计算机科学
COMPUTER SCIENCE

基于双倍双精度施密特正交化方法的QR分解算法

金洁茜，谢和虎，杜配冰，全哲，姜浩

引用本文

金洁茜，谢和虎，杜配冰，全哲，姜浩基于双倍双精度施密特正交化方法的QR分解算法[J]．计算机科学，2023，50(6)：45-51.

JIN Jiexi，XIE Hehu，DU Peibing，QUAN Zhe，JIANG Hao. QR Decomposition Based on Double-double Precision Gram-Schmidt Orthogonalization Method [J]. Computer Science，2023，50(6)：45-51.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230200209

基于双倍双精度施密特正交化方法的 QR 分解算法

金洁茜[1]　谢和虎[2]　杜配冰[3]　全　哲[1]　姜　浩[4]
1 湖南大学信息科学与工程学院　长沙 410082
2 中国科学院数学与系统科学研究院　北京 100190
3 西北核技术研究所　西安 710024
4 国防科技大学计算机学院　长沙 410073
(jinjiexi@hnu.edu.cn)

摘　要　当矩阵的规模较大或者条件数较高时,格拉姆-施密特(Gram-Schmidt)正交化算法和其相关修正算法时常表现出数值不稳定性的现象。为了解决该问题,探索了修正 Gram-Schmidt 算法(MGS)中舍入误差的累积效应,然后基于无误差变换技术和双倍双精度算法,设计并实现了双倍双精度修正 Gram-Schmidt 正交化算法(DDMGS)。该算法的精度测试中显示所提算法较分块施密特正交化(BMGS_SVL,BMGS_CWY,BCGS_PIP 与 BCGS_PIO)的变体算法具有更好的数值稳定性,证明了 DDMGS 算法能够有效地减少矩阵的正交性损失,提升数值精度,展示了所提算法的可靠性。在算法的性能测试中,首先计算并比较了不同算法的浮点计算量(flops),随后将所提 DDMGS 算法与修正施密特正交化算法在 ARM 和 Intel 两款处理器上作比较,虽然 DDMGS 算法的运行时间分别是 MGS 的 5.03 倍和 18.06 倍左右,但获得了明显的精度提升效果。

关键词:施密特正交化算法;QR 分解;无误差变换;双倍双精度算法;舍入误差;浮点算术
中图法分类号　TP391

QR Decomposition Based on Double-double Precision Gram-Schmidt Orthogonalization Method

JIN Jiexi[1],XIE Hehu[2],DU Peibing[3],QUAN Zhe[1] and JIANG Hao[4]
1 College of Computer Science and Electronic Engineering,Hunan University,Changsha 410082,China
2 Academy of Mathematics and Systems Science,Chinese Academy of Sciences,Beijing 100190,China
3 Northwest Insititute of Nuclear Technology,Xi'an 710024,China
4 College of Computer Science and Technology,National University of Defense Technology,Changsha 410073,China

Abstract　The Gram-Schmidt orthogonalization algorithm and its related modified algorithms often show numerical instability when computing ill-conditioned or large-scale matrices. To solve this problem,this paper explores the cumulative effect of round-off errors of modified Gram-Schmidt algorithm(MGS),and then designs and implements a double-double precision modified Gram-Schmidt orthogonalization algorithm(DDMGS) based on the error-free transformation technology and double-double precision algorithm. A variety of accuracy tests illustrate that DDMGS algorithm has better numerical stability than the varients of BMGS_SVL,BMGS_CWY,BCGS_PIP and BCGS_PIO algorithms,which proves that DDMGS algorithm can effectively reduce the loss of orthogonality of matrix,improve the numerical accuracy,and demonstrate the stability of our algorithm. In the performance test, the floating point computations(flops) of different algorithms are calculated and then compared DDMGS algorithm with the modified Gram-Schmidt algorithm on ARM and Intel processors,the runtime of the DDMGS algorithm proposed in this paper is about 5.03 and 18.06 times that of MGS respectively,but the accuracy is improved significantly.

Keywords　Gram-Schmidt algorithm,QR decomposition,Error-free transformation,Double-double precision algorithm,Roundoff error,Floating-point arithmetic

1　引言

格拉姆-施密特(Gram-Schmidt)正交化在矩阵计算中起着非常重要的作用,常常被用于最小二乘问题和矩阵 QR 分解的计算。假设 $\boldsymbol{A}\in\boldsymbol{R}^{m\times n}$ 是列满秩的实矩阵,且 $m\geqslant n$,应用 Gram-Schmidt 算法计算矩阵分解:

到稿日期:2023-02-27　返修日期:2023-04-11
基金项目:国家自然科学基金(61907034);国家重点研发计划(2020YFA0709803);科学挑战专题(TZ2016002)
This work was supported by the National Natural Science Foundation of China(61907034),National Key Research and Development Program of China(2020YFA0709803) and Science Challenge Project(TZ2016002).
通信作者:姜浩(haojiang@nudt.edu.cn)

$$\boldsymbol{A}=\boldsymbol{QR} \tag{1}$$

其中，$\boldsymbol{Q}\in\boldsymbol{R}^{m\times n}$是正交矩阵，$\boldsymbol{R}\in\boldsymbol{R}^{n\times n}$是上三角矩阵。矩阵的QR分解在线性代数中有着广泛应用，有助于求解矩阵特征值和一些病态方程，还能加速求解速度。大型稀疏线性系统和特征值问题的Krylov子空间方法也依赖于Gram-Schmidt正交化[1]。

