SOLVING A CLASS OF BOUNDARY VALUE PROBLEMS BY LSQR

by

Yu-Yang QIU*

College of Statistics and Mathematics, Zhejiang Gongshang University, Hangzhou, China

Original scientific paper
https://doi.org/10.2298/TSCI160715059Q

Boundary value problems arising in fluid mechanics and thermal science can be transformed uniformly to a set of linear equations, whose coefficient matrix is circulant. This paper adopts a matrix iteration LSQR to solve the inverse of coefficient matrix. The solution process is elucidated step by step, and the numerical results reveal the effectiveness and feasibility of the presented method.

Key words: boundary value problems, circulant, iteration LSQR, inverse

Introduction

Some boundary value problems arising in fluid mechanics and thermal science problems can be written as the following form [1-6]:

\[ \begin{cases} L u(x,y) = f(x,y), (x,y) \in \Omega, \\ B u(x,y) = g(x,y), (x,y) \in \partial \Omega \end{cases} \] (1)

where \( L \) is a linear differential operator, \( B \) – a boundary operator, \( f(x,y) \) and \( g(x,y) \) are two known functions, and \( \Omega \in \mathbb{R}^2 \) is an open bounded domain with boundary \( \partial \Omega \). By the method of fundamental solutions, problem (1) can be transformed uniformly to a set of linear equations \( C_n z = b \) with unknown vector \( z \), the circulant or block-circulant matrix \( C_n \), and the given vector \( b \) [1]. So the key to the problem (1) is to solve the inverse of the circulant matrix, \( C_n \).

Generally, we should add the additional conditions such as positive definite on \( C_n \) to get its inverse, and fast Fourier transformation is necessary [7-11]. However, it may not maintain workable as \( n \) gets bigger.

In this paper, we adopt an iteration algorithm LSQR to solve the inverse of coefficient matrix [12]. Our ideas are based on the following observations: 1 – since the inverse of the circulant matrix \( C_n \) is circulant [7], it can be mapped to the independent parameter vector with its special structure, 2 – the inverse of the circulant matrix can be transformed to solve a linear equation, and it will be obtained by vector form iteration LSQR involved Kronecker product, and 3 – the LSQR only involves matrix-vector product, however, it has Kronecker product which will increase the computational complexity. So we intend to release Kronecker product and get the corresponding matrix form iteration. In this paper, the solution process is elucidated step by step and numerical results reveal the effectiveness and feasibility of the presented method.

* Corresponding author, e-mail: yuyangqiu77@163.com
Notation. In the rest of paper, $R^{m \times n}$ denotes the space of real $m \times n$ matrix. For any matrix $X = \begin{bmatrix} x_1, x_2, \ldots, x_n \end{bmatrix} \in R^{m \times n}$, $X^T$ stands for its transpose, and the long vector expanded by columns of $X$ is denoted by $\text{vec}(X) = [x_1^T, x_2^T, \ldots, x_n^T]^T$. For any vector $v \in R^n$, $v(i)$ is its $i^{th}$ component. The norm $\| \cdot \|_F$ is the Frobenius norm of matrix, while $\| \cdot \|_2$ is 2-norm of vector or matrix. The notation $\otimes$ is Kronecker product.

The inverse of circulant matrix and LSQR involved Kronecker product

The circulant matrix from the problem (1) has the following form:

$$ C_n = \begin{pmatrix} c_0 & c_1 & \cdots & c_{n-2} & c_{n-1} \\ c_{n-1} & c_0 & \cdots & c_{n-3} & c_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_2 & c_3 & \cdots & c_0 & c_1 \\ c_1 & c_2 & \cdots & c_{n-1} & c_0 \end{pmatrix} $$

where $c_i$, $i = 0, 1, \ldots, n-1$ are parameters. Suppose:

$$ G = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix} $$

then

$$ C_n = c_0 G^0 + c_1 G + \cdots + c_{n-1} G^{n-1} \quad (2) $$

where

$$ G^0 = I_n $$

Note that $C_n$ is a circulant matrix, its inverse matrix $C_{\text{inv}}$ is also circulant [7]. So $C_{\text{inv}}$ can be rewritten:

$$ C_{\text{inv}} = h_0 G^0 + h_1 G + \cdots + h_{n-1} G^{n-1} \quad (3) $$

where $h_i = 0, 1, \ldots, n-1$ are the parameters as in eq. (2). Hence:

$$ \text{vec} C_{\text{inv}} = C_T h \quad (4) $$

where

$$ C_T = [\text{vec}(G^0), \text{vec}(G^1), \cdots, \text{vec}(G^{n-1})] $$

and

$$ h = (h_0, h_1, \ldots, h_{n-1})^T $$

Obviously, the inverse circulant matrix $C_{\text{inv}}$ can be represented by the unknown vector $h$ with (4). Therefore:

$$ C_n C_{\text{inv}} = I_n $$
is equivalent to:

$$Ah = b$$ \hspace{1cm} (5)

with

$$A = (I_n \otimes C_n) C_T \quad \text{and} \quad b = \text{vec}(I_n)$$

That is, we can solve eq. (5) to get the vector $h$ and reconstruct the inverse matrix $C_{inv}$ by eq. (3). So the key to $C_{inv}$ is the solution of eq. (5). We introduce the iteration LSQR to solve eq. (5), since it has good numerical performance [7].

**Iteration LSQR**

1 – Initialization.

Set $h = h_{init}$, $r = b - Ah_{init}$.

$$\beta_0 = \| r \|_2, \quad u = \frac{r}{\beta_0}, \quad v = A^T u, \quad \alpha_1 = \| v \|_2, \quad v = \frac{v}{\alpha_1}, \quad \overline{\phi}_1 = \beta_0, \quad \overline{\alpha}_1 = \alpha_1, \quad w = v$$

2 – Iteration. For $i = 1, 2, \ldots$ until convergence.

2.1 – Compute $u$.

$$u = Av - \alpha_i u, \quad \beta_i = \| u \|_2, \quad u = \frac{u}{\beta_i}$$

2.2 – Compute Givens rotation $G_i = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$.

if $| \overline{\alpha}_i | \geq | \beta_i |$, then $t = \frac{\beta_i}{\overline{\alpha}_i}$, $\eta_i = \sqrt{1 + t^2}$, $c = \frac{1}{\eta_i}$, $s = tc$, $\gamma_i = \eta_i \alpha_i$,

else $t = \frac{\overline{\alpha}_i}{\beta_i}$, $\eta_i = \sqrt{1 + t^2}$, $s = \frac{1}{\eta_i}$, $c = ts$, $\gamma_i = \eta_i \beta_i$

2.3 – Compute $v$.

$$v = A^T u - \beta_i v, \quad \alpha_{i+1} = \| v \|_2, \quad v = \frac{v}{\alpha_{i+1}}$$

2.4 – Update $\overline{\alpha}_{i+1}$ and $\overline{\phi}_i$.

$$\xi_i = s \alpha_{i+1}, \quad \overline{\alpha}_{i+1} = c \alpha_{i+1}, \quad \overline{\phi}_{i+1} = -s \overline{\phi}_i, \quad \phi_i = c \overline{\phi}_i$$

2.5 – Update $h$ and $w$.

$$h = h + \frac{\phi_i}{\gamma_i} w, \quad w = v - \frac{\xi_i}{\gamma_i} w$$

2.6 – Check convergence $| \alpha_{i+1} c_i / \| A \|_F | \leq \tau$ with the given number $\tau$.

Applying iteration LSQR on eq. (5), we get the corresponding algorithm LSQR_v (see [7] for details).
Iteration LSQR without Kronecker product

The two basic operations in iteration LSQR_v are $Av$ and $A^T u$, where $v \in \mathbb{R}^n$, $u \in \mathbb{R}^n$. They will be very big when $n$ increases, since the matrix $A$ has Kronecker product, which will increase the computational complexity. So we intend to release it and get the corresponding matrix form iteration.

