Preconditioned Legendre Spectral Galerkin Methods for the Non-separable Elliptic Equation

Xuhao Diao · Jun Hu · Suna Ma

Received: 19 April 2021 / Revised: 10 December 2021 / Accepted: 17 December 2021 / Published online: 23 February 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
The Legendre spectral Galerkin method of self-adjoint second order elliptic equations usually results in a linear system with a dense and ill-conditioned coefficient matrix. In this paper, the linear system is solved by a preconditioned conjugate gradient (PCG) method where the preconditioner \( M \) is constructed by approximating the variable coefficients with a \((T+1)\)-term Legendre series in each direction to a desired accuracy. A feature of the proposed PCG method is that the iteration step increases slightly with the size of the resulting matrix when reaching a certain approximation accuracy. The efficiency of the method lies in that the system with the preconditioner \( M \) is approximately solved by a one-step method based on the incomplete LU factorization technique with no fill-in, denoted by ILU(0). The ILU(0) factorization of \( M \in \mathbb{R}^{(N-1)^d \times (N-1)^d} \) can be computed using \( O(T^{2d} N^d) \) operations, and the number of nonzeros in the factorization factors is of \( O(T^d N^d) \), \( d = 1, 2, 3 \). A conclusion of the algorithm is to fast solve the resulting system from the Legendre Galerkin spectral method for Poisson equations with Dirichlet boundary conditions, which has a complexity of \( O(N^d) \). To further speed up the PCG method, an algorithm is developed for fast matrix-vector multiplications by the resulting matrix of Legendre-Galerkin spectral discretization, without the need to explicitly form it. The complexity of the fast matrix-vector multiplications is of \( O(N^d (\log_2 N)^2) \). In view that \( T \) is independent of \( N \) in one dimension and is set to be of order \( O(\log_2 N) \) in two and three dimensions, the PCG method has a \( O(N^d (\log_2 N)^{2d}) \) quasi-optimal complexity for a \( d \) dimensional domain with \((N-1)^d\) unknowns, \( d = 1, 2, 3 \). In addition, a fast direct solver for the three-dimensional Poisson equation is developed, which is of \( O(N^3 (\log_2 N)^2) \) and improves the existing results on the computational complexity. Numerical examples are given to demonstrate the efficiency of proposed preconditioners and the algorithm for fast matrix-vector multiplications.

✉ Suna Ma
masuna@pku.edu.cn

Xuhao Diao
diaoxuhao@pku.edu.cn

Jun Hu
hujun@math.pku.edu.cn

1 School of Mathematical Sciences, Peking University, Beijing 100871, China
Keywords  Spectral method · Non-separable elliptic equation · Preconditioned conjugate gradient method · Dense and ill-conditioned matrix · Incomplete LU factorization

Mathematics Subject Classification  Primary 65N35 · 65F10 · 65N22

1 Introduction

Spectral methods are an important tool in engineer and scientific computing for solving differential equations due to their high order of accuracy; see [3,12,25,26] and the references therein. However, for problems with general variable coefficients, spectral methods lead to a linear system with a dense and ill-conditioned matrix. Moreover, the dense matrix is usually not explicitly available, since it is costly to form it. In practice, it becomes rather prohibitive to solve the linear system by a direct solver or even an iterative method without preconditioning for the multi-dimensional cases, when the size of the matrix is large.

Over the years there has been intensive research on the spectral collocation method for solving problems with variable coefficients, since it is easy to implement, once the differentiation matrices are precomputed. However, the practitioners are plagued with the involved ill-conditioned linear systems (e.g., the condition number of the \( r \)-th order differential operator grows like \((N^{2r})^d\) \[9,26\], \(d\) is the dimension of space). This drawback causes severe degradation of expected spectral accuracy \[33\], while the accuracy of machine zero can be well observed from the well-conditioned spectral-Galerkin method (see e.g., \[29\]). One significant attempt to circumvent this barrier is to use a lower-order method (finite differences or finite elements \[2,8,11,17,18\]) or integration operator \[5,13,34\] as a preconditioner and to take advantage of the fact that the matrix-vector multiplication from a Fourier- or Chebyshev-spectral discretization can be performed in a quasi-optimal complexity. Another approach is the finite element multigrid preconditioning method proposed by Shen et al. \[27\].

This paper is concerned with preconditioned Legendre-Galerkin methods for the second order elliptic equation with variable coefficients. In contrast, it is more convenient for the spectral-Galerkin method, thanks to the weak formulation, than the spectral-collocation method in terms of error analysis. The Legendre-Galerkin method of self-adjoint second order elliptic equations leads to symmetric linear systems, which is easier to design methods for the numerical solution such as the preconditioned conjugate gradient (PCG) method. However, its efficiency is limited by the lack of fast transforms between the physical space (values at the Legendre Gauss points) and the spectral space (coefficients of the Legendre polynomials). In traditional spectral methods, a fast algorithm for Legendre expansions is a procedure to fast evaluate the Legendre expansion at Chebyshev points, and conversely, to fast evaluate the coefficients of the Legendre expansion from the table of its values at the Chebyshev-Gauss-Lobatto points. Recently, a series of work were done for fast discrete Legendre transforms between the values at the Legendre Gauss points and the coefficients of the Legendre polynomials \[15,19,21,32\]. In particular, an \(O(N(\log_2 N)^2/\log_2(\log_2 N))\) algorithm based on the FFT was proposed in \[15\] in one dimension for computing the discrete Legendre transform with a degree \(N−1\) Legendre expansion at \(N\) Legendre points.

There is some literature on fast solvers of Legendre-Galerkin methods for constant-coefficient elliptic problems. A fast direct solver was presented for the Legendre-Galerkin approximation of the two and three dimensional Helmholtz equations by Shen in \[29\], whose complexity is of \(O(N^{d+1})\), where \(d = 2, 3\) and \(N\) is the cutoff number of the polynomial expansion in each direction. An improved two-dimensional algorithm was constructed
by further exploring the matrix structures of the Legendre-Galerkin discretization in [28], whose complexity is of $O(N^2 \log_2 N)$, which was extended to the Legendre-Galerkin spectral approximation of the three dimensional Helmholtz equation in [1] showing empirically that the complexity of the method is not worse than $O(N^4)$. For variable-coefficient problems, an early work is the Chebyshev-Legendre Galerkin method for second-order elliptic problems introduced in [23], which is based on the Legendre-Galerkin formulation, and only the coefficients of Legendre expansions and the values at the Chebyshev-Gauss-Lobatto points are used in the computation.