Gram-Schmidt变换是21世纪初以来被广泛讨论的新的影像融合方法之一，与其他的融合方法相比，采用Gram-Schmidt方法得到的影像更清晰，保留的信息更全面，是一种较好的影像融合方法。例如，江苏省测绘工程院的Huang等[2]于2010年提出了基于Gram-Schmidt变换进行影像融合的方法，采用了QuickBird影像数据，通过比较Gram-Schmidt变换和HIS色彩变换的融合结果发现，基于Gram-Schmidt变换的影像融合方法具有更好的图像保真效果。

Gram-Schmidt方法除了能够应用于图像和视频处理，还能应用于一系列前沿科技领域。例如，悉尼大学的Zhang等[3]于2021年提出了Gram-Schmidt正交过程的高效量子算法(QGSP)和一种有效的读出协议，其在整个读出协议中，对Gram-Schmidt正交归一化过程进行了量子推广。

随着科技的发展，现如今Gram-Schmidt正交化还被广泛应用于实际生活中，如信号处理、统计计算、经济模型等应用领域。然而在数值计算中，施密特正交化算法具有数值不稳定性，因为其在计算过程中存在舍入误差，并且舍入误差具有累积性，因此在有限精度的病态矩阵中会导致大量误差的产生，得到不可靠的计算结果。

针对Gram-Schmidt正交化的数值不稳定性问题，需要高精度算法来减少正交损失，提升计算结果的可靠性。最直接的手段是利用更高的工作精度，其中软件模拟高精度比较知名的是Bailey[4-5]开发的double-double算法与软件，其要求两个double类型的浮点数无重合，采用该方法进行运算能够令计算结果达到近似双倍双精度。双倍双精度算法的核心思想是Ogita[6]等于2005年基于Dekker[7]，Kahan[8]和Knuth[9]的算法正式提出的无误差变换技术(Error-Free Transformation Technology，EFT)。

为降低Gram-Schmidt正交化产生的正交性损失，本文基于无误差变换技术改进了应用最为广泛的修正的Gram-Schmidt算法(Modified Gram-Schmidt，MGS)[10]，设计并实现了高精度修正施密特正交化算法(Double-double Modified Gram-Schmidt，DDMGS)。通过数值实验可以看出，本文算法得到的正交矩阵的正交性损失较小，在问题病态程度受限的条件下，能够给出足够精确的数值计算结果。精度实验显示所提DDMGS算法比Carson在文献[11]与文献[12]中提出的分块施密特正交化算法具有更好的数值稳定性，性能实验显示本文提出的高精度修正施密特正交化算法DDMGS与高精度经典施密特正交化算法(Double-double Classical Gram-Schmidt，DDCGS)的计算速度几乎一致，但DDMGS的精确性更高。

本文第2节概述了本课题的研究意义，以及论文结构；第2节回顾了Gram-Schmidt算法的发展历程以及它的各种变体；第3节引入了浮点数的相关概念与无误差变换技术；第4节介绍了双倍双精度算法和经典的Gram-Schmidt算法(Classical Gram-Schmidt，CGS)[13]与修正的Gram-Schmidt算法，设计并实现了双倍双精度的Gram-Schmidt算法；第5节采用不同的病态矩阵进行测试，对比CGS、MGS和双倍双精度Gram-Schmidt算法计算结果的正交性损失与运行时间，且将双倍双精度的Gram-Schmidt算法与文献[11]和文献[12]中的算法结果的正交性损失进行了比较，通过分析测试结果，验证了本文算法的可靠性；最后总结全文并展望未来。

2 研究现状

1907年，Schmidt发表了一篇关于积分方程的论文[13]，在该文中提出了一种正交化算法，该算法后来被称为经典Gram-Schmidt算法。Schmidt在论文中指出这些公式实质上是由Gram[14]提出的。1935年，Wong将Schmidt和Gram提出的算法联系在一起，用于描述正交化过程[15]。然而数值实验表明，经典Gram-Schmidt算法在存在舍入误差的情况下是不可靠的且数值不稳定。为改善该情况，研究者们提出了很多数值稳定且相对有效的算法。目前应用最为广泛的施密特正交化算法是修正的Gram-Schmidt算法和迭代的Gram-Schmidt算法(Iterative Classical Gram-Schmidt，ICGS)，因为它们具有良好的数值稳定性。而修正的Gram-Schmidt算法最早是由Laplace于1816年推导得出，但他并未从正交的角度来解释该算法[16]。1976年，Daniel等发现使用重正交化方法能够确保数值稳定性，即保持计算的列的正交性，进而提出了带有重正交的CGSR算法和停止迭代的标准，令矩阵的正交性得到提升[17]。

除了以上两种应用最为广泛的算法外，研究者基于Gram-Schmidt方法还提出了众多既能够提高性能又能保证矩阵的正交性达到机器精度的算法。为提升施密特正交化算法的性能，1991年，Jably等提出了一种分块的MGSR算法，该算法采用了重正交法，其生成的矩阵$\boldsymbol{Q}$的正交性损失与矩阵$\boldsymbol{A}$的条件数成正比[18]。2013年，美国宾州州立大学的Barlow等提出了一种重正交块经典Gram-Schmidt算法(BCGS2)，该算法可以使用矩阵-矩阵运算来实现，这使得它在现代架构上比基于矩阵-向量运算和向量-向量运算的正交分解算法更加高效[19]。伊朗科尔曼大学的Rivaz等[20]于2016年提出了一种通过更新算法的内环来修正MGS正交化的Optimized Modified Gram-Schmidt算法(OMGS)，减小了MGS算法过程中更多固有误差，改善了正交性损失。