For the vector $v$ there exists a circulant matrix $V \in \mathbb{R}^{n \times n}$ such that:

$$vec(V) = C^T v$$

where

$$V = \sum_{i=1}^{n} v(i) G_i^{i-1}$$

so

$$Av = (I \otimes C_n) C_T v = (I \otimes C_n) vec(V) = vec(C_n V)$$

Now we want to correspond the vector $A^T u$ to the matrix $P_C(A^T u)$ in circulant matrix space. For this end, we should find the matrix $U$ such that $U = vec(U)$, and represent $g_C(U)$ which is the co-ordinate of $U$. It is not difficult to verify that:

$$P_C[(C_T)^T u] = \sum_{i=0}^{n-1} trace(U^T G_i^j) G_i^j$$

and

$$g_C(U)_j = \frac{trace(U^T G_i^j)}{n}, \quad j = 0, 1, \ldots, n - 1$$

Set

$$\bar{U} = (C_n)^T U$$

then

$$P_C[A^T vec(U)] = P_C[C_T vec(\bar{U})] = \sum_{i=0}^{n-1} trace(\bar{U}^T G_i^j) G_i^j$$

So algorithm LSQR_v can be rewritten as its matrix form iteration LSQR_M.

**Iteration LSQR_M**

1 – Initialization.

Set

$$H = H_0, \quad R = I_n - C_n H_0, \quad R_0 = ||R||_F, \quad U = \frac{R}{\beta_0}, \quad \bar{U} = C_n^T U,$$

$$V_T = P_C[C_T^T vec(\bar{U})], \quad \alpha_t = ||g_C(V_t)||_2, \quad V = \frac{V_t}{\alpha_t}, \quad \bar{\alpha_t} = \beta_0,$$

$$\bar{\alpha_t} = \alpha_t, \quad W = V, \quad N_T = 0$$
2 – Iteration. For \( i = 1, 2, \cdots \) until convergence.

2.1 – Compute \( U \).

\[ U = C_n V - \alpha_i U, \quad \beta_i = \| U \|_F, \quad U = \frac{U}{\beta_i} \]

2.2 – Compute Givens rotation \( G_i = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \).

if \( |\alpha_i| \geq |\beta_i| \),

\[ t = \frac{\beta_i}{\alpha_i}, \quad \eta_i = \sqrt{1 + t^2}, \quad c = \frac{1}{\eta_i}, \quad s = tc, \quad \gamma_i = \eta_i \alpha_i, \]

else \( t = \frac{\alpha_i}{\beta_i}, \quad \eta_i = \sqrt{1 + t^2}, \quad s = \frac{1}{\eta_i}, \quad c = ts, \quad \gamma_i = \eta_i \beta_i \)

2.3 – Compute:

\[ \tilde{U} = C_n^T U, \quad V_i = P_c[C_T \text{vec}(\tilde{U})], \quad \bar{V} = V_i - \beta_i V, \quad \alpha_{i+1} = ||g_c(V_i)||_2, \]

\[ V = \frac{\bar{V}}{\alpha_{i+1}}, \quad \bar{N}_i = N_i + \alpha_i^2 + \beta_i^2 \]

2.4 – Update \( \alpha_{i+1} \) and \( \phi_i \).

\[ \xi_i = s \alpha_{i+1}, \quad \bar{\alpha}_{i+1} = c \alpha_{i+1}, \quad \bar{\phi}_{i+1} = -s \bar{\phi}_i, \quad \phi_i = c \bar{\phi}_i \]

2.5 – Update \( H \) and \( W \).

\[ H = H + \begin{pmatrix} \phi_i \\ \bar{N}_i \end{pmatrix} W, \quad W = V - \begin{pmatrix} \xi_i \\ \bar{V}_i \end{pmatrix} W \]

2.6 – Check convergence \( |\alpha_{k+1}| \leq (N)_{1/2} \) with the given number \( r \).