The goal of this article is to fast solve the linear system resulting from the Legendre-Galerkin spectral discretization of second order elliptic equations with variable coefficients by the preconditioned conjugate gradient (PCG) method. The novelties of the paper lie in the following three folds:

- Firstly, the preconditioner is constructed by using a truncated Legendre series to approximate the variable coefficients. It is in the case that the iterative step of the PCG method only increase slightly with the size of the resulting matrix. A closely related preconditioner of [22,23] is to use a constant-coefficient problem to precondition variable-coefficient problems. However, for coefficients with large variations, iterative methods with that preconditioner usually converge very slowly [27].
- Secondly, by means of fast discrete Legendre transforms, an algorithm is developed for fast matrix-vector multiplications by the resulting matrix without the need to explicitly form it. As a result, they can be done in $O(N^d (\log_2 N)^2)$ operations.
- Last but not least, the system with the preconditioner as the coefficient matrix is approximately solved by a one-step method based on the ILU(0) factorization. Numerical experiments show that the one-step method exhibits more excellent performance than classic V-cycle algebraic multigrid (AMG) methods in solving the sparse system resulting from the Legendre-Galerkin spectral discretization. Thanks to the sparse structure of the preconditioner $M$, the complexity is of $O(N^d (\log_2 N)^2d)$ in view that the cutoff number $T$ of the Legendre series in each direction used to approximate the coefficient functions is independent of $N$ with $d = 1$; and is set to be of order $O(\log_2 N)$ with $d = 2, 3$. Theorem 4.1 indicates, in two and three dimensions, $T=O(\log_2 N)$ for analytic coefficients and $T=O(N^{4(d−1)/d})$ for functions belonging to the space $Q^k$ defined in (4.12). However, numerical results verify that the constant $T$ still achieves the effect that the number of iterations of the PCG method is independent of the discretization parameter $N$. As a special case of elliptic equations with variable coefficients, the linear system resulting from the Legendre-Galerkin method for the Poisson equation is solved in $O(N^d)$ operations by the PCG method with the ILU(0) preconditioning, from the points of numerical experiment results. In addition, the fast direct solver for the three-dimensional Poisson equation is developed, which is of $O(N^3 (\log_2 N)^2)$ and improves the existing result on the computational complexity.

The remainder of this article is organized as follows. Some preliminaries are given in Sect. 2. Section 3 introduces the Legendre-Galerkin method of the second-order elliptic equation with non-separable coefficients. In Sect. 4, the preconditioned conjugate gradient method with implementation issues is described. In Sect. 5, some numerical experiments are presented to illustrate the efficiency of both the algorithm for fast matrix-vector multiplications and the proposed preconditioner. The conclusion is in the last section.
2 Preliminaries

In this section, some properties of Legendre polynomials and a useful transform are presented.

2.1 Legendre Polynomials

Denote by $L_n(x)$ the Legendre polynomial of degree $n$ which satisfies the following three-term recurrence relation:

$$
L_0(x) = 1, \quad L_1(x) = x, \\
(n + 1)L_{n+1}(x) = (2n + 1)xL_n(x) - nL_{n-1}(x), \quad n \geq 1.
$$

The Legendre polynomials are orthogonal to each other with respect to the uniform weight function,

$$
\int_{-1}^{1} L_m(x)L_n(x)dx = \frac{2}{2m+1}\delta_{m,n}, \quad m, n \geq 0,
$$

where $\delta_{m,n}$ is the Kronecker delta symbol. Moreover, they satisfy the derivative recurrence relation

$$
(2n + 1)L_n(x) = L'_n(x) - nL'_{n-1}(x), \quad n \geq 1, \quad (2.1)
$$

and symmetric property

$$
L_n(-x) = (-1)^n L_n(x), \quad L_n(\pm 1) = (\pm 1)^n. \quad (2.2)
$$

Lemma 2.1 ([6]) For $m, n \geq 0$, it holds that

$$
L_m(x)L_n(x) = \sum_{s=0}^{\min(m,n)} \frac{m + n + \frac{1}{2} - 2s}{m + n + \frac{1}{2} - s} \frac{C_{s} C_{m-s} C_{n-s}}{C_{m+n-s}} L_{m+n-2s}(x), \quad (2.3)
$$

where

$$
C_r = \frac{1 \cdot 3 \ldots (2r - 3)(2r - 1)}{r!2^r}.
$$

2.2 Discrete Legendre Transforms

Given $N + 1$ values $c_0, c_1, \ldots, c_N$, the backward discrete Legendre transform (BDLT) calculates the discrete sums:

$$
f_k = \sum_{n=0}^{N} c_n L_n(x_k), \quad 0 \leq k \leq N, \quad (2.4)
$$

where the Legendre-Gauss quadrature nodes $x_0, x_1, \ldots, x_N$ are the roots of $L_{N+1}(x)$. Given $f_0, f_1, \ldots, f_N$, the forward discrete Legendre transform (FDLT) computes $c_0, c_1, \ldots, c_N$, which reads

$$
c_n = \frac{2n + 1}{2} \sum_{k=0}^{N} w_k f_k L_n(x_k), \quad 0 \leq n \leq N, \quad (2.5)
$$
where \( w_0, w_1, \ldots, w_N \) are the Legendre-Gauss quadrature weights. Assuming that \((L_n(x_j))_{j,n=0,\ldots,N}\) have been precomputed, the discrete Legendre transforms (2.4) and (2.5) can be carried out by a standard matrix-vector multiplication routine in about \( N^2 \) flops. In this paper, the algorithm in [15] is used for the fast computation of the discrete Legendre transforms (2.4) and (2.5) which is of \( O(N(\log_2 N)^2/\log_2(\log_2 N)) \) complexity, because it has no precomputational cost and only involves the FFT and Taylor approximations.

3 The Legendre-Galerkin Method of Non-separable Second Order Elliptic Equations

Consider non-separable second order elliptic equations of the form

\[
\begin{aligned}
-\text{div}(\beta(x)\nabla u) + \alpha(x)u &= f, \quad x \in \Omega = (-1, 1)^d, \\
|u|_{\partial\Omega} &= 0,
\end{aligned}
\tag{3.1}
\]

where \( d = 1, 2, 3 \), the coefficient functions \( \beta(x) \), \( \alpha(x) \) and \( f(x) \) are continuous, and \( 0 < b_1 \leq \beta(x) \leq b_2, 0 \leq \alpha(x) < a \) in \( \Omega \) for some positive constants \( b_1, b_2, a \).

Let \( P_N \) be the space of polynomials of degree less than or equal to \( N \), and \( X N = \{v \in P_N : v(\pm1) = 0\} \).

Denote \( X^d_N = (X_N)^d \). Then the Legendre-Galerkin approximation to (3.1) is: Find \( u_N \in X^d_N \) such that

\[
(\beta(x)\nabla u_N, \nabla v_N) + (\alpha(x)u_N, v_N) = (f, v_N), \quad \forall v_N \in X^d_N.
\tag{3.2}
\]

where \((u, v) = \int_{\Omega} uv dx\) is the scalar product in \( L^2(\Omega) \).

