Pure Mathematics 理论数学, 2020, 10(5), 425-432
Published Online May 2020 in Hans. http://www.hanspub.org/journal/pm
https://doi.org/10.12677/pm.2020.105052

Hans汉斯

A Fast Algorithm for Solving Tridiagonal Toeplitz Linear Systems with Iterative Refinement

Shan Li[1], Zhongyun Liu[1], Yulin Zhang[2]

[1]School of Mathematics and Statistics, Changsha University of Science and Technology, Changsha Hunan
[2]Centro de Matemática, Universidade do Minho, Braga, Portugal
Email: 1192928960@qq.com

Received: Apr. 16th, 2020; accepted: May 5th, 2020; published: May 12th, 2020

Abstract

This paper mainly discusses how to numerically solve tridiagonal Toeplitz linear systems $Ax = b$ efficiently. Since the coefficient matrix A has a special structure, we can design a direct algorithm to quickly solve $Ax = b$. We will apply the above algorithm to the calculation of practical examples and find that the calculation precision of some examples is not as high as that of computer's machine accuracy. In order to improve the precision of algorithm, this paper further carries out iterative refinement to quickly solve the tridiagonal Toeplitz linear equations of $Ax = b$. Numerical experiments show that the computational accuracy of our algorithm can reach computer's machine accuracy by iterative refinement [1].

Keywords

Toeplitz Matrices, Iterative Refinement, Fast Algorithm

迭代精化下求解三对角Toeplitz线性方程组的快速算法

李　姗[1]，刘仲云[1]，张育林[2]

[1]长沙理工大学数学与统计学院，湖南　长沙
[2]Minho大学数学中心，布拉加，葡萄牙
Email: 1192928960@qq.com

收稿日期：2020年4月16日；录用日期：2020年5月5日；发布日期：2020年5月12日

文章引用: 李姗, 刘仲云, 张育林. 迭代精化下求解三对角 Toeplitz 线性方程组的快速算法[J]. 理论数学, 2020, 10(5): 425-432. DOI: 10.12677/pm.2020.105052

摘　要

本文主要讨论如何对三对角Toeplitz线性方程组 $Ax=b$ 进行高精度数值求解。由于系数矩阵A这种比较特殊的结构，使得我们可以设计出快速求解 $Ax=b$ 的直接算法。我们将该算法应用到实际例子的计算过程中，发现大部分例子计算效果显著，但部分例子的计算精度还不能达到计算机机器精度。针对这类达不到计算机机器精度的例子，本文将在快速求解三对角Toeplitz线性方程组 $Ax=b$ 的直接算法基础上，进一步进行迭代精化，从而提高这类例子的计算精度。数值实验表明通过迭代精化，我们算法计算精度可以达到计算机机器精度[1]。

关键词

Toeplitz矩阵，迭代精化，快速算法

1. 引言

本文考虑迭代精化下求解三对角 Toeplitz 线性方程组

$$Ax=b, \tag{1.1}$$

$$A=\begin{bmatrix} \alpha & \gamma & & \\ \beta & \ddots & \ddots & \\ & \ddots & \ddots & \gamma \\ & & \beta & \alpha \end{bmatrix}$$

其中 $A=\mathrm{Tritoep}(\beta,\alpha,\gamma)$ 为三对角 Toeplitz 矩阵。

关于三对角 Toeplitz 线性方程组的求解方法主要分为两类：一类是直接法如高斯消元法、循环约简法和特殊 LU 分解法[2] [3] [4]等等。另一类就是迭代法如古典迭代法(Smith 迭代法，ADI 迭代法[5] [6] [7] [8]等)和投影迭代法(Krylov 子空间法等)。在不考虑舍入误差的情况下，对于小型稀疏求解线性方程组的问题，我们通常使用直接法求解。因为直接法可以通过改善置换策略，引入尽量少的填充，充分利用硬件的性能设计算法。但当遇到大型稀疏求解线性方程组的问题时，直接解法计算量太大且数值不稳定，而迭代解法数值稳定且易于并行计算，所以这时我们通常使用迭代法求解。

