南開大學
Nankai University

计算机学院

并行程序设计

Gröbner 基计算中的高斯消元并行化改进

小组成员姓名：丁屹、卢麒萱

小组成员学号：2013280、2010519

专业：计算机科学与技术

2022 年 3 月 30 日

目录

1 研究问题

1.1 高斯消元法

数学上，高斯消元法（或译：高斯消去法），是线性代数规划中的一个算法，可用来为线性方程组求解。但其算法十分复杂，不常用于加减消元法，求出矩阵的秩，以及求出可逆方阵的逆矩阵。不过，如果有过百万条等式时，这个算法会十分省时。一些极大的方程组通常会用迭代法以及花式消元来解决。当用于一个矩阵时，高斯消元法会产生出一个“行梯阵式”。高斯消元法可以用在计算机中来解决数千条等式及未知数。亦有一些方法特地用来解决一些有特别排列的系数的方程组。

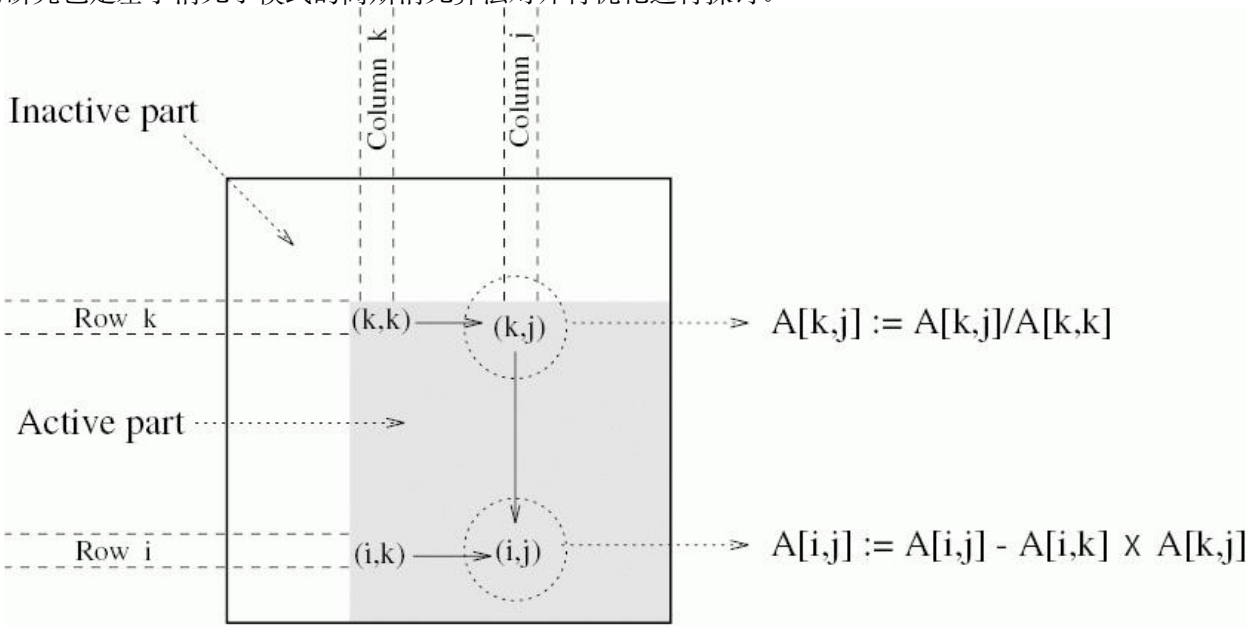
高斯消元法的算法复杂度是 $O(n^3)$；这就是说，如果系数矩阵的是 $n \times n$，那么高斯消元法所需要的计算量大约与 n^3 成比例。

高斯消元法可以用在电脑中来解决数千条等式及未知数。不过，如果有过百万条等式时，这个算法会十分费时。一些极大的方程组通常会用迭代法来解决。亦有一些方法特地用来解决一些有特别排列的系数的方程组。

高斯消元法可用在任何域中。

高斯消元法对于一些矩阵来说是稳定的。对于普遍的矩阵来说，高斯消元法在应用上通常也是稳定的，不过亦有例外。

随着多核处理器的日益普及，现在我们可以利用线程级并行高斯消元算法来提高计算的速度。我们的研究也是基于消元子模式的高斯消元算法对并行优化进行探讨。



普通高斯消去算法

```
1  procedure LU (A)
2  begin
3  for k := 1 to n do
4  for j := k + 1 to n do
5  A[k, j] := A[k, j] / A[k, k];
6  endfor;
7  A[k, k] := 1.0;
8  for i := k + 1 to n do
9  for j := k + 1 to n do
```

```
10    A[i, j] := A[i, j] - A[i, k] * A[k, j];
11    endfor;
12    A[i, k] := 0;
13    endfor;
14      endfor;
15    end LU
```