Denote

\[
\phi_k(x) := \frac{L_k(x) - L_{k+2}(x)}{\sqrt{2(2k + 3)}}.
\]

Due to (2.2), it is easy to know that the function \( \phi_k(x) \) satisfies the boundary condition of problem (3.1). Hence, the basis functions of the space \( X_N \) can be chosen as

\[
\phi_0(x), \phi_1(x), \ldots, \phi_{N-2}(x).
\]

- **One dimensional case** Assume \( u_N = \sum_{n=0}^{N-2} \hat{u}_n \phi_n(x) \), and denote

\[
A = \left[ (\beta(x)\phi_k', \phi_j') \right]_{0 \leq k, j \leq N-2},
\]

\[
B = \left[ (\alpha(x)\phi_k, \phi_j) \right]_{0 \leq k, j \leq N-2},
\]

\[
\hat{u} = (\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{N-2})^T,
\]

\[
F = \left( f_0, f_1, \ldots, f_{N-2} \right)^T, \quad f_k = (f, \phi_k).
\]

- **Two dimensional case** The multi-dimensional basis functions are constructed by using the tensor product of one-dimensional basis functions. In two dimensions, they read

\[
\varphi_{k,j}(x) := \phi_k(x)\phi_j(y), \quad k, j = 0, 1, \ldots, N-2.
\]
Assume \( u_N = \sum_{k,j=0}^{N-2} \hat{u}_{k,j} \varphi_{k,j}(x) \), and denote

\[
A = \left[ (\beta(x) \nabla \varphi_{k,j}, \nabla \varphi_{m,n}) \right]_{0 \leq k,j,m,n \leq N-2},
\]

\[
B = \left[ (\alpha(x) \varphi_{k,j}, \varphi_{m,n}) \right]_{0 \leq k,j,m,n \leq N-2},
\]

\[
\hat{u} = \left( \hat{u}_{0,0}, \hat{u}_{1,0}, \ldots, \hat{u}_{N-2,0}; \ldots; \hat{u}_{0,N-2}, \hat{u}_{1,N-2}, \ldots, \hat{u}_{N-2,N-2} \right)^T,
\]

\[
F = \left( f_0, f_1, \ldots, f_{N-2}, f_0, f_1, \ldots, f_{N-2} \right)^T,
\]

\[
f_{k,j} = (f, \varphi_{k,j}).
\]

**Three dimensional case** Similarly, the three-dimensional basis functions are as follows

\[
\psi_{i,j,k}(x) := \phi_i(x)\phi_j(y)\phi_k(z), \quad i, j, k = 0, 1, \ldots, N - 2.
\]

Assume \( u_N = \sum_{i,j,k=0}^{N-2} \hat{u}_{i,j,k} \psi_{i,j,k}(x) \), and denote

\[
A = \left[ (\beta(x) \nabla \psi_{i,j,k}, \nabla \psi_{m,n,l}) \right]_{0 \leq i,j,m,n,l \leq N-2},
\]

\[
B = \left[ (\alpha(x) \psi_{i,j,k}, \psi_{m,n,l}) \right]_{0 \leq i,j,m,n,l \leq N-2},
\]

\[
\hat{u} = \left( \hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{N-2} \right)^T,
\]

\[
\hat{u}_k = \left( \hat{u}_{0,0,k}, \hat{u}_{1,0,k}, \ldots, \hat{u}_{N-2,0,k}; \ldots; \hat{u}_{0,N-2,k}, \hat{u}_{1,N-2,k}, \ldots, \hat{u}_{N-2,N-2,k} \right),
\]

\[
F = \left( f_0, f_1, \ldots, f_{N-2} \right)^T,
\]

\[
f_k = \left( f_{0,0,k}, f_{1,0,k}, \ldots, f_{N-2,0,k}; \ldots; f_{0,N-2,k}, f_{1,N-2,k}, \ldots, f_{N-2,N-2,k} \right),
\]

\[
f_{i,j,k} = (f, \psi_{i,j,k}).
\]

Then the equation (3.2) is equivalent to the following algebraic system:

\[
(A + B)\hat{u} = F. \tag{3.3}
\]

For variable coefficients \( \alpha(x) \) and \( \beta(x) \), the matrices \( A \) and \( B \) in equation (3.3) are usually dense and ill-conditioned. Hence, it is prohibitive to use a direct inversion method or an iterative method without preconditioning. Moreover, it is imperative to use an iterative method with a good preconditioner.

### 4 Preconditioned Conjugate Gradient Method

In this article, the symmetric definite linear system (3.3) is solved by the preconditioned conjugate gradient (PCG) method. In each iteration of the PCG method, it needs to solve a system with \( M \) as the coefficient matrix and involves a matrix-vector multiplication. The complexity of these two steps dominates that of the algorithm. In order to accelerate the convergence rate of conjugate gradient type methods, an efficient preconditioner \( M \) is needed. Moreover, an accurate numerical solver is needed to solve the preconditioning equation \( Mz = r \). At last, fast matrix vector multiplications have to be used to further reduce the complexity of the algorithm.
4.1 Proposed Preconditioner

A preconditioner is prescribed for the coefficient matrix \( A + B \) of the linear system (3.3) in the following way.

Since \( \beta(x) \) and \( \alpha(x) \) are continuous, they can be approximated by a finite number of Legendre polynomials to any desired accuracy. That is, for any \( \epsilon_1 > 0, \epsilon_2 > 0 \), there exists \( t_1, t_2 \in \mathbb{N} \) and \( \hat{p}_{t_1} \in (P_{t_1})^d \), \( \hat{p}_{t_2} \in (P_{t_2})^d \) such that

\[
\| \beta(x) - \hat{p}_{t_1} \|_{L^\infty((-1,1)^d)} < \epsilon_1, \tag{4.1}
\]
\[
\| \alpha(x) - \hat{p}_{t_2} \|_{L^\infty((-1,1)^d)} < \epsilon_2. \tag{4.2}
\]

It is stressed that \( t_1 \) and \( t_2 \) can be surprisingly small when \( \beta(x) \), \( \alpha(x) \) are analytic or many times differentiable. For practical purposes, the preconditioner \( M \) is constructed by replacing \( \beta(x) \) with \( \hat{p}_{t_1} \) in the matrix \( A \) and \( \alpha(x) \) with \( \hat{p}_{t_2} \) in the matrix \( B \). Taking the one-dimensional case for example, one way to compute \( \hat{p}_{t_1} \) and \( \hat{p}_{t_2} \) is to truncate Legendre series of the functions \( \beta(x) \) and \( \alpha(x) \), which read

\[
\beta(x) = \sum_{t=0}^{\infty} \hat{\beta}_t L_t(x), \quad \hat{\beta}_t = \frac{2t + 1}{2} \int_{-1}^{1} \beta(x) L_t(x) dx,
\]
\[
\alpha(x) = \sum_{t=0}^{\infty} \hat{\alpha}_t L_t(x), \quad \hat{\alpha}_t = \frac{2t + 1}{2} \int_{-1}^{1} \alpha(x) L_t(x) dx.
\]

The polynomials \( \hat{p}_{t_1} \) and \( \hat{p}_{t_2} \) are formed by extracting the first \( t_1+1 \) and \( t_2+1 \) terms of \( \beta(x) \) and \( \alpha(x) \), respectively, as follows:

\[
\hat{p}_{t_1}(x) = \sum_{t=0}^{t_1} \hat{\beta}_t L_t(x), \quad \hat{p}_{t_2}(x) = \sum_{t=0}^{t_2} \hat{\alpha}_t L_t(x). \tag{4.3}
\]

Thus the preconditioner \( M \) in one dimension can be expressed as

\[
M = \left[ (p_{t_1}(x) \phi_j^t(x), \phi_j^t(x)) + (p_{t_2}(x) \phi_i^t(x), \phi_j^t(x)) \right]_{0 \leq i, j \leq N-2}. \tag{4.4}
\]

The coefficients \( \hat{\beta}_t \) and \( \hat{\alpha}_t \) can be derived by numerical quadrature in \( O(t_1^{d+1}) \) operations and \( O(t_2^{d+1}) \) operations \((d=1,2,3)\), respectively.

