Tridiagonal Matrix Algorithm

Tamas Kis | tamas.a.kis@outlook.com | https://tamaskis.github.io

CONTENTS

1 TRIDIAGONAL MATRIX ALGORITHM

1.1 Tridiagonal Linear Systems

A **tridiagonal linear system** is one of the form

$$\boxed{\mathbf{A}\mathbf{x} = \mathbf{d}} \tag{1}$$

where

$$\boxed{\underbrace{\begin{bmatrix} b_1 & c_1 & & & & \\ a_1 & b_2 & c_2 & & & \\ & a_2 & \ddots & \ddots & & \\ & & \ddots & \ddots & c_{n-2} & \\ & & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & & a_{n-1} & b_n \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}}_{\mathbf{d}}} \tag{2}$$

and where $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{x}, \mathbf{d} \in \mathbb{R}^n$. Owing to the fact that it only has three nonzero diagonals, the matrix $\mathbf{A}$ is referred to as a **tridiagonal matrix**[1]. The **tridiagonal vectors** $\mathbf{a} \in \mathbb{R}^{n-1}$, $\mathbf{b} \in \mathbb{R}^n$, and $\mathbf{c} \in \mathbb{R}^{n-1}$ are defined below in Eq. (3).

$$\boxed{\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_{n-1} \end{bmatrix}} \tag{3}$$

These tridiagonal vectors form the tridiagonal matrix $\mathbf{A}$, as shown in Eq. (1) [1].

The **tridiagonal matrix algorithm** (also known as the **Thomas algorithm**) is an algorithm that can efficiently solve the tridiagonal linear system for $\mathbf{x}$. There are two general implementations of this algorithm; one whose inputs are the tridiagonal vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, and the other which operates directly on the tridiagonal matrix $\mathbf{A}$. We name these algorithms accordingly:

Algorithm	**Reason for Name**
`tridiagonal_vector` (Algorithm 1)	The tridiagonal *vectors*, $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, are input to this function.
`tridiagonal_matrix` (Algorithm 4)	The tridiagonal *matrix*, $\mathbf{A}$, is input to this function.

Two additional algorithms (Algorithms 2 and 3) are also detailed, but these primarily serve as stepping stones towards developing Algorithm 4.

1.2 Tridiagonal Matrix Algorithm: Vector Implementation

The tridiagonal matrix algorithm defined by Algorithm 1 below is adapted from [1, 3, 4].

[1] In many references, a tridiagonal matrix is often defined with one of the following two convention:

$$\mathbf{A} = \begin{bmatrix} a_1 & b_1 & & & & \\ c_1 & a_2 & b_2 & & & \\ & c_2 & \ddots & \ddots & & \\ & & \ddots & \ddots & b_{n-2} & \\ & & & c_{n-2} & a_{n-1} & b_{n-1} \\ & & & & c_{n-1} & a_n \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & \ddots & \ddots & & \\ & & \ddots & \ddots & c_{n-2} & \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{bmatrix}$$

The first is typically used when defining a tridiagonal linear system [2], while the second is used almost exclusively when defining the tridiagonal matrix algorithm [3, 4]. However, for the second convention above, the a_i's range from a_2 to a_n, which is inconvenient from a programming standpoint; therefore, I defined them here as ranging from a_1 to a_{n-1}. This convention is also reflected in Algorithms 1, 2, and 3.

Algorithm 1: `tridiagonal_vector`

Solves the tridiagonal linear system $\mathbf{A}\mathbf{x} = \mathbf{d}$ for $\mathbf{x}$ using the vector implementation of the tridiagonal matrix algorithm.