Toeplitz 矩阵是一类特殊结构的矩阵，它在数学、科学计算和工程中有着广泛的应用，如信号处理中的图像恢复存储问题、代数微分方程、时间序列和控制理论等。本文主要讨论如何对三对角 Toeplitz 线性方程组 $Ax=b$ 进行高精度数值求解。由于系数矩阵 A 这种比较特殊的结构，我们在前面一篇论文中已经设计出快速求解 $Ax=b$ 的直接算法。其系数矩阵主要是上次对角占优、下次对角占优、若对角占优。具体情况如下：

我们首先从系数矩阵为次对角占优情况开始考虑。

$$C=\begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ 1 & & & & 0 \end{bmatrix}$$

我们容易得到 $\hat{A}=CA$ 并且具有如下 2×2 的块结构

$$\hat{A}=\left[\begin{array}{ccccc} \beta & \alpha & \gamma & & \\ & \beta & \alpha & \ddots & \\ & & \ddots & \ddots & \gamma \\ & & & \beta & \alpha \\ \hline \alpha & \gamma & & & 0 \end{array}\right]\equiv\left[\begin{array}{c|c} A_{11} & \boldsymbol{p} \\ \hline \boldsymbol{w}^{\mathrm{T}} & 0 \end{array}\right]. \tag{1.2}$$

我们对 $\hat{A}$ 作如下的 2×2 的块 LU 分解

$$\hat{A}=\begin{bmatrix} I & \\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}_{11}^{-1} & 1 \end{bmatrix}\begin{bmatrix} A_{11} & \boldsymbol{p} \\ & -\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}_{11}^{-1}\boldsymbol{p} \end{bmatrix}. \tag{1.3}$$

$-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}_{11}^{-1}\boldsymbol{p}$ 是叫做 $\hat{A}$ 的 Schur 补。

同样地，要求问题(1.1)的解也就变成了求如下线性方程组的解

$$\hat{A}x=\hat{b} \tag{1.4}$$

其中 $\hat{b}=Cb=\left(b_2,b_3,\cdots,b_n,b_1\right)^{\mathrm{T}}$。对 x 和 $\hat{b}$ 做如下格式的划分

$$x=\begin{bmatrix} \boldsymbol{x}_1 \\ x_n \end{bmatrix},\quad \hat{b}=\begin{bmatrix} \boldsymbol{b}_2 \\ b_1 \end{bmatrix}.$$

用块 LU 分解法分解(1.3)，我们就可以通过求下面方程的解来获得方程(1.1)的解

$$\begin{cases} A_{11}\boldsymbol{x}_1+x_n\boldsymbol{p}=\boldsymbol{b}_2, \\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_1=b_1. \end{cases} \tag{1.5}$$

$$\begin{cases} x_n=\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{v}-b_1\right)\Big/\boldsymbol{w}^{\mathrm{T}}\boldsymbol{u}, \\ \boldsymbol{x}_1=\boldsymbol{v}-x_n\boldsymbol{u}. \end{cases} \tag{1.6}$$

为了获得 $\boldsymbol{x}=\left[\boldsymbol{x}_1^{\mathrm{T}}x_n\right]^{\mathrm{T}}$ 的解，我们首先求解如下方程中的 $\boldsymbol{u}$ 和 $\boldsymbol{v}$

$$\begin{cases} A_{11}\boldsymbol{u}=\boldsymbol{p}, \\ A_{11}\boldsymbol{v}=\boldsymbol{b}_2. \end{cases} \tag{1.7}$$

通过如下算法，我们得到式(1.7)中方程组的解 $\boldsymbol{u}$ 和 $\boldsymbol{v}$，具体算法如下：

算法 1 向后代入法求解 $A_{11}z=q$

1：输入 β,α,γ,q；

2：计算 $\alpha_1=\dfrac{\alpha}{\beta},\gamma_1=\dfrac{\gamma}{\beta}$；

3：计算 $z_{n-1}=\alpha_1;z_{n-2}=\gamma_1-\alpha_1 z_{n-1}$；

4：$i=2,\cdots,n-2$

计算 $z_{n-i-1}=-\alpha_1 z_{n-i}-\gamma_1 z_{n-i+1}$；

5：输出 $z=\left[z_1,z_2,\cdots,z_{n-1}\right]^{\mathrm{T}}$。

我们注意到，为了得到(1.6)的解 $\boldsymbol{x}_1$ 和 x_n，我们需要解(1.7)中的两个线性方程组，其中 A_{11} 是一个上三对角矩阵。显然，向量 $\boldsymbol{u}$ 和 $\boldsymbol{v}$ 可以通过向后代入法求解。此外，由于 A_{11} 是对角占优的，计算出的向量 $\boldsymbol{u}$ 和 $\boldsymbol{v}$ 都是稳定可靠的。