Note that the preconditioner \( M \) for one-dimensional case is a banded matrix with a fixed bandwidth dependent of \( t_1 \) and \( t_2 \) from the following proposition.

**Proposition 4.1** Denote

\[
M_1 = [\beta_{ji}]_{0 \leq i, j \leq N-2}, \quad \beta_{ji} = (p_{t_1}(x) \phi_i^j(x), \phi_j^t(x)),
\]
\[
M_2 = [\alpha_{ji}]_{0 \leq i, j \leq N-2}, \quad \alpha_{ji} = (p_{t_2}(x) \phi_i^j(x), \phi_j^t(x)).
\]

If \( t_1 \) and \( t_2 \) in (4.4) are fixed, the bandwidth \( q_1 \) of banded matrix \( M_1 \) and the bandwidth \( q_2 \) of banded matrix \( M_2 \) are as follows:

\[
q_1 = \begin{cases} 
2t_1 - 1, & t_1 \text{ even,} \\
2t_1 + 1, & t_1 \text{ odd}
\end{cases}, \quad \beta(x) \text{ is an odd function;}
\]
\[
q_1 = \begin{cases} 
2t_1 + 1, & t_1 \text{ even,} \\
2t_1 - 1, & t_1 \text{ odd}
\end{cases}, \quad \beta(x) \text{ is an even function;}
\]

Springer
where expansion coefficients in terms of $C_j$ respectively, where

Proof In the case $d = 1$, it follows from (2.3) that both $p_{t_1}(x)\phi'_k(x)$ and $p_{t_2}(x)\phi_k(x)$ can be represented in Legendre series, i.e.,

$$p_{t_1}(x)\phi'_k(x) = -\sqrt{\frac{2k + 3}{2}}L_{k+1}(x)\sum_{t=0}^{t_1} \tilde{\beta}_j L_t(x), \quad t_1 \geq k + 1,$$

$$p_{t_2}(x)\phi_k(x) = \frac{L_k(x) - L_{k+2}(x)}{\sqrt{2(2k + 3)}}\sum_{t=0}^{t_2} \tilde{\alpha}_j L_t(x), \quad t_2 \geq k,$$

where $\tilde{\beta}_j$ and $\tilde{\alpha}_j$ are Legendre expansion coefficients in terms of $C_j$ in (2.3) and $\hat{\alpha}_j$, $\hat{\beta}_j$ are Legendre expansion coefficients in terms of $C_j$ in (2.3) and $\hat{\alpha}_j$, $\hat{\beta}_j$. Together with parity arguments on $\beta(x)$ and $\alpha(x)$, this leads to the conclusion.

In what follows, it is concerned with the eigenvalues of the matrices arising from the Legendre-Galerkin approximation (3.2) to the Poisson equation, i.e., the elliptic equation (3.1) with constant coefficients. The corresponding algebraic system is rewritten in the following tensor product form

$$(A + \alpha B)\hat{u} = F,$$

where $\alpha \geq 0$ is a constant, and

$$A = \begin{cases} I, & d = 1, \\ I \otimes \mathcal{M} + \mathcal{M} \otimes I, & d = 2, \\ \mathcal{M} \otimes \mathcal{M} \otimes I + \mathcal{M} \otimes I \otimes \mathcal{M}, & d = 3, \end{cases}$$

$$B = \begin{cases} \mathcal{M}, & d = 1, \\ \mathcal{M} \otimes \mathcal{M}, & d = 2, \\ \mathcal{M} \otimes \mathcal{M} \otimes \mathcal{M}, & d = 3, \end{cases}$$

$$m_{ij} = (\phi_i(x), \phi_j(x)) = \begin{cases} \frac{2}{(2i+1)(2i+5)}, & j = i, \\ \frac{-1}{\sqrt{(2i+3)(2i+7)(2i+5)}}, & j = i + 2, \\ \frac{-1}{\sqrt{(2i+3)(2i+1)(2i+5)}}, & j = i - 2. \end{cases}$$

Lemma 4.1 Let $\lambda_{\text{max}}(A)$ and $\lambda_{\text{min}}(A)$ be the largest and the smallest eigenvalue of $A$, respectively, where $A$ is defined in (4.5). Similarly, the largest and the smallest eigenvalue of $B$ are denoted by $\lambda_{\text{max}}(B)$ and $\lambda_{\text{min}}(B)$, where $B$ is defined in (4.6). It holds that

$$\lambda_{\text{max}}(A) < d, \quad \lambda_{\text{min}}(A) > d(2N)^{-4(d-1)}$$

$$\lambda_{\text{max}}(B) < 1, \quad \lambda_{\text{min}}(B) > (2N)^{-4d}$$

Proof Firstly, for the case $d = 1$, i.e., $A = I, B = \mathcal{M}$, it yields that

$$\frac{r^T M r}{r^T r} = \frac{1}{\sum_{i=0}^{N-2} \gamma_i^2} \left[ \int_{-1}^{1} u(x) u(x) dx \right] \leq \frac{1}{\sum_{i=0}^{N-2} \gamma_i^2} \left[ \int_{-1}^{1} \left( \sum_{i=0}^{N-2} \gamma_i \phi_i \right)^2 dx \right]$$
the first step is to prove

\[ \lambda \]

It will be proved by induction that

\[ s_i \]

Thus,

Due to the fact

\[ M \]

where

\[ s_k \]

Then, it will be shown that \( \lambda_{\min}(\mathcal{M}) \geq 1/(2N)^4 \), and \( \lambda_{\min}(\mathcal{M}) \leq 3/N^4 \). To this end, the first step is to prove \( \lambda_{\min}(\mathcal{M}) \geq 1/(2N)^4 \). Since \( \mathcal{M} \) has alternating zero and nonzero elements, it can be split into two symmetric tridiagonal matrices. Namely,

\[ M(e) = [m_{2i,2j}]_{i,j=0,1,\ldots,[N/2]}, \]

\[ M(o) = [m_{2i-1,2j-1}]_{i,j=1,\ldots,[N/2]-1}. \]  

Due to the fact \( \lambda_{\min}(\mathcal{M}) = \min(\lambda_{\min}(M^{(o)}), \lambda_{\min}(M^{(e)})) \), it is sufficient to show

\[ \lambda_{\min}(M^{(e)}) > 1/(2N)^4, \quad \lambda_{\min}(M^{(o)}) > 1/(2N)^4. \]  