Inputs:

- $\mathbf{a} \in \mathbb{R}^{n-1}$ - tridiagonal vector
- $\mathbf{b} \in \mathbb{R}^{n}$ - tridiagonal vector
- $\mathbf{c} \in \mathbb{R}^{n-1}$ - tridiagonal vector
- $\mathbf{d} \in \mathbb{R}^{n}$ - vector

Procedure:

1. Determine n, given that $\mathbf{d} \in \mathbb{R}^n$.
2. Preallocate the vector $\mathbf{x} \in \mathbb{R}^n$.
3. Forward elimination.

for $i = 2$ **to** n

$$w = \frac{a_{i-1}}{b_{i-1}}$$
$$b_i = b_i - wc_{i-1}$$
$$d_i = d_i - wd_{i-1}$$

end

4. Backward substitution.

$$x_n = \frac{d_n}{b_n}$$

for $i = n-1$ **to** 1 **by** -1

$$x_i = \frac{d_i - c_i x_{i+1}}{b_i}$$

end

5. Solution of tridiagonal linear system.

return $\mathbf{x}$

Outputs:

- $\mathbf{x} \in \mathbb{R}^n$ - solution of the tridiagonal linear system $\mathbf{A}\mathbf{x} = \mathbf{d}$

Note:

- The tridiagonal matrix ($\mathbf{A}$) for the tridiagonal linear system ($\mathbf{A}\mathbf{x} = \mathbf{d}$) is defined in terms of the tridiagonal vectors ($\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$) as

$$\mathbf{A} = \begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & \ddots & \ddots & & \\ & & \ddots & \ddots & c_{n-2} & \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{bmatrix}$$

1.3 Tridiagonal Matrix Algorithm: Matrix Implementation

The tridiagonal matrix algorithm essentially takes the tridiagonal vector algorithm from Section 1.2 and adapts it to the case where we input the tridiagonal matrix ($\mathbf{A}$) instead of the tridiagonal vectors ($\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$). The algorithms presented in Sections 1.3.1 and 1.3.2 are stepping stones towards the shortest implementation (i.e. the own that should actually be implemented in code) presented in Section 1.3.3.

1.3.1 Naive Version

The implementation of the tridiagonal matrix algorithm provided by Algorithm 2 is a rather naive one where we extract the tridiagonal vectors ($\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$) one-by-one from the tridiagonal matrix $\mathbf{A}$.

Algorithm 2: `tridiagonal_matrix_naive`

Solves the tridiagonal linear system $\mathbf{A}\mathbf{x} = \mathbf{d}$ for $\mathbf{x}$ using the matrix implementation of the tridiagonal matrix algorithm (naive version).

Inputs:

- $\mathbf{A} \in \mathbb{R}^{n \times n}$ - tridiagonal matrix
- $\mathbf{d} \in \mathbb{R}^n$ - vector

Procedure:

1. Determine n, given that $\mathbf{d} \in \mathbb{R}^n$.
2. Preallocate the vectors $\mathbf{a} \in \mathbb{R}^{n-1}$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^{n-1}$, and $\mathbf{x} \in \mathbb{R}^n$.
3. Extract $\mathbf{a}$ from $\mathbf{A}$.

   for $i = 2$ **to** n
   $$a_{i-1} = A_{i,i-1}$$
   end

4. Extract $\mathbf{b}$ from $\mathbf{A}$.

   for $i = 1$ **to** n
   $$b_i = A_{i,i}$$
   end

5. Extract $\mathbf{c}$ from $\mathbf{A}$.

   for $i = 2$ **to** n
   $$c_{i-1} = A_{i-1,i}$$
   end

6. Forward elimination.

   for $i = 2$ **to** n
   $$w = \frac{a_{i-1}}{b_{i-1}}$$
   $$b_i = b_i - wc_{i-1}$$
   $$d_i = d_i - wd_{i-1}$$
   end

7. Backward substitution.

$$x_n = \frac{d_n}{b_n}$$

for $i = n-1$ **to** 1 **by** -1

$$x_i = \frac{d_i - c_i x_{i+1}}{b_i}$$

end

8. Solution of tridiagonal linear system.

return $\mathbf{x}$

Outputs:

- $\mathbf{x} \in \mathbb{R}^n$ - solution of the tridiagonal linear system $\mathbf{A}\mathbf{x} = \mathbf{d}$

1.3.2 Better Version

We can save some computational effort by reducing the number of for loops in Algorithm 2. For smaller systems, this doesn't make a huge impact, but for larger systems, it can halve the time it takes to solve. We can note that four of the loops go "forward" in i, so we can combine them (with the caveat that we must extract b_1 separately since its loop starts from 1 and not 2). Defining this "better" algorithm,

Algorithm 3: `tridiagonal_matrix_better`

Solves the tridiagonal linear system $\mathbf{A}\mathbf{x} = \mathbf{d}$ for $\mathbf{x}$ using the matrix implementation of the tridiagonal matrix algorithm (better version).