基于以上分析，我们现在可以重新设计求解(1.1)的算法如下

算法 2 一个求解三对角 Toeplitz 线性方程组 $A\boldsymbol{x}=\boldsymbol{b}$ 的算法

1：输入 $\beta,\alpha,\gamma,\boldsymbol{w},\boldsymbol{p},\boldsymbol{b}_2,b_1$；

2：调用算法 1 去解 $A_{11}\boldsymbol{u}=\boldsymbol{p}$， $A_{11}\boldsymbol{v}=\boldsymbol{b}_2$ 在(1.7)中定义；

3：计算 x_n 和 $\boldsymbol{x}_1$ 根据(1.6)；

4：输出 $\boldsymbol{x}=\left[\boldsymbol{x}_1^{\mathrm{T}}x_n\right]^{\mathrm{T}}$。

算法 2 的稳定性取决于第三步的求解 $x_n=\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{v}-b_1\right)\big/\boldsymbol{w}^{\mathrm{T}}\boldsymbol{u}$。如果 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{u}$ 不是足够小，那么 x_n 的就是相当精确的。因此，我们可以得出结论，我们求解这类线性方程组的算法在数值上是稳定的，计算出的解是可靠的，如果 A 是下次对角占优的并且 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{u}$ 不是足够小。

对于计算复杂度，我们的算法需要大约 $12n$ (flops)的浮点运算，选主元的 LU 分解法需要 13n (flops)浮点运算，我们的方法与选主元的 LU 分解法相比需要更少的浮点运算量。对于内存存储空间，我们的算法只需要存储 2 个大小为 n 向量，少于选主元的 LU 分解法需要存储 5 个大小为 n 向量。特别是，我们的算法需要较少的数据传输。在数据传输过程中它只需要读取 1 个向量即右边的向量并写入一个向量(即方程的解)。但是选主元的 LU 分解法需要读取个向量。正如我们所知，现代计算机有多层的内存结构，有不同的存储级别，如较小的高速缓存和较大的低速磁盘存储。在计算过程中，数据在不同级别的高速缓存中传输。因此，较少数据传输的算法可能会显示出更好的计算性能。这使得该算法比选主元的 LU 分解方法更有效。

现在，我们考虑上次对角占优的情况。设 J 为交换对角矩阵，在交叉对角上为 1 (从左下到右上)，其他地方为 0，即

$$J=\begin{bmatrix}0&0&\cdots&0&1\\0&0&\cdots&1&0\\\vdots&\vdots&\ddots&\vdots&\vdots\\0&1&\cdots&0&0\\1&0&\cdots&0&0\end{bmatrix},\tag{1.8}$$

因为 A 是上次对角占优的，$\tilde{A}=JAJ=\mathrm{Tritoep}\left(\gamma,\alpha,\beta\right)$ 是下次对角占优的。因此，我们可以将原来的线性线性方程组(1.1)转化为以下新的线性方程组

$$\tilde{A}\tilde{\boldsymbol{x}}=\tilde{\boldsymbol{b}}\tag{1.9}$$

$\tilde{x}=J\boldsymbol{x}$，$\tilde{\boldsymbol{b}}=J\boldsymbol{b}$。因此，结合算法 2 得到下面的算法 3 用于求解方程(1.1)。

算法 3 一个求解三对角 Toeplitz 线性方程组 $A\boldsymbol{x}=\boldsymbol{b}$ 的算法

1：输入 $\beta,\alpha,\gamma,\tilde{\boldsymbol{b}}$；

2：调用算法 2 去得到方程(1.9)的解 $\tilde{\boldsymbol{x}}$；

3：输出 $\boldsymbol{x}=J\tilde{\boldsymbol{x}}$。

由于 J 是一个置换矩阵，因此算法 3 的稳定性、计算复杂度和内存存储都与算法 2 相同。

最后，我们讨论了不可约对角占优的情形。已知，在这种情况下，A 是一个 H 矩阵，使用 LU 分解法不需要选主元。显然，这种情况下 LU 分解法不会导致任何非零元素的填充，但需要更多的内存存储空间。为了避免这个问题，我们采用以下方法：