Denote \( M^{(e)} = \begin{bmatrix} \alpha_1 & \beta_2 \\ \beta_2 & \alpha_2 & \cdots \\ \cdots & \cdots & \cdots & \beta_n \\ \beta_n & \alpha_n \end{bmatrix} \). The \( LDL^T \) decomposition of \( M^{(e)} - 1/(2N)^4 I \) is

\[
M^{(e)} - 1/(2N)^4 I = \begin{bmatrix} 1 \\ l_1 & \cdots \\ \cdots & \cdots & \cdots \\ l_{n-1} & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ \cdots \\ \cdots \\ s_n \end{bmatrix} \begin{bmatrix} 1 & l_1 & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & l_{n-1} \\ \cdots & \cdots & \cdots & \cdots & 1 \end{bmatrix},
\]

where

\[ s_1 = \alpha_1 - 1/(2N)^4, \quad s_i = \alpha_i - 1/(2N)^4 - \frac{\beta_i^2}{s_{i-1}}, \quad i = 2, 3, \ldots, n. \]

It will be proved by induction that \( s_i > 0, i = 1, 2, 3, \ldots, n \). Note that

\[ \alpha_i = m_{2(i-1),2(i-1)} = \frac{2}{(4i-3)(4i+1)}, \]

\[ \beta_i = m_{2(i-2),2(i-1)} = \frac{1}{(4i-3)(4i-1)(4i-5)}. \]

It is obvious to find \( s_1 = \frac{2}{3} - \frac{1}{(2N)^4} > \frac{\alpha_1}{2} > 0 \) for \( N \geq 3 \). Assume \( s_k > \frac{\alpha_k}{2} > 0 \), then

\[
s_{k+1} - \frac{\alpha_{k+1}}{2} = \frac{\alpha_{k+1}}{2} - \frac{1}{(2N)^4} - \frac{\beta_{k+1}^2}{s_k} > \frac{1}{(k+1)(k+5)} - \frac{1}{(2N)^4} - \frac{(4k-3)(4k+1)}{(k+1)^2(k+3)(k-1)}
\]
Above together with Courant-Fischer min-max theorem yields

\[
\lambda \leq \frac{1}{(2N)^4}
\]

which implies that \(\lambda_{\min}(M^{(e)}) > 1/(2N)^4\). Similarly, \(\lambda_{\min}(M^{(o)}) > 1/(2N)^4\).

The second step is to show that

\[
\lambda_{\min}(M^{(e)}) \leq 3/N^4, \quad \lambda_{\min}(M^{(o)}) \leq 3/N^4.
\]

Take one second-order determinant of \(M^{(e)} - 3/N^4I\) as below:

\[
\mathcal{D} = \det \begin{bmatrix} 2 & -1/2 \sqrt{(2k + 3)(2k - 1)} & \frac{3}{4} \sqrt{(2k + 3)(2k - 1)} & \frac{1}{2} \sqrt{(2k + 3)(2k - 1)} \\ \frac{1}{2} \sqrt{(2k + 3)(2k - 1)} & \frac{1}{2} \sqrt{(2k + 3)(2k - 1)} & \frac{3}{4} \sqrt{(2k + 3)(2k - 1)} & -1 \frac{1}{2} \sqrt{(2k + 3)(2k - 1)} \\ \frac{3}{4} \sqrt{(2k + 3)(2k - 1)} & -1 & \frac{1}{2} \sqrt{(2k + 3)(2k - 1)} & \frac{3}{4} \sqrt{(2k + 3)(2k - 1)} \\ \frac{1}{2} \sqrt{(2k + 3)(2k - 1)} & \frac{3}{4} \sqrt{(2k + 3)(2k - 1)} & -1 & \frac{1}{2} \sqrt{(2k + 3)(2k - 1)} \end{bmatrix}
\]

\[
\mathcal{D} = \frac{9}{N^8} - \frac{12}{(N - 3)(N + 5)} \frac{3}{N^4} + \frac{12}{(N - 3)(N + 5)(N + 3)(N - 1)^2}
\]

where \(2 \leq k \leq N - 2\). When \(N\) is even and \(k = N/2\), it yields

\[
\mathcal{D} = \frac{9}{N^8} - \frac{4}{(N - 3)(N + 5)} \frac{12}{N^4} + \frac{12}{(N - 3)(N + 5)(N + 3)(N - 1)^2}
\]

\[
\mathcal{D} = \frac{9}{N^8} - \frac{12}{(N - 3)(N + 5)} \frac{3}{N^4} + \frac{12}{(N - 3)(N + 5)(N + 3)(N - 1)^2}
\]

\[
\mathcal{D} = \frac{9}{N^8} - \frac{12}{(N - 3)(N + 5)} \frac{3}{N^4} + \frac{12}{(N - 3)(N + 5)(N + 3)(N - 1)^2}
\]

which implies \(\lambda_{\min}(M^{(e)}) < 3/N^4\). Similarly, \(\lambda_{\min}(M^{(o)}) < 3/N^4\).

Secondly, in the case \(d = 2, 3\), thanks to (4.5) and (4.6), the conclusion (4.8) holds. And the proof is completed.

The conclusion of the above lemma plays an important role in the proof of Theorem 4.1 below.

**Theorem 4.1** Denote \(C = A + B\). If \(\epsilon_1 < \frac{b_1}{4(2N)^4(d-1)}\), \(\epsilon_2 < \frac{d(b_1 - \epsilon_1)}{4(2N)^4(d-1)}\), where \(\epsilon_1\) and \(\epsilon_2\) are mentioned in (4.1) and (4.2), the condition number of \(M^{-1}C\) is bounded by a constant.

**Proof** It follows from (4.1), (4.2) and (4.8) that,

\[
\frac{r^T(M - C)r}{r^T r} \leq \epsilon_1 - \frac{r^T Ar}{r^T r} \leq \epsilon_1 \lambda_{\max}(A) + \epsilon_2 \lambda_{\max}(B)
\]

\[
< d\epsilon_1 + \epsilon_2, \quad \forall 0 \neq r \in \mathbb{N}^{(N-1)^d},
\]

\[
\frac{r^T r}{r^T Mr} \leq \frac{1}{(b_1 - \epsilon_1)^2 r^T Ar} \leq \frac{1}{(b_1 - \epsilon_1) \lambda_{\min}(A)},
\]

\[
< \frac{(2N)^4(d-1)}{d(b_1 - \epsilon_1)}, \quad \forall 0 \neq r \in \mathbb{N}^{(N-1)^d}.
\]

Above together with Courant-Fischer min-max theorem yields

\[
\|C - M\|_2 < d\epsilon_1 + \epsilon_2, \quad \|M^{-1}\|_2 < \frac{(2N)^4(d-1)}{d(b_1 - \epsilon_1)}.
\]
Denote by \( \nu \) an arbitrary eigenvalue of \( M^{-1}C \), it holds that
\[
|1 - \nu| \leq \| I - M^{-1}C \|_2 \leq \| M^{-1} \|_2 \| M - C \|_2.
\]

If \( \epsilon_1 < \frac{b_1}{4(2N)^{4(d-1)}} \), \( \epsilon_2 < \frac{d(b_1 - \epsilon_1)}{4(2N)^{4(d-1)}} \), it implies that
\[
|1 - \nu| := \delta < \frac{1}{2}.
\]