Block Gram-Schmidt算法在许多科学计算应用中是必不可少的内核，但对许多常用变体的稳定性的处理依然是目前的研究热点。因此，捷克查理大学的Carson等[11]于2021年开发了两种新的BCGS变体——BCGS-PIO和BCGS-PIP。与标准BCGS相比，这些新型BCGS具有更稳定的数值性能，同时保持了较低的通信成本。同年，Carson等给出了块Gram-Schmidt算法的全面分类[12]，还推导出了低同步变体的新块版本，并在各种具有挑战性的样例中证明了它们的性能与稳定性。

3 预备知识

由于施密特正交化具有数值不稳定性，计算过程中的舍入误差会使最终结果的正交性变得很糟糕，因此本文引入了

无误差变换技术，改进经典和修正的 Gram-Schmidt 正交化方法，以减小舍入误差的累积效应。本节首先简要介绍了浮点数算术标准的相关概念，随后阐述了无误差变换技术和相关算法。

3.1 浮点数背景

1985 年，IEEE 组织制定了 IEEE 二进制浮点数算术标准，简称 IEEE-754(1985)，这是目前使用最广泛的浮点数运算标准，它支持舍入，溢出等操作，规定了 4 种表示浮点数值的方式，分别为单精度、双精度、延伸单精度与延伸双精度[21]，并指明了 4 种舍入格式，分别为向负无穷舍入、向正无穷舍入、向零舍入以及就近舍入。一般将舍入方式默认为就近舍入(Round to Nearest)。在本文中，所有的计算都是以二进制格式进行的，采用就近舍入格式，并且假设在计算过程中既不发生上溢也不发生下溢。

3.2 无误差变换技术

2005 年 Ogita 等[6]提出了无误差变换技术的概念，其定义如下：两个浮点数 $a,b\in F$，$\circ\in\{+,-,\times\}$，且 $fl(a\circ b)\in F$，存在：

$$x=fl(a\circ b),a\circ b=x+y,y\in F \tag{2}$$

其中，x 表示 a 和 b 经过浮点计算后的最好的近似结果，$y\in F$ 表示运算结果的精确舍入误差。上述运算中没有除法，因为 y 在除法中不一定是浮点数。为实现高精度除法只能采用近似或修正的无误差变换。

Dekker 于 1971 年提出了浮点数加法的无误差变换算法——FastTwoSum 算法[7](见算法 1)，将一对浮点数 (a,b) 转换成一对新的浮点数 (x,y)，令结果满足 $a+b=x+y$，但需要满足前提条件(先验信息)$|a|\geqslant|b|$，其计算过程需 3flops。1998 年，Knuth 提出了 TwoSum 算法[9](见算法 2)，该算法不需要满足 $|a|\geqslant|b|$，但其计算过程需要更多的浮点运算量，共需 6flops。

算法 1 两个浮点数 a 和 b 的加法无误差变换[7]($|a|\geqslant|b|$)

function[x,y]=FastTwoSum(a,b)

1. x=fl(a+b)
2. y=fl(b−(x−a))

算法 2 两个浮点数 a 和 b 的加法无误差变换[9]

function[x,y]=TwoSum(a,b)

1. x=fl(a+b)
2. z=fl(x−a)
3. y=fl((a−(x−z))+(b−z))

Dekker 基于 Kahan 于 1965 年提出的 Split 算法[8]，提出了 TwoProd 算法[7](见算法 3)。该算法满足无误差变换技术的定义，用于计算两个浮点数相乘的运算结果，其运算需要 17flops。

算法 3 两个浮点数乘法的无误差变换[7]

function[x,y]=TwoProd(a,b)

1. x=a×b
2. $[a_h,a_l]$=Split(a)
3. $[b_h,b_l]$=Split(b)
4. $y=a_l\times b_l-(((x-a_h\times b_h)-a_l\times b_h)-a_h\times b_l)$

双倍双精度算法的核心思想就是无误差变换技术，部分基础的双倍双精度算法便是由上述的 3 个算法构造而来，例如，add_dd_d 算法与 add_dd_dd 算法由 TwoSum 算法以及 FastTwoSum 算法构造得到，prod_dd_d 算法和 prod_dd_dd 算法由 TwoProd 算法与 FastTwoSum 算法组合而成。可以看出，在双倍双精度算法中，FastTwoSum 算法是不可或缺的，因为它可以将双倍双精度算法的输出格式规格化。

4 高精度施密特正交化算法实现

施密特正交化是求欧氏空间的标准正交基的一种方法，它包括正交化和单位化两个步骤。应用最广泛的施密特正交化方法是经典 Gram-Schmidt 算法(CGS)[13](见算法 4)和修正的 Gram-Schmidt 算法(MGS)[10](见算法 5)。

算法 4 经典 Gram-Schmidt 算法[13]

function $[\mathbf{Q},\mathbf{R}]$=CGS($\mathbf{A}$)

1. for k=1:n
2. $\quad w=\mathbf{A}_k$
3. $\quad$for j=1:k−1
4. $\quad\quad R_{jk}=dot(\mathbf{Q}_j^T,\mathbf{A}_k)$
5. $\quad\quad w=w-\mathbf{Q}_j\times\mathbf{R}_{jk}$
6. $\quad$end
7. $\quad\mathbf{R}_{kk}=\|w\|_2$
8. $\quad\mathbf{Q}_k=w/\mathbf{R}_{kk}$
9. end

算法 5 修正 Gram-Schmidt 算法[10]

function $[\mathbf{Q},\mathbf{R}]$=MGS($\mathbf{A}$)