In iteration LSQR\_M, the Kronecker product has been released, and the iteration only involve the matrix-matrix and matrix-vector product.

**Numerical example**

In this section, we test our iteration LSQR\_M. The matrix size, \( n \), varies from \( n = 20 \) to \( n = 500 \), the circulant matrix \( C_n \) is constructed:

\[ c = \text{rand}(n,1), \quad C_n = \sum_{i=1}^{n} c(i) G^{i-1} \]

where \( G \) is set by:

\[ e = \text{ones}(n,1), \quad Me = e e^T \]

\[ G = \text{diag}[\text{diag}(Me,1,1)] + \text{diag}[\text{diag}(Me,n-1,1-n)] \]

Two errors are denoted by \( \varepsilon_{\text{left}} = ||C_{\text{inv}}C_n - I_n||_F \) and \( \varepsilon_{\text{right}} = ||C_nC_{\text{inv}} - I_n||_F \), respectively. For the given stopping criteria \( r \), the iteration numbers and the CPU time seem to depend on the matrix size \( n \). As \( n \) increases, the central processing unit (CPU) time grows quickly, but \( \varepsilon_{\text{left}} \) and \( \varepsilon_{\text{right}} \) change a little. In tab. 1 we list the CPU time, \( \varepsilon_{\text{left}}, \varepsilon_{\text{right}}, \) and iteration numbers for different values of \( n \) with \( r = 10^{-11} \), respectively.
Table 1. The LSQR_M for the inverse of circulant matrix

<table>
<thead>
<tr>
<th>N</th>
<th>$\varepsilon_{\text{left}}$</th>
<th>$\varepsilon_{\text{right}}$</th>
<th>CPU time</th>
<th>Iteration numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>$4.06514 \cdot 10^{-11}$</td>
<td>$4.06409 \cdot 10^{-11}$</td>
<td>0.01</td>
<td>14</td>
</tr>
<tr>
<td>50</td>
<td>$1.55306 \cdot 10^{-11}$</td>
<td>$1.55305 \cdot 10^{-11}$</td>
<td>0.24</td>
<td>37</td>
</tr>
<tr>
<td>100</td>
<td>$8.34321 \cdot 10^{-11}$</td>
<td>$8.34312 \cdot 10^{-11}$</td>
<td>1.01</td>
<td>68</td>
</tr>
<tr>
<td>200</td>
<td>$5.64657 \cdot 10^{-10}$</td>
<td>$5.64656 \cdot 10^{-10}$</td>
<td>10.32</td>
<td>133</td>
</tr>
<tr>
<td>300</td>
<td>$2.98367 \cdot 10^{-10}$</td>
<td>$2.98368 \cdot 10^{-10}$</td>
<td>40.44</td>
<td>177</td>
</tr>
<tr>
<td>400</td>
<td>$1.00367 \cdot 10^{-9}$</td>
<td>$1.00366 \cdot 10^{-9}$</td>
<td>127.34</td>
<td>189</td>
</tr>
<tr>
<td>500</td>
<td>$4.79357 \cdot 10^{-9}$</td>
<td>$4.79368 \cdot 10^{-9}$</td>
<td>200.46</td>
<td>215</td>
</tr>
</tbody>
</table>

Conclusion

This paper reports the iteration LSQR for the inverse of a circulant matrix, which is usually the key to a class of boundary value problems. Compared with the existing methods, our iteration only involves matrix-matrix product and it is suitable for all non-singular circulant matrix.

Acknowledgment

The research was supported in part by the National Natural Science Foundation of China, Natural Science Foundation of Zhejiang Province and Foundation for Young Talents of ZJGSU under Grant No. Y6110639, 11201422, 11571312, 1020XJ1314019.

References