如果 $\beta>\gamma$，那么我们选择算法 2 解方程(1.1)，如果 $\beta<\gamma$，然后调用算法 3 解解方程(1.1)。我们把以上情况总结，得到综合算法如下：

算法 4 一个求解三对角 Toeplitz 线性方程组 $A\boldsymbol{x}=\boldsymbol{b}$ 的算法

1：输入 $\beta,\alpha,\gamma,\boldsymbol{b}$ (若 $\beta>\gamma$)或 $\tilde{\boldsymbol{b}}$ (若 $\beta<\gamma$)；

2：调用算法 2 (若 $\beta>\gamma$)或调用算法 3 (若 $\beta<\gamma$)去求解方程(1.1)的解 $\boldsymbol{x}$；

3：输出 $\boldsymbol{x}$。

算法 4 的计算复杂度和内存存储与算法 2 或 3 基本相同。

与选主元 LU 分解法相比，我们提出的三对角 Toeplitz 线性方程组快速直接算法(算法 4)具有以下优点：不仅需要更少的计算时间和存储空间以及数据传输，而且具有更高的计算精度。我们将该算法应用到实际例子的计算过程中，发现大部分例子计算效果显著，但部分例子的计算精度还不能达到计算机机器精度。为解决这类问题，我们将在快速求解三对角 Toeplitz 线性方程组 $Ax=b$ 的直接算法基础上进行迭代精化(详情见算法 5)，从而减小我们的计算误差，提高解的精度。

2. 迭代精化算法及收敛性分析

求解(1.1)的精化迭代法的迭代格式[12]为：

$$x^{(k+1)}=x^{(k)}+d^{(k)} \tag{2.1}$$

$$Ad^{(k)}=r^{(k)},r^{(k)}=b-Ax^{(k)} \tag{2.2}$$

其中 $x^{(0)}$ 为初始迭代向量，$k=0,1,\cdots$。当 $d^{(k)}$ 为(2.2)的精确解时，迭代格式(2.1)~(2.2)退化为单步迭代精化。不难看出，该迭代格式可以提高解 $x^{(k)}$ 的精度。另外，如果我们用高精度方法求解(2.2)，那么我们把迭代格式(2.1)~(2.2)称为迭代精化。

引理 2.1 [12]假设不考虑舍入误差，精确计算残差 $r^{(k)}$，$d^{(k)}$ 为(2.2)的精确解，则 $x^{(k+1)}$ 为(1.1)的精确解[9]。

证明：由(2.2)式知 $Ad^{(k)}=r^{(k)}$，$r^{(k)}=b-Ax^{(k)}$，从而 $Ax^{(k+1)}=Ax^{(k)}+Ad^{(k)}=b$。这意味着 $x^{(k+1)}$ 是(1.1)的精确解，证毕。

结合引理 2.1 与算法 4，我们得到迭代精化[10]的具体算法如下：

算法 5 迭代精化求解三对角 Toeplitz 线性方程组 $Ax=b$

1：输入迭代步数 $Iter=10$，最大迭代步数 $maxiter=1000$，$\eta=\dfrac{\left\|r^{(k)}-Ad^{(k)}\right\|}{\left\|r^{(k)}\right\|}$；

2：用算法 4 求解线性方程组 $Ax=b$，得到初始解 x，令 $x^{(0)}=x$ ($x^{(0)}$ 为初始迭代向量)；

3：计算 $r^{(k)}=b-Ax^{(k)}$，用算法 4 解线性方程 $Ad^{(k)}=r^{(k)}$ 得到 $d^{(k)}$；

4：更新 $x^{(k+1)}=x^{(k)}+d^{(k)}$，$k=k+1$；

5：若满足 $Iter\le maxiter$ 且 $\eta\ge 1e-15$，转第 3 步；否则，转第 6 步；

6：输出 $x^{(k+1)}$，η。

定理 2.1 [12] 假设 r 是双精度并且 $\kappa(A)\cdot\varepsilon < c \equiv \frac{1}{3n^3 g+1} < 1$，$n$ 表示矩阵 A 的阶数，g 表示主元增长因子，$\kappa(A)=\|A\|\cdot\|A^{-1}\|$。那么迭代精化收敛于：