1. for k=1:n
2. $\quad w=A_k$
3. $\quad$for j=1:k−1
4. $\quad\quad\mathbf{R}_{jk}=dot(\mathbf{Q}_j^T,w)$
5. $\quad\quad w=w-\mathbf{Q}_j\times\mathbf{R}_{jk}$
6. $\quad$end
7. $\quad\mathbf{R}_{kk}=\|w\|_2$
8. $\quad\mathbf{Q}_k=w/\mathbf{R}_{kk}$
9. end

由于 CGS 和 MGS 算法是按照列实现的，我们首先给出 QR 分解的列表示，如式(3)所示，算法 4 和算法 5 中的 $Q_k=q_k$，$A_k=a_k$。

$$(a_1\ \ a_2\ \ \cdots\ \ a_n)=(q_1\ \ q_2\ \ \cdots\ \ q_n)\begin{pmatrix}r_{11}&r_{12}&\cdots&r_{1n}\\&r_{22}&&r_{2n}\\&&\ddots&\vdots\\&&&r_{nn}\end{pmatrix} \tag{3}$$

在 CGS 算法的基础上修改计算顺序即可得到 MGS 算法，CGS 算法是一次减去所有投影，而 MGS 算法是计算一次投影便在 a_k 上减去投影。它们在数学上是等价的，但具有不同的数值特性。MGS 算法在数值上更加稳定，CGS 计算得到的正交矩阵的正交性常常会严重损失。

Bjorck 于 1994 年通过 Laeuchli 矩阵证明 MGS 算法得到的正交性损失取决于条件数，并与条件数成正比[22]，其数值关系如式(4)所示：

$$\|\boldsymbol{E}_{MGS}\|_2=\|\boldsymbol{I}-\boldsymbol{Q}^T\boldsymbol{Q}\|_2\approx\mathcal{O}(u)\kappa_2(\boldsymbol{A}) \tag{4}$$

4.1 双倍双精度算法

提高精度可以从硬件和软件两个方面来实现，在软件中，我们可以使用任意精度库，而工作精度可以自行决定。然而许多应用程序不需要任意精度，只需使用工作精度的较小倍，

例如2倍或4倍，就能够满足计算需求。这种固定精度的算法比任意精度的算法要快得多。

加利福尼亚大学的Hida和劳伦斯伯克利国家实验室的Li与Bailey开发的QD Library[23]中有double-double数据格式与quad-double数据格式，double-double数据格式可令计算结果达到近似双倍双精度。本文用到的double-double算法如表1所列。

表1 部分基础double-double算法

Table 1 Some basic double-double algorithms

函数名	函数意义
add_dd_d	double-double和double数加法
add_dd_dd	double-double和double-double数加法
prod_dd_d	double-double和double数乘法
prod_dd_dd	double-double和double-double数乘法
div_dd_d	double-double和double数除法
div_dd_dd	double-double和double-double数除法

浮点数对(x_h, x_l)为一个double-double格式，x_h和x_l是两个不重叠的双精度浮点数，x为双倍双精度数，表示x_h和x_l的精确和。即：

$$x = x_h + x_l \tag{5}$$

$$|x_l| \leqslant \frac{1}{2} ulp(x_h) \leqslant u|x_h| \tag{6}$$

4.2 双倍双精度施密特正交化算法

观察算法4和算法5可以发现，算法的主体由点积运算dot、向量乘加运算AXPY、二范数计算nrm和除法运算div 4个核心操作组成。为了实现双倍双精度的施密特正交化算法，需要上述4个核心操作的双倍双精度算法。根据表1，本文设计并构建了表2中的相关函数。

表2 核心double-double算法

Table 2 Core double-double algorithms

函数名	函数意义
sqrt_dd	double-double数的开方
norm_dd	double-double向量的二范数
dot_dd_d	double-double和double数向量的点积
dot_dd_dd	double-double和double-double向量的点积

向量的乘加运算通过add_dd_dd算法与prod_dd_dd算法即可实现，除法运算可由div_dd_dd算法完成。由此，首先基于双倍双精度的加法运算与乘法运算设计双倍双精度的点积算法。第一步，通过prod_dd_dd算法计算向量$\vec{a}$与向量$\vec{b}$第一个数的高位与低位的乘积结果，得到精确值q_h和误差值q_l，满足$q_h+q_l=(a_h(1)+a_l(1))\times(b_h(1)+b_l(1))$。第二步，同样采用prod_dd_dd算法，计算得到$r_h+r_l=(a_h(i)+a_l(i))\times(b_h(i)+b_l(i))$，此时$i=2$。第三步，完成精确值与误差值的累加，通过add_dd_dd算法更新q_h与q_l，即$q_h+q_l=r_h+r_l+q_h+q_l$。不断重复第二步与第三步，每循环一次i值加1，直到i值等于向量长度，最后得到的q_h与q_l分别表示向量$\vec{a_h}$，$\vec{a_l}$，$\vec{b_h}$与$\vec{b_l}$进行双倍双精度点积运算dot_dd_dd(见算法6)后得到的精确值与误差值。将prod_dd_dd算法替代为prod_dd_d即可得到dot_dd_d算法(见算法7)。若要实现DDCGS算法，仅需将DDMGS算法中的dot_dd_dd算法替换为dot_dd_d算法，即可得到DDCGS算法。

算法6 两个double-double向量的点积算法

function $[q_h, q_l]$=dot_dd_dd($\vec{a_h}$, $\vec{a_l}$, $\vec{b_h}$, $\vec{b_l}$)