As a result,
\[
\kappa(M^{-1}C) = \frac{\lambda_{\max}(M^{-1}C)}{\lambda_{\min}(M^{-1}C)} \leq \frac{|1 + \delta|}{|1 - \delta|} < 3,
\]
which completes the proof. \( \square \)

Theorem 2.5 in \cite{35} gives two error bounds on the approximation error of the truncated Legendre series in the uniform norm, which can be extended to the two-dimensional and three-dimensional functions being approximated by the tensor product of Legendre series. A combination of that result and Theorem 4.1 gives an upper bound of \( t_1 \) and \( t_2 \) for the two-dimensional and three-dimensional cases, since the condition on \( \epsilon_1 \) and \( \epsilon_2 \) is independent of \( N \) with \( d = 1 \):

- if \( \beta(x) \) and \( \alpha(x) \) are analytic,
  \[
t_1 = O(\log_2 N), \quad t_2 = O(\log_2 N);
  \]
  \( \text{(4.10)} \)
- if \( \beta(x), \alpha(x) \in \mathbb{Q}^k \),
  \[
t_1 = O(N^{\frac{4(d-1)}{k-0.5}}), \quad t_2 = O(N^{\frac{4(d-1)}{k-0.5}}),
  \]
  \( \text{(4.11)} \)

where the space \( \mathbb{Q}^k \) is defined as follows
\[
\mathbb{Q}^k = \{ f : \| f \|_T < \infty \},
\]
with the norm,
\[
\| f \|_T = \begin{cases} 
\sum_{m=0}^{k} \sum_{n=0}^{k} \int_{-1}^{1} \int_{-1}^{1} [(1 - x)(1 - y)]^{-1/4} \left| \frac{\partial^{m+n}f(x, y)}{\partial x^m \partial y^n} \right| dx dy, & d = 2, \\
\sum_{m=0}^{k} \sum_{n=0}^{k} \sum_{l=0}^{k} \int_{-1}^{1} \int_{-1}^{1} \int_{-1}^{1} [(1 - x)(1 - y)(1 - z)]^{-\frac{1}{2}} \left| \frac{\partial^{m+n+l}f(x, y, z)}{\partial x^m \partial y^n \partial z^l} \right| dx dy dz, & d = 3.
\end{cases}
\]

However, the numerical results in Sect. 5.2 indicate that the condition on \( t_1 \) and \( t_2 \) is not sharp.

**Remark 4.1** It is worthwhile to point out that Theorem 4.1 is a theoretical conclusion. Numerical experiments in Sect. 5 show that our algorithm which will be introduced in Sect. 4.2 to approximately solve the inverse of \( M \) still achieves the effect that the number of iterations of the PCG method is independent of the discretization parameter \( N \).
4.2 Incomplete LU Preconditioning for Banded Linear Systems

Without loss of generality, assume $\alpha(x) = 0$ in problem (3.1). And the preconditioner $M$ is constructed by using a $(T+1)$-term Legendre series in each direction to approximate the coefficient function $\beta(x)$. And $T$ is set to be of order $O(\log_2 N)$ for the two and three dimensional cases. In what follows, it will be shown that the preconditioning equation $Mz = r$ is approximately solved by a one-step process based on the ILU(0) factorization, see for instance [31, Chapter 10], in $O(N^d(\log_2 N)^{2d})$ operations for $d = 1, 2, 3$.

To approximately solve $Mz = r$, proceed as follows:

Step 1. Perform the ILU(0) factorization to obtain a sparse unit lower triangular matrix $L$ and a sparse upper triangular matrix $U$;

Step 2. Solve the unit lower triangular system $Ly = r$ by a forward substitution;

Step 3. Solve the upper triangular system $Uz = y$ by a backward substitution.

To evaluate the complexity of the one-step process to solve a system with the coefficient matrix $M$, the number of nonzeros in $M$ is considered. Taking two-dimensional problems as an example, the matrix $M$ can be rewritten in the following formulation:

$$ p_T(x) = \sum_{t=0}^{T} \sum_{k=0}^{K} \hat{\beta}_{tk}L_t(x)L_k(y), $$

$$ M = \sum_{t=0}^{T} \sum_{k=0}^{K} \beta_{tk} \left[ M_{1y}^{(k)} \otimes S_{1x}^{(t)} + S_{1y}^{(k)} \otimes M_{1x}^{(t)} \right], $$

where

$$ M_{1y}^{(k)} = \left[ (L_k(y)\phi_j(y), \phi_n(y)) \right]_{0 \leq j, n \leq N-1}, $$

$$ S_{1x}^{(t)} = \left[ (L_t(x)\phi'_i(x), \phi'_m(x)) \right]_{0 \leq i, m \leq N-1}, $$

$$ S_{1y}^{(k)} = \left[ (L_k(y)\phi'_j(y), \phi'_n(y)) \right]_{0 \leq j, n \leq N-1}, $$

$$ M_{1x}^{(t)} = \left[ (L_t(x)\phi'_i(x), \phi'_m(x)) \right]_{0 \leq i, m \leq N-1}. $$

It follows from Proposition 4.1 that each matrix $M_{1y}^{(T)}$, $S_{1x}^{(T)}$, $S_{1y}^{(T)}$, $M_{1x}^{(T)}$ has $O(TN)$ nonzero elements. Thus, the number of nonzeros in $M$ is of $O(T^2N^2)$. Then it is deduced that the number of nonzeros in $M$ for three-dimensional problems is of $O(T^3N^3)$.

From the algorithm, the cost of performing the ILU(0) factorization is of $O(T^{2d}N^d)$, which essentially depends on the number of nonzero elements in $M$. And the complexity of performing either the forward substitution or the backward substitution is of $O(T^{d}N^d)$. As a result, the one-step process to approximately solve the preconditioning equation $Mz = r$ costs $O(N^d(\log_2 N)^{2d})$ numerical operations, $d = 1, 2, 3$.

4.3 Fast Matrix-Vector Multiplications

The fast transforms of the Legendre expansions provide the possibility for fast matrix-vector multiplications of vectors by the discrete matrix $A + B$ resulting from the Legendre-Galerkin method.
Denote \( \Lambda = (a, b)^d \) and \( P^d_N = (P_N)^d \). Define the interpolation operator \( I_N : C(\Lambda) \to P^d_N(\Lambda) \) such that for any \( u \in C(\Lambda) \),

\[
(I_N u)(x) = u(x), \quad x \in \{x_0, x_1, \ldots, x_N\}^d,
\]
where \( x_0, x_1, \ldots, x_N \) are the Legendre-Gauss quadrature nodes mentioned above.