$$\frac{\|x^i - A^{-1}b\|}{\|A^{-1}b\|} = o(\varepsilon). \tag{2.3}$$

定理 2.2 [12] 令 x^* 为线性方程组(1.1)的精确解。对第 k 步迭代，用算法 5 求解(2.2)的初始迭代向量均为 $d^{(0)}=x^{(0)}$ (其中 $x^{(0)}$ 为用算法 4 求得的线性方程组(1.1)的解)，相应的迭代序列为 $d^{(k)}$，且 $d^{(k)}$ 满足终止检验公式

$$\eta = \frac{\|r^{(k)} - Ad^{(k)}\|}{\|r^{(k)}\|} < \varepsilon \tag{2.4}$$

则迭代序列 $x^{(k)}$ 满足

$$\|x^{(k+1)} - x^*\| \le \kappa(A)\varepsilon\|x^{(k)} - x^*\| \tag{2.5}$$

特别地，当 $\kappa(A)\varepsilon < 1$，迭代序列 $x^{(k)}$ 收敛到 $x^{(*)}$ [11]。

证明： 由(2.1)、(2.2)及(2.4)式知

$$x^{(k+1)} - x^* = x^{(k)} - x^* + d^k = A^{-1}\left[Ad^k - r^k\right] \tag{2.6}$$

从而

$$\begin{aligned}\|x^{(k+1)} - x^*\| &\le \|A^{-1}\|\|Ad^k - r^k\| \le \varepsilon\|A^{-1}\|\|r^k\| \le \varepsilon\|A^{-1}\|\|A\|\|x^{(k)} - x^*\| \\ &= \varepsilon\kappa(A)\|x^{(k)} - x^*\| < \|x^{(k)} - x^*\|\end{aligned} \tag{2.7}$$

当 $\kappa(A)\varepsilon < 1$时，迭代序列 x^k 收敛，证毕。

针对算法 5，只要 $\varepsilon < 1e-15$，则 $x^{(k+1)}$ 满足终止检验公式(2.4)，那么 $x^{(k+1)}$ 就是满足计算机机器精度的解[12]。

3. 数值实验

下面我们用一些例子证明我们算法的有效性。本文所有数值实验的电脑运行环境为：Intel(R) Core(TM) i3-2310M CPU @2.10 by Matlab 7.4.0.287 (R2014a)。分别用快速直接算法和加迭代精化的快速直接法分别对三对角 Toeplitz 线性方程组进行求解。选取典型代表例子如下：

例 1：$A = \text{Tritoep}(-1-c, 2, -1+c)$；

例 2：$A = \text{Tritoep}(-1-c, 2+c, -1)$。

在例 1、例 2 中的矩阵 A 是由一维对流扩散方程离散差分得到的方程的系数矩阵。在所有例子中二阶差分都采用中心差分格式。但在例 1 和例 2 中一阶差分分别使用中心差分格式和向后差分格式。在数值实验中，我们计算了许多不同 n 和参数 c 的例子。我们可以得到共同的结论。接下来我们展示一些有代表性的例子结果。在实验中，n 表示矩阵 A 的阶数，*Iter* 表示迭代次数，CPU 表示计算时间，右端向量 $\boldsymbol{b}=A\boldsymbol{e}$，$R$ 表示计算相对残差，即($R=\frac{\|\boldsymbol{b}-A\boldsymbol{x}\|}{\|\boldsymbol{b}\|}$)，Fast algorithm 表示快速算法，Iter algorithm 表示对快速算法进行迭代精化。

数值结果表 1 是例 1 取 $c=0.1$ 和 $c=0.7$ 的两个例子，数值结果表 2 是例 2 取 $c=0.1$ 和 $c=0.2$ 的两个例子。我们发现当我们用直接算法 1 计算三对角 Toeplitz 线性方程组 $Ax=b$ 时，存在计算精度无法达到机器精度的情况。但是通过我们进一步迭代精化，当迭代一定次数(本例 Iter = 10)时，我们的计算精度就可以达到计算机机器精度。

Table 1. $A=\text{Tritoep}(-1-c,2,-1+c)$

表 1. $A=\text{Tritoep}(-1-c,2,-1+c)$

	$b=Ae(c=0.1)$				$b=Ae(c=0.7)$			
n	Fast algorithm		Iter algorithm		Fast algorithm		Iter algorithm	
	CPU	R	CPU	R	CPU	R	CPU	R
2^{19}	2.155e−2	4.612e−13	4.906e−1	6.614e−16	2.381e−2	1.021e−13	3.987e−1	7.111e−16
2^{20}	4.449e−2	6.523e−13	9.057e−1	9.353e−16	4.512e−2	1.444e−13	8.595e−1	9.944e−16
2^{21}	9.401e−2	9.225e−13	1.827e+0	1.323e−16	9.031e−2	2.042e−13	1.551e+0	1.463e−16