1. $[q_h, q_l]$=prod_dd_dd($a_h(1)$, $a_l(1)$, $b_h(1)$, $b_l(1)$)
2. for i=2:s
3. $[r_h, r_l]$=prod_dd_dd($a_h(i)$, $a_l(i)$, $b_h(i)$, $b_l(i)$)
4. $[q_h, q_l]$=add_dd_dd(r_h, r_l, q_h, q_l)
5. end

算法7 double-double向量与double向量的点积算法

function $[q_h, q_l]$=dot_dd_d($\vec{a_h}$, $\vec{a_l}$, $\vec{b}$)

1. $[q_h, q_l]$=prod_dd_d($a_h(1)$, $a_l(1)$, $b(1)$)
2. for i=2:s
3. $[r_h, r_l]$=prod_dd_d($a_h(i)$, $a_l(i)$, $b(i)$)
4. $[q_h, q_l]$=add_dd_dd(r_h, r_l, q_h, q_l)
5. end

接下来我们需要完成二范数算法的设计。向量的二范数计算与点积算法类似，首先将向量$\vec{w}$与向量$\vec{w}$进行点积计算，将得到的值开方即可得到向量的二范数。为提升二范数算法的精度，我们根据式(7)与式(8)实现将sqrt()修改为双倍双精度的开方算法sqrt_dd(见算法8)，用r_h和r_l存储计算得到的解，r_h表示运算结果的近似解，r_l可以理解为修正量。结合dot_dd_dd算法和sqrt_dd算法即可得到双倍双精度的二范数算法norm_dd(见算法9)。

$$x = 1.0/sqrt(a) \tag{7}$$

$$sqrt(a) = a\times x + [a-(a\times x)^2]\times x/2 \tag{8}$$

算法8 double-double格式数的开方运算

function $[r_h, r_l]$=sqrt_dd(a_h, a_l)

1. $[x_h, x_l]$=div_dd_d(1.0, 0.0, sqrt(a_h))
2. $[b_h, b_l]$=prod_dd_dd(a_h, a_l, x_h, x_l)
3. $[m_h, m_l]$=prod_dd_dd(b_h, b_l, b_h, b_l)
4. $[t_h, t_l]$=add_dd_dd(a_h, a_l, -m_h, -m_l)
5. $[k_h, k_l]$=prod_dd_dd(t_h, t_l, x_h, x_l)
6. $[t_h, t_l]$=div_dd_d(k_h, k_l, 2.0)
7. $[r_h, r_l]$=add_dd_dd(b_h, b_l, t_h, t_l)

算法9 double-double格式数的二范数运算

function $[r_h, r_l]$ =norm_dd($\vec{x_h}$, $\vec{x_l}$)

1. $[t_h, t_l]$ =dot_dd_dd($\vec{x_h}'$, $\vec{x_l}'$, $\vec{x_h}$, $\vec{x_l}$)
2. $[r_h, r_l]$=sqrt_dd(t_h, t_l)

基于上述双倍双精度算法，即可实现双倍双精度的修正Gram-Schmit算法DDMGS(见算法10)和双倍双精度的经典Gram-Schmit算法DDCGS(见算法11)。

算法10 双倍双精度修正Gram-Schmidt算法

function [**Q**, **R**]=DDMGS(A)

1. for k=1:n
2. wh=A_k
3. wl=zeros(m, 1)
4. for j=1:k-1
5. $[\mathbf{R}h_{jk}, Rl_{jk}]$=dot_dd_dd($\mathbf{Q}h_j^T$, $\mathbf{Q}l_j^T$, wh, wl)
6. for i=1:m
7. $[yh_i, yl_i]$=prod_dd_dd(-$\mathbf{Q}h_{ij}$, -$\mathbf{Q}l_{ij}$, $\mathbf{R}h_{jk}$, Rl_{jk})
8. $[wh_i, wl_i]$=add_dd_dd(yh_i, yl_i, wh_i, wl_i)
9. end
10. end
11. $[\mathbf{R}h_{kk}, \mathbf{R}l_{kk}]$=norm_dd(wh, wl)
12. for i=1:m
13. $[\mathbf{Q}h_{ik}, \mathbf{Q}l_{ik}]$=div_dd_dd($wh_i$, wl_i, $\mathbf{R}h_{kk}$, Rl_{kk})

```
14. end
15. end
```

算法 11 高精度经典 Gram-Schmidt 算法

```
function [Q,R]=DDCGS(A)
1. for k=1:n
2.    wh=A_k
3.    wl=zeros(m,1)
4.    for j=1:k-1
5.       [Rh_jk,Rl_jk]=dot_dd_d(Qh_j^T, Ql_j^T,A_k)
6.       for i=1:m
7.          [yh_i,yl_i]=prod_dd_dd(-Qh_ij,-Ql_ij,Rh_jk,Rl_jk)
8.          [wh_i,wl_i]=add_dd_dd(yh_i,yl_i,wh_i,wl_i)
9.       end
10.   end
11.   [Rh_kk,Rl_kk]=norm_dd(wh,wl)
12.   for i=1:m
13.      [Qh_ik, Ql_ik]=div_dd_dd(wh_i,wl_i,Rh_kk,Rl_kk)
14.   end
15. end
```

5 数值实验

为验证本文 DDMGS 算法的可靠性，我们选择了 14 个不同类型的病态矩阵作为测试矩阵进行数值实验，测试矩阵数量多，涵盖面较广，比较不同算法运行得到的正交性损失与运行时间，通过计算 $\|\boldsymbol{I}-\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}\|_2$ 得到算法运行结果矩阵 $\boldsymbol{Q}$ 的正交性损失。