观察高斯消去算法，注意到伪代码第 4，5 行第一个内嵌循环中的 A[k, j] := A[k, j] / A[k, k] 以及伪代码第 8、9、10 行双层 for 循环中的 A[i, j] := A[i, j] - A[i, k] * A[k, j] 都是可以进行向量化的循环。可以通过 SIMD 扩展指令对这两步进行并行优化。

1.2 消元子模式的高斯消元算法

本算法源自布尔 Gröbner 基计算。Gröbner 基理论是一种在国外被普遍认同的用于求解多变元高次方程系统的有效算法, 其概念最早由 Buchberger 提出。其本质是从多项式环中任意理想的生成元出发, 刻画和计算出一组具有“好的”性质的生成元, 进而研究理想的结构并进行某些理想运算; 由于数学、科学和工程学中的许多问题都可以用多元多项式方程组表示 (例如, 理想, 模块和矩阵),Gröbner 基的代数算法在理论物理学、应用科学和工程学中具有广泛的应用。在 HFE80 的 Gröbner 基计算过程中，高斯消元时间占比可以达到 90% 以上。考虑到此算法中高斯消元的特殊性，为加快高斯消元速度，设计此算法。

1.2.1 符号说明

R: 所有消元子构成的集合

$R[i]$: 首项为 i 的消元子

E: 所有被消元行构成的数组

$E[i]$: 第 i 个被消元行

$lp(E[i])$: 被消元行第 i 行的首项

这里首项的含义: 首项是指某行下标最大的非零项的下标，如某行为 011000，从左到右下标分别为 5,4,3,2,1,0，那么首项为 4，因为该行非零项下标为 3,4，其中最大值为 4。

1.2.2 算法伪代码

Algorithm 1 串行算法伪代码

```
1:  function GAUSS
2:      for i := 0 to m − 1 do
3:          while E[i]! = 0 do
4:              if R[lp(E[i])]! = NULL then
5:                  E[i] := E[i]−R[lp(E[i])]
6:              else
7:                  R[lp(E[i])] := E[i]
8:                  break
9:              end if
10:         end while
11:     end for
```

12: **return** E
13: **end function**

其中外层循环表示遍历每个被消元行。内层循环表示针对每个被消元行，如果该行未被消为 0，那么根据其首项选择消元子进行消元；当存在合适的消元子，则用该消元子进行消元；否则将该被消元行作为消元子，参与后续高斯消元过程。并行算法内容由串行算法改进。

2 背景知识

2.1 数据结构

可采用位向量方式存储每个消元子和被消元行，优点是消元操作变为位向量异或操作，算法实现简单，且适合并行化，易达到更高的并行效率，缺点是 Gröbner 基计算中产生的消元子和被消元行非常稀疏，非零元素（1 元素）在 5 % 以下，位向量存储和计算可能并非最优。也可采用类似倒排链表的存储及方式——可认为是稀疏 0 / 1 矩阵的紧凑存储方式，每个消元子和被消元行只保存 1 元素的位置，且按升序排列，从而类似倒排链表数据结构。优点是存储空间占用更少，缺点是算法设计更复杂，并行化难度高。

2.2 数据访问

矩阵规模可能非常庞大，达到数百万行 / 列，难以全部放入内存。此时，需要设计的是外存算法，考虑如何分批次将数据读入内存进行处理，同时又保证正确性。对外存算法，算法分析和时间测试除了考虑计算之外，还要考虑 I / O 时间。

2.3 并行化方法

1. 对位向量存储方式，两行间消元操作的并行化很直接，无论 SIMD、多线程还是 MPI、GPU，将向量拆分，子向量的异或即自然形成任务，可分配给不同的计算单元。当矩阵规模大到百万级别时，采取这种并行方式就够了。但当矩阵规模没有那么大时，一个消元操作计算量不足以支撑较大规模并行，就需要考虑消元操作间的并行，设计适合的并行任务划分，在提高并发度的同时保证正确性。

2. 对类倒排链表存储方式，可以考虑循环展开和多线程优化。

3. 当矩阵规模非常庞大，需使用外存算法时就要同时考虑计算和访存。多线程并行化时可考虑计算和访存异步模式，在前台线程进行消元计算时、后台线程读取下一步要处理的数据，这需要仔细设计计算和 I / O 步骤，降低依赖关系，以便实现异步模式。

3 研究计划

4 参考文献

[1][2]

参考文献

[1] 牟晨琪. 计算机代数. http://cmou.net/files/PolyAlg2-2020.pdf, 2020.

[2] 狄鹏. Grobner 基生成算法的并行. https://www.academia.edu/18478407/Grobner%E5%9F%BA%E7%94%9F%E6%88%90%E7%AE%97%E6%B3%95%E7%9A%84%E5%B9%B6%E8%A1%8C, 2008.