东南大学国家大学生创新性实验计划项目《宽带无线通信系统中高效矩阵运算及其 DSP 实现》的实验成果撰写的，项目编号为 111028609。感谢东南大学信息科学与工程学院在项目的研究过程中提供了实验环境与实验平台，同时感谢项目指导老师金石老师与江彬老师，他们在算法的改进过程中提供了很有价值的建议和指导。

## 参考文献 (References)

[1] J. Demmel, K. Veselic. Jacobi's method is more accurate than QR. SIAM Journal on Matrix Analysis and Applications, 1992, 11: 1204-1246.

[2] A. H. Sameh. On Jacobi and Jacobi-like algorithms for a parallel computer. Mathematics of Computation, 1971, 25(115): 579-590.

[3] R. P. Brent, F. T. Luk. The solution of singular-value and symmetric eigenvale problems on multiprocessor arrays. SIAM Journal on Scientific and Statistical Computing, 1985, 6(1): 69-84.

[4] M. Gentleman. Least squares computations by Givens transformations without square roots. Journal of the Institute of Mathematics and its Applications, 1973, 12(3): 329-336.

[5] A. A. Anda, H. Park. Fast plane rotations with dynamic scaling. SIAM Journal on Matrix Analysis and Applications, 1994, 15(1): 162-174.

[6] J. Yan, S. Osher. A local discontinuous Galerkin method for directly solving Hamilton-Jacobi equations, Journal of Computational Physics, 2011, 230(1): 232-244.

[7] M. Baggio, J. de Boer and K. Holsheimer. Hamilton-Jacobi renormalization for Lifshitz spacetime. Journal of High Energy Physics, 2012. http://arxiv.org/abs/1107.5562