5.1 测试矩阵

首先选用 $\boldsymbol{X}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ 作为测试矩阵进行数值实验，$\boldsymbol{U}$ 和 $\boldsymbol{V}$ 是通过 orth(rand $n(m,n)$)生成的随机正交矩阵，生成 $\boldsymbol{V}$ 时，令 $m=n$，$\boldsymbol{\Sigma}$ 是一个 $n\times n$ 的对角矩阵，通过设置不同的最小奇异值，可以得到不同的条件数。

除了提到的 $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ 矩阵，我们还选取了 13 个不同类型的病态矩阵进行数值实验，进一步对 Gram-Schmidt 算法及其变体进行数值测试，表 3 列出了 14 个测试矩阵的生成方法。

表 3 14 个测试矩阵

Table 3 14 test matrices

矩阵	矩阵生成语句
$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$	$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}},\boldsymbol{\Sigma}=diag(1,\cdots,\mu)$
Laeuchli[24]	gallery('lauchli',n,μ)
L'	A=gallery('lauchli',n,μ);$A(2,1)=1$;
Pei[25]	gallery('pei',n,μ)
Ar	$ones(n)+rand(n)*\mu$
Hilbert	hilb(n)
Invhilbert	invhilb(n)
Involutory	gallery('invol',n)
Lotkin	gallery('lotkin',n)
Frank	gallery('frank',n,k)
Prolate	gallery('prolate',n,μ)
Glued[26]	create_gluedmatrix(condA_glob,condA,m,nglued,nbglued)
R1[27]	文献[27]
R2[27]	文献[27]

5.2 算法精度测试

本节对本文提出的高精度施密特正交化算法进行精确性评估。首先采用 $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ 矩阵作为测试矩阵，测试施密特正交化算法以及双倍双精度的施密特正交化算法计算得到的正交损失，得到测试矩阵的条件数-正交性损失图，如图 1 所示，图中实线表示 MGS 算法的误差界，虚线表示 DDMGS 算法的误差界。

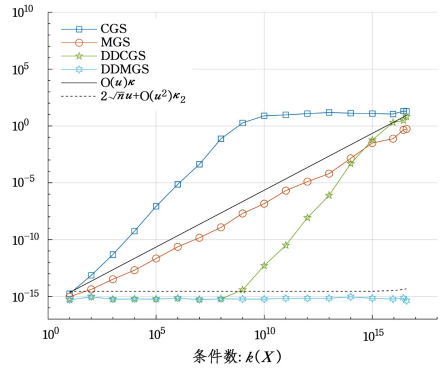

图 1 $X=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ 矩阵条件数-正交性损失图

Fig. 1 Condition number-orthogonality loss graph of $X=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$

观察图 1 发现，当条件数小于 1×10^8 时，仅有 CGS 算法与 MGS 算法产生的正交损失随着条件数的增加而变大。当条件数大于 1×10^8 时，CGS 算法的正交损失达到 9.00，数值结果变得不可靠，此时的 MGS 算法的正交性损失仍随着条件数的增大而增大，DDCGS 算法产生的正交损失也逐渐增大，甚至超过了 MGS 算法得到的正交损失。同时，只有 DDMGS 算法得到的正交性损失仍保持在 1×10^{-16} 左右。不难发现，DDMGS 算法的正交性损失保持在$[1\times10^{-16},1\times10^{-15}]$之间，且总是小于其余算法得到的正交损失。说明 MGS 算法结合双倍双精度算法能够产生正交性损失极小的矩阵 $\boldsymbol{Q}$。

随后，将本文提出的算法与 Carson 等提出的算法[11-12]进行比较，首先采用不同条件数的 $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ 矩阵作为测试矩阵，比较不同算法运行得到的正交性损失。由于 $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ 矩阵中的 $\boldsymbol{U}$ 与 $\boldsymbol{V}$ 是随机生成的正交矩阵，故图 2 与图 3 中的双倍双精度 Gram-Schmidt 算法在不同的图中曲线不一致。

图 2 与图 3 分别给出了 BMGS 算法及其变体 BMGS-SVL 算法和 BMGS-CWY 算法、BCGS 算法及其变体 BCGS-PIO 算法和 BCGS-PIP 算法、双倍双精度施密特正交化算法的比较。综合以上两图可知，DDMGS 算法产生的正交损失小于 BMGS 算法和 BCGS 算法及它们的变体，说明 DDMGS 算法的可靠性优于文献[11-12]中的施密特正交化算法。

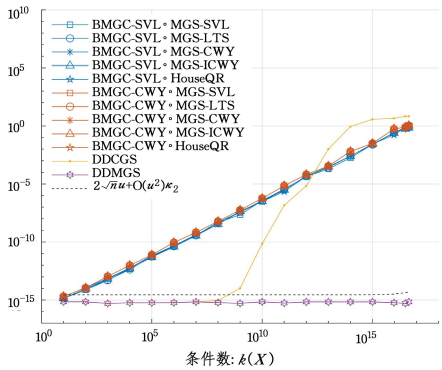

图 2 $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ 矩阵中 BMGS 算法的低同步变体 BMGS-SVL 和 BMGS-CWY 与双倍双精度算法对比

Fig. 2 Comparison of BMGS algorithm and its low-sync variants BMGS-SVL and BMGS-CWY with double-double precision algorithm in $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$