Table 2. $A=\text{Tritoep}(-1-c,2+c,-1)$

表 2. $A=\text{Tritoep}(-1-c,2+c,-1)$

	$b=Ae(c=0.1)$				$b=Ae(c=0.2)$			
n	Fast algorithm		Iter algorithm		Fast algorithm		Iter algorithm	
	CPU	R	CPU	R	CPU	R	CPU	R
2^{19}	2.113e−2	1.081e−12	4.298e−1	1.072e−16	2.237e−2	7.241e−13	4.285e−1	1.104e−16
2^{20}	4.152e−2	1.439e−12	8.010e−1	1.797e−16	4.132e−2	1.024e−12	7.914e−1	1.561e−16
2^{21}	8.451e−2	2.036e−12	1.550e+0	2.219e−16	8.942e−2	1.448e−12	1.539e+0	2.371e−16

数值实验表明，当采用直接算法求解三对角 Toeplitz 线性方程组的时候，存在计算精度达不到计算机机器精度的情况。但是通过我们的迭代精化算法，可以使三对角 Toeplitz 线性方程组的解都能够达到计算机机器精度。

基金项目

2019 年硕士研究生校级科研创新项目(CX2019SS34)。

参考文献

[1] Saad, Y. (2000) Iterative Methods for Sparse Linear Systems. 2nd Edition, SIAM, Philadelphia, PA.

[2] Yan, W.-M. and Chung, K.-L. (1994) A Fast Algorithm for Solving Special Tridiagonal Systems. *Computing*, **52**, 203-211. https://doi.org/10.1007/bf02238076

[3] Garey, L.E. and Shaw, R.E. (2001) A Parallel Method for Linear Equations with Tridiagonal Toeplitz Coefficient Matrices. *Computer & Mathematics with Applications*, **42**, 1-11. https://doi.org/10.1016/s0898-1221(01)00125-0

[4] Kim, H.J. (1990) A Parallel Algorithm Solving a Tridiagonal Toeplitz Linear System. *Parallel Computing*, **13**, 289-294. https://doi.org/10.1016/0167-8191(90)90131-r

[5] Benner, P., Li, R.C. and Truhar, N. (2009) On the ADI Method for Sylvester Equations. *Journal of Computational and Applied Mathematics*, **233**, 1035-1045. https://doi.org/10.1016/j.cam.2009.08.108

[6] Kurschner, P., Benner, P. and Saak, J. (2014) Self-Generating and Efficient Shift Parameters in ADI Methods for Large

Lyapunov and Sylvester Equations. *Electronic Transactions on Numerical Analysis*, **43**, 142-162.

[7] Levenberg, N. and Reichel, L. (1993) A Generalized ADI Iterative Method. *Numerische Mathematik*, **66**, 215-233. https://doi.org/10.1007/bf01385695

[8] Lu, A. and Wachspress, E.L. (1991) Solution of Lyapunov Equations by Alternating Direction Implicit Iteration. *Computers and Mathematics with Applications*, **21**, 43-58. https://doi.org/10.1016/0898-1221(91)90124-m

[9] Horn, R.A. and Johnson, C.R. (1985) Matrix Analysis. Cambridge University Press, Cambridge.

[10] Bai, Z.-Z., Golub, G.H. and Ng, M.K. (2003) Hermitian and Skew-Hermitian Splitting Methods for Non-Hermitian Positive Definite Linear Systems. *SIAM Journal on Matrix Analysis and Applications*, **24**, 603-626. https://doi.org/10.1137/s0895479801395458

[11] Garey, L.E., Majedi, M. and Shaw, R.E. (2008) A Parallel Sewing Method for Solving Tridiagonal Toeplitz Strictly Diagonally Dominant Systems. I.P.D.P.S., 1-8. https://doi.org/10.1109/ipdps.2008.4536466

[12] Demmel, J.W. 应用数值线性代数[M]. 王国荣, 译. 北京: 人民邮电出版社, 2007.