Given the coefficient vector \( \mathbf{p} \) of \( u_N \in X^d_N \), the matrix-vector multiplication of \( (A + B)\mathbf{p} \) is performed as follows (with the operation counts of each step in parentheses):

Step 1. Compute the Legendre coefficients of \( \nabla u_N \) and \( u_N \) respectively; \( (\mathcal{O}(N^d)) \)

Step 2. Perform the BDLT of \( \nabla u_N \) and \( u_N \) respectively; \( (\mathcal{O}(N^d \log_2 N)^2) \)

Step 3. Compute \( \beta(x) \nabla u_N \) and \( \alpha(x) u_N \) at the Legendre-Gauss quadrature nodes and then the FDLT of \( I_N(\beta(x) \nabla u_N), I_N(\alpha(x) u_N) \); \( (\mathcal{O}(N^d (\log_2 N)^2)) \)

Step 4. Compute the matrix-vector multiplication of \( (A + B)\mathbf{p} \). \( (\mathcal{O}(N^d)) \)

For clarity of presentation, fast matrix-vector multiplications for the one-dimensional case as an example are described in details. The other cases in two and three dimensions can be performed similarly.

Given \( \mathbf{p} = (\hat{p}_0, \hat{p}_1, \ldots, \hat{p}_{N-2})^T \), the computation

\[
(A\mathbf{p})_j = (I_N(\beta u'_N), \phi_j'), \quad j = 0, 1, \ldots, N - 2
\]

without explicitly forming the matrix \( A \) is presented as follows.

1. Using (2.1) to determine \( \{\hat{p}'\} \) from

\[
u'_N(x) = \sum_{k=0}^{N-2} \hat{p}_k \phi'_k(x) = \sum_{k=0}^{N} \hat{p}_k \Lambda_1(x);
\]

2. (BDLT) Compute

\[
u'_N(x_j) = \sum_{k=0}^{N} \hat{p}_k \Lambda_1(x_j), \quad j = 0, 1, \ldots, N;
\]

3. (FDLT) Determine \( \{\hat{\beta}_k\} \) from

\[
I_N(\beta u'_N)(x_j) = \sum_{k=0}^{N} \hat{\beta}_k \Lambda_1(x_j), \quad j = 0, 1, \ldots, N;
\]

4. For \( j = 0, 1, \ldots, N - 2 \), compute

\[
(A\mathbf{p})_j = \left(I_N(\beta u'_N), -\sqrt{\frac{2j + 3}{2}} \Lambda_1(x_j)\right) = -\sqrt{\frac{2}{2j + 3}} \hat{\beta}_{j+1}.
\]

Similarly, the computation

\[
(B\mathbf{p})_j = (I_N(\alpha u_N), \phi_j), \quad j = 0, 1, \ldots, N - 2
\]
without explicitly forming the matrix \( B \) is presented as follows.

1. Determine \( \{\hat{p}_k^{(1)}\} \) from

\[
u_N(x) = \sum_{k=0}^{N-2} \hat{p}_k \phi_k(x) = \sum_{k=0}^{N} \hat{p}_k^{(1)} \Lambda_1(x);
\]
2. (BDLT) Compute

\[ u_N(x_j) = \sum_{k=0}^{N} \hat{p}_k^{(1)} L_k(x_j), \quad j = 0, 1, \cdots, N; \]

3. (FDLT) Determine \( \{ \hat{\alpha}_k \} \) from

\[ I_N(\alpha u_N)(x_j) = \sum_{k=0}^{N} \hat{\alpha}_k L_k(x_j), \quad j = 0, 1, \cdots, N; \]

4. For \( j = 0, 1, \cdots, N - 2 \), compute

\[ (Bp)_j = \left( I_N(\alpha u_N), \phi_j(x) \right) = \frac{2\hat{\alpha}_j}{(2j + 1)\sqrt{2(2j + 3)}} - \frac{2\hat{\alpha}_{j+2}}{(2j + 5)\sqrt{2(2j + 3)}}. \]

Finally, the total cost for evaluating \( (A + B)p \) is dominated by fast discrete Legendre transforms, and is of \( O(N^d (\log_2 N)^2) \).

### 4.4 Fast Direct Solver for the Poisson Equation in Three Dimensions

The PCG method with the ILU(0) preconditioning provides an iterative method to solve the Poisson equation. A fast direct solver for the two- and three-dimensional Helmholtz equation based on the Legendre-Galerkin approximation was developed in [29], whose complexity is of \( O(N^{d+1}) \), where \( d = 2, 3 \) and \( N \) is the cutoff number of the polynomial expansion in each direction. An improved two-dimensional algorithm was constructed by further exploring the matrix structures of the Legendre-Galerkin discretization in [28], whose complexity is of \( O(N^2 \log_2 N) \), which is based on the extended cyclic reduction algorithm [30]. In this subsection, a fast direct solver for the three-dimensional Poisson equation is presented, which is of \( O(N^3 (\log_2 N)^2) \).

#### 4.4.1 The Generalized Cyclic Reduction Algorithm

The generalized cyclic reduction algorithm is the extension of the extended cyclic reduction method [30]. And it is used to solve the following block-tridiagonal system

\[
\begin{bmatrix}
B_1 & C_1 & \ldots & \ldots \\
A_2 & B_2 & C_2 & \ddots \\
\ddots & \ddots & \ddots & \ddots \\
A_{n-1} & B_{n-1} & C_{n-1} & \ldots & B_n & C_n & \ldots & \ldots \\
& A_n & & & B_n & C_n & \ldots & \ldots \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_{n-1} \\
x_n \\
\end{bmatrix}
= \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_{n-1} \\
y_n \\
\end{bmatrix},
\] (4.13)

where \( A_i, B_i, C_i \) are of the form \( \zeta \mathcal{H} + \eta I \in \mathbb{R}^{m \times m} \), \( \zeta \) and \( \eta \) are scalars, \( \mathcal{H} \) is a tridiagonal matrix, \( n = 2^k - 1 \). Note that the blocks \( A_i, C_i \) of the block-tridiagonal system solved by the extended cyclic reduction algorithm are in the form \( A_i = a_i I, C_i = c_i I \), where \( a_i \) and \( c_i \) are scalars. We follow the notations in [30]. The solution is obtained by the following three phases:

(0) Preprocessing phase. Instead of storing the matrices, it is to compute by the Newton’s method and store the zeros of \( B_i^{(r)} \) that represent them, where \( B_i^{(r)} \) are the polynomials of
\( \mathcal{H} \). Define \( B_i^{(r)}(1) = I, B_i^{(r)} \) for \( r = 1, 2, \ldots, k - 1 \) and \( i = 2^r, 2 \cdot 2^r, \ldots, (2^{k-r} - 1)2^r \), is determined by

\[
B_i^{(r)} = (B_i^{(r-2)}_{i-2^r} B_i^{(r-2)}_{i+2^r})^{-1} \left( R_i^{(r-1)} V_i^{(r-1)} B_i^{(r-2)}_{i-2^r} B_i^{(r-2)}_{i+2^r} + R_i^{(r-1)} V_i^{(r-1)} B_i^{(r-2)}_{i-2^r} B_i^{(r-2)}_{i+2^r} \right),
\]

where

\[
R_i^{(r)} = R_i^{(r-1)} R_i^{(r-1)} I_i^{2^{-r}-1}, \quad R_i^{(0)} = 0, \quad R_i^{(0)} = A_i, i = 2, 3, \ldots, n,
\]