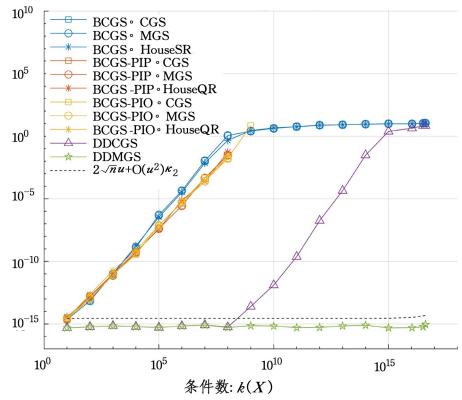

图 3 $\boldsymbol{U\Sigma V}^{\mathrm{T}}$ 矩阵中 BCGS 算法及其变体 BCGS-PIO 和 BCGS-PIP 与双倍双精度算法对比

Fig. 3 Comparison of BCGS algorithm and its variants BCGS-PIO and BCGS-PIP with double-double precision algorithm in $\boldsymbol{U\Sigma V}^{\mathrm{T}}$

采用 5.1 节中介绍的 14 个病态矩阵作为测试矩阵，测试 CGS 算法、MGS 算法、DDCGS 算法和 DDMGS 算法运算结果的正交性损失，得到的结果如图 4 所示。

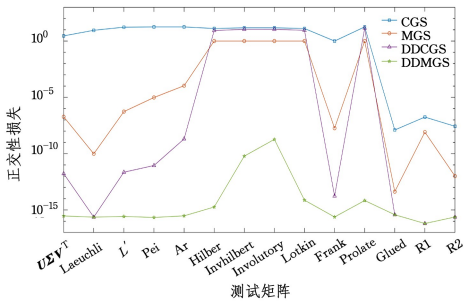

图 4 Gram-Schmidt 算法及其变体正交损失对比

Fig. 4 Comparison of orthogonal loss between Gram-Schmidt algorithm and its variants

从图 4 的结果可以看出，DDMGS 算法相比原始的算法得到的数值结果有一定程度的优化，能令正交性损失保持在 1×10^{-10} 以下。DDMGS 算法在大部分测试矩阵上的优化效果优于 DDCGS 算法，DDCGS 算法仅在 Laeuchli 矩阵、Glued 矩阵、R1 矩阵与 R2 矩阵中表现较好，可知 DDMGS 算法在几乎所有情况下都是一种有效的正交化工具。

5.3 算法性能测试

除了精确性评估外，还需进一步对本文提出的高精度 Gram-Schmidt 算法进行性能评估。首先计算 Gram-Schmidt 算法和双倍双精度 Gram-Schmidt 算法的计算复杂度(flops)，并在不同实验平台上测试以上 4 个算法处理不同规模的矩阵的运行时间。我们将加减乘除运算看作一个浮点运算，假设测试矩阵 $\boldsymbol{A}\in\boldsymbol{R}^{m\times n}$ 是列满秩的实矩阵，且 $m\geqslant n$，易得：

CGS：$2mn^2+5mn$ flops

MGS：$2mn^2+5mn$ flops

DDCGS：$43mn^2+187mn-10n^2+142n$ flops

DDMGS：$44mn^2+188mn-11n^2+141n$ flops

本文选取了 ARM 架构的 FT2000+和 X86 架构的 Intel Xeon E5-2678 v3 两个实验平台的 CPU 进行性能评估，测试施密特正交化算法及双倍双精度施密特正交化算法处理 500×500 的 Hilbert 矩阵、401×400 的 Laeuchli 矩阵以及 300×300 的 Pei 矩阵的运行时间，得到的结果如表 4 所列。由表 4 可知，CGS 与 MGS 的运行时间最短，在不同服务器上，施密特正交化算法与双倍双精度算法的运行时间比率不一致，在 5.0 倍与 16.1 倍左右。综上所述，双倍双精度算法能在一定程度上提升 Gram-Schmidt 算法的数值精度，减少其正交性损失。DDMGS 算法与 DDCGS 算法的运行效率相当，但对于本文的 14 个测试矩阵，通过 DDMGS 算法得到的矩阵 $\boldsymbol{Q}$ 的正交损失更低，其精确性更高，几乎达到了双倍双精度，因此 DDMGS 算法具有相当大的优势。

表 4 算法性能统计

Table 4 Algorithm performance statistics

(单位：s)

CPU	矩阵	CGS	MGS	DDCGS	DDMGS	$\frac{\text{DDGS}}{\text{GS}}$
Intel Xeon E5-2678 v3	Hilbert	0.64	0.57	10.37	10.55	17.29
Intel Xeon E5-2678 v3	Laeuchli	0.38	0.32	5.37	5.43	15.43
Intel Xeon E5-2678 v3	Pei	0.17	0.12	2.23	2.26	15.48
FT 2000+	Hilbert	1.08	0.84	4.03	3.88	4.12
FT 2000+	Laeuchli	0.50	0.39	2.12	2.03	4.66
FT 2000+	Pei	0.16	0.11	0.87	0.83	6.30

结束语 本文针对 Gram-Schmidt 算法的数值不稳定性问题展开研究，为降低算法的正交性损失，基于无误差变换技术，设计并实现了双倍双精度 Gram-Schmidt 算法。采用 14 个病态矩阵作为测试矩阵，通过与 Gram-Schmidt 算法以及 Carson 等提出的 Gram-Schmidt 算法进行对比，验证了双倍双精度 Gram-Schmidt 算法的精确性，且 DDMGS 算法更具可靠性。

根据 Graillat 等[28]与 Jiang 等[29]所述，补偿算法同样能够有效提升算法的数值精度，其计算结果精度与应用双倍双精度的算法近似，并且运算速度更快。下一步准备基于补偿算法思想，针对 MGS 算法以及其余存在数值不稳定性问题的算法进行优化。

参 考 文 献

[1] LEON S J, BJORCK A, GANDER W. Gram-Schmidt orthogonalization: 100 years and more[J]. Numerical Linear Algebra with Applications, 2013, 20(3): 492-532.