Inputs:

- $\mathbf{A} \in \mathbb{R}^{n\times n}$ - tridiagonal matrix
- $\mathbf{d} \in \mathbb{R}^n$ - vector

Procedure:

1. Determine n, given that $\mathbf{d} \in \mathbb{R}^n$.
2. Preallocate the vectors $\mathbf{a} \in \mathbb{R}^{n-1}$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^{n-1}$, and $\mathbf{x} \in \mathbb{R}^n$.
3. Extract first element of $\mathbf{b}$ from $\mathbf{A}$.

$$b_1 = A_{1,1}$$

4. Forward loop.

for $i = 2$ **to** n

(a) Extract relevant elements of $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ from $\mathbf{A}$.

$$a_{i-1} = A_{i,i-1}$$
$$b_i = A_{i,i}$$
$$c_{i-1} = A_{i-1,i}$$

(b) Forward elimination.

$$w = \frac{a_{i-1}}{b_{i-1}}$$
$$b_i = b_i - wc_{i-1}$$
$$d_i = d_i - wd_{i-1}$$

end

5. Backward loop (backward substitution).

$$x_n = \frac{d_n}{b_n}$$

for $i = n-1$ **to** 1 **by** -1

$$x_i = \frac{d_i - c_i x_{i+1}}{b_i}$$

end

6. Solution of tridiagonal linear system.

return $\mathbf{x}$

Outputs:

- $\mathbf{x} \in \mathbb{R}^n$ - solution of the tridiagonal linear system $\mathbf{Ax} = \mathbf{d}$

1.3.3 Best Version

Finally, instead of defining/preallocating the vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, we can access the elements of $\mathbf{A}$ directly. We refer to this as the "best" implementation; it is best in the sense that it requires the least lines of code. It is also generally faster than the implementation presented in Section 1.3.2.

Algorithm 4: `tridiagonal_matrix`

Solves the tridiagonal linear system $\mathbf{Ax} = \mathbf{d}$ for $\mathbf{x}$ using the matrix implementation of the tridiagonal matrix algorithm.

Inputs:

- $\mathbf{A} \in \mathbb{R}^{n\times n}$ - tridiagonal matrix
- $\mathbf{d} \in \mathbb{R}^n$ - vector

Procedure:

1. Determine n, given that $\mathbf{d} \in \mathbb{R}^n$.
2. Preallocate the vector $\mathbf{x} \in \mathbb{R}^n$.
3. Forward elimination.

for $i = 2$ **to** n

$$w = \frac{A_{i,i-1}}{A_{i-1,i-1}}$$
$$A_{i,i} = A_{i,i} - wA_{i-1,i}$$
$$d_i = d_i - wd_{i-1}$$

end

4. Backward substitution.

$$x_n = \frac{d_n}{A_{n,n}}$$

for $i = n-1$ **to** 1 **by** -1

$$x_i = \frac{d_i - A_{i,i+1}x_{i+1}}{A_{i,i}}$$

end

5. Solution of tridiagonal linear system.

return $\mathbf{x}$

Outputs:

- $\mathbf{x} \in \mathbb{R}^n$ - solution of the tridiagonal linear system $\mathbf{Ax} = \mathbf{d}$

REFERENCES

[1] James Hateley. *Linear Systems of Equations and Direct Solvers*. MATH 3620 Course Reader (Vanderbilt University). 2019.

[2] *Tridiagonal matrix algorithm*. Wikipedia. Accessed: December 14, 2021. URL: https://en.wikipedia.org/wiki/Tridiagonal_matrix.

[3] *Tridiagonal matrix algorithm*. Wikipedia. Accessed: January 9, 2021. URL: https://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm.

[4] *Tridiagonal matrix algorithm – TDMA (Thomas algorithm)*. CFD Online. Accessed: January 9, 2021. URL: https://www.cfd-online.com/Wiki/Tridiagonal_matrix_algorithm_-_TDMA_(Thomas_algorithm).