\( V_i^{(r)} = V_i^{(r-1)} V_i^{(r-1)} I_i^{2^{-r}-1}, \quad V_i^{(0)} = 0, \quad V_i^{(0)} = C_i, i = 2, 3, \ldots, n. \)

(1) Reduction phase. Define \( p_i^{(r)}(0) = y_i, i = 1, 2, \ldots, n \). It is to compute \( p_i^{(r)} \) for \( r = 0, 1, \ldots, k - 2 \) and \( i = 2^{r+1}, 2 \cdot 2^{r+1}, \ldots, (2^{k-r} - 1)2^{r+1} \) by

\[
p_i^{(r+1)}(0) = R_i^{(r)} B_i^{(r-2)}_{i-2^r}^{-1} q_i^{(r+1)} + V_i^{(r)} B_i^{(r-1)}_{i+2^r}^{-1} q_i^{(r+1)} - p_i^{(r)},
\]

where

\[
q_i^{(r)} = \left( B_i^{(r)} \right)^{-1} B_i^{(r-1)} B_i^{(r-1)} p_i^{(r)}.
\]

(2) Back-substitution phase. Define \( x_0 = x_{n+1} = 0 \). It is to compute \( x_i \) for \( r = k - 1, k - 2, \ldots, 0 \) and \( i = 2^r, 3 \cdot 2^r, \ldots, (2^{k-r} - 1)2^r \) by

\[
x_i = \left( B_i^{(r)} \right)^{-1} B_i^{(r-1)} B_i^{(r-1)} q_i^{(r)} - R_i^{(r)} B_i^{(r-1)} q_i^{(r)} - x_i^{2^r}.
\]

It is pointed that \( R_i^{(r)}, V_i^{(r)} \) in (4.14), (4.15) and (4.16) are the polynomials of the tridiagonal matrix \( \mathcal{H} \) and \( \alpha_i^{(r)}, \gamma_i^{(r)} \) in the corresponding position are scalars in the extended cyclic reduction algorithm [30]. The fundamental to the extended cyclic reduction algorithm is that each block equation is scaled by a matrix \( (B_i^{(r-1)} B_i^{(r-1)} B_i^{(r-1)})^{-1} \) so that the algorithm is stable and its complexity can be controlled to the order we need. This still holds for the generalized cyclic reduction algorithm to solve the block-tridiagonal system (4.13). The detailed proof process is similar to that in the theorem in Sect. 6 of [30].

### 4.4.2 Fast Direct Solver

The corresponding algebraic system for the three-dimensional Poisson equation with homogeneous Dirichlet boundary conditions is written in the following tensor product form:

\[
\left[ \mathcal{M} \otimes \mathcal{M} \otimes I_{N-1} + \mathcal{M} \otimes I_{N-1} \otimes \mathcal{M} + I_{N-1} \otimes \mathcal{M} \otimes \mathcal{M} + \alpha \mathcal{M} \otimes \mathcal{M} \otimes \mathcal{M} \right] \hat{u} = F,
\]

where \( \alpha \geq 0 \) is a constant. Let \( U \) and \( F \) be the matrices assembled by the column vector \( \hat{u} \) and \( F \) respectively. The above system can be split into eight subsystems:

\[
\left[ I_{N/2} \otimes \mathcal{M}^{(i)} \otimes \mathcal{M}^{(j)} + \mathcal{M}^{(i)} \otimes I_{N/2} \otimes \mathcal{M}^{(j)} + I_{N/2} \otimes \alpha \mathcal{M}^{(i)} \otimes \mathcal{M}^{(j)} \otimes \mathcal{M}^{(k)} \right] U^{(i,j,k)} = F^{(i,j,k)},
\]

where \( i, j, k = e, o \) as defined in (4.9), \( U^{(i,j,k)} \) and \( F^{(i,j,k)} \) consist of components of \( U \) and \( F \) respectively.
The fast direct solver on (4.18) includes two steps:

Step 1. Acting the matrix \((I_{N/2} \otimes Q)^{-1}\) with
\[
Q = I_{N/2} \otimes \mathcal{M}^{(k)} + \mathcal{M}^{(j)} \otimes I_{N/2} + \alpha \mathcal{M}^{(j)} \otimes \mathcal{M}^{(k)}
\]
on both sides of the equation (4.18). It leads to
\[
\begin{align*}
(I_{N/2} \otimes \tilde{\mathcal{M}} + \mathcal{M}^{(i)} \otimes I_{(N/2)^2}) U^{(i,j,k)} &= (I_{N/2} \otimes Q)^{-1} F^{(i,j,k)},
\end{align*}
\]
where
\[
\tilde{\mathcal{M}} = Q^{-1}(\mathcal{M}^{(j)} \otimes \mathcal{M}^{(k)}).
\]

The right-hand side \((I_{N/2} \otimes Q)^{-1} F^{(i,j,k)}\) can be obtained by solving \(N/2\) times \(Q^{-1}\). Note that \(Q\) is derived from the Legendre-Galerkin method for the two-dimensional Poisson equation and is a block-tridiagonal matrix in which each block is a tridiagonal matrix in the case \(\alpha \neq 0\) in (4.17). In the end, the system with a block-tridiagonal matrix is solved by using the generalized cyclic reduction algorithm.

Step 2. Solving the system (4.19) by the extended cyclic reduction algorithm in [30]. In the implementation of the reduction phase and the back-substitution phase, the following inverse matrix
\[
(\tilde{\mathcal{M}} + h I_{(N/2)^2})^{-1} = \left(\mathcal{M}^{(j)} \otimes \mathcal{M}^{(k)} + h Q\right)^{-1} \cdot Q
\]
is needed. Denote
\[
T := \mathcal{M}^{(j)} \otimes \mathcal{M}^{(k)} + h Q,
\]
then the system with the coefficient matrix \(T\) can be solved by the similar process to obtain the inverse matrix of \(Q\) in Step 1.

4.4.3 Complexity

We first resort to the computational complexity of the extended cyclic reduction algorithm. Its main complexity consists of two parts:

1. The complexity of reduction phase. Recall that \(k = \log_2(n + 1)\) in [30], the operation count for solving tridiagonal systems with the matrices in the form \(\mathcal{M}^{(e)} + \alpha I \in \mathbb{R}^{m \times m}\) is
\[
c_r = 9m[(2k - 6)(n + 1) + 2k + 6].
\]

In addition, the operation count for scalar-vector multiplication and vector-vector addition is
\[
e_r = \sum_{r=1}^{k-1} (2^{k-r} - 1) \cdot 4m + \sum_{r=0}^{k-2} 2^{k-r-1} \cdot [(2^{r+1} - 2) \cdot 2m]
= [(2k - 2)2^k - 4k + 4]m.
\]

2. The complexity of back-substitution phase. The operation count for solving tridiagonal systems with the matrices in the form \(\mathcal{M}^{(e)} + \alpha I\) is
\[
c_b = 9m[(2k - 5)(n + 1) + 2k + 5].
\]
In addition, the operation count for scalar-vector multiplication