[2] HUANG J, GU H. QuickBird image fusion based on Gram-Schmidt transform[C]// Geographic Information and Internet of Things Forum and Proceedings of the 2010 Annual Academic Conference of the Jiangsu Society of Surveying and Mapping. 2010.

[3] ZHANG K, HSIEH M H, LIU L, et al. Quantum Gram-Schmidt processes and their application to efficient state readout for quantum talgorithms[J]. Physical Review Research, 2021, 3(4): 043095.

[4] BAILEY D H. A fortran-90 double-double library [J/OL]. https://www.davidhbailey.com/dhbsoftware, 2001.

[5] BAILEY D H, KRASNY R, PELZ R. Multiple precision, multiple processor vortex sheet roll-up computation[R]. Society for Industrial and Applied Mathematics(SIAM), Philadelphia, PA

(United States), 1993.
[6] OGITA T, RUMP S M, OISHI S. Accurate sum and dot product [J]. SIAM Journal on Scientific Computing, 2005, 26(6): 1955-1988.
[7] DEKKER T J. A floating-point technique for extending the available precision[J]. Numerische Mathematik, 1971, 18(3): 224-242.
[8] KAHAN W. Pracniques: further remarks on reducing truncation errors[J]. Communications of the ACM, 1965, 8(1): 40.
[9] KNUTH D E. The art of computer programming: Vol. 3[M]. Newyork: Pearson Education, 1997.
[10] STRANG G. Introduction to linear algebra(5th ed)[M]. Wellesley-Cambridge Press, 1993.
[11] CARSON E, LUND K, ROZLOZNIK M. The Stability of Block Variants of Classical Gram-Schmidt[J]. SIAM Journal on Matrix Analysis and Applications, 2021, 42(3): 1365-1380.
[12] CARSON E, LUND K, ROZLOZNÍK M, et al. Block Gram-Schmidt algorithms and their stability properties[J]. Linear Algebra and its Applications, 2022, 638: 150-195.
[13] SCHMIDT E. Zur Theorie der linearen und nichtlinearen Integralgleichungen. III. Teil[J]. Mathematische Annalen, 1908, 65(3): 370-399.
[14] GRAM J P. Ueber die Entwickelung reeller Functionen in Reihen mittelst der Methode der kleinsten Quadrate[J]. J Reine Angew Math, 1883, 94: 41-73.
[15] WONG Y K. An application of orthogonalization process to the theory of least squares[J]. The Annals of Mathematical Statistics, 1935, 6(2): 53-75.
[16] DE LAPLACE P S. Théorie analytique des probabilités: Vol. 7 [M]. Courcier, 1820.
[17] DANIEL J W, GRAGG W B, KAUFMAN L, et al. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization [J]. Mathematics of Computation, 1976, 30(136): 772-795.
[18] JALBY W, PHILIPPE B. Stability analysis and improvement of the block Gram-Schmidt algorithm[J]. SIAM Journal on Scientific and Statistical Computing, 1991, 12(5): 1058-1073.
[19] BARLOW J L, SMOKTUNOWICZ A. Reorthogonalized block classical Gram-Schmidt [J]. Numerische Mathematik, 2013, 123(3): 395-423.
[20] RIVAZ A, MOGHADAM M M, SADEGHI D, et al. An approach of orthogonalization within the Gram-Schmidt algorithm [J]. Computational and Applied Mathematics, 2018, 37(2): 1250-1262.
[21] Standard for binary floating point arithmetic: ANSI/IEEE standard 754-2008[S]. IEEE, 2008.
[22] BJORCK A. Numerics of gram-schmidt orthogonalization[J]. Linear Algebra and Its Applications, 1994, 197: 297-316.
[23] HIDA Y, LI X S, BAILEY D H. Algorithms for quad-double precision floating point arithmetic[C]// Proceedings 15th IEEE Symposium on Computer Arithmetic. IEEE, 2001: 155-162.
[24] LAUCHLI P. Jordan-elimination und Ausgleichung nach kleinsten Quadraten[J]. Numerische Mathematik, 1961, 3(1): 226-240.
[25] PEI M L. A test matrix for inversion procedures[J]. Communications of the ACM, 1962, 5(10): 508.
[26] SMOKTUNOWICZ A, BARLOW J L, LANGOU J. A note on the error analysis of classical Gram-Schmidt [J]. Numerische Mathematik, 2006, 105(2): 299-313.
[27] NISHI T, RUMP S M, OISHI S. On the generation of very ill-conditioned integer matrices[J]. Nonlinear Theory and Its Applications, IEICE, 2011, 2(2): 226-245.
[28] GRAILLAT S, JÉZÉQUEL F, PICOT R. Numerical validation of compensated summation algorithms with stochastic arithmetic [J]. Electronic Notes in Theoretical Computer Science, 2015, 317: 55-69.
[29] JIANG H. Study on reliable computing and rounding error analysis in floating-point arithmetic[D]. Changsha: National University of Defense Technology, 2013.

JIN Jiexi, born in 1999, postgraduate, is a member of China Computer Federation. Her main research interest is high performance computing.

JIANG Hao, born in 1983, Ph.D, professor, assoicate research fellow. His main research interests include high performance computing and numerical analysis.

（责任编辑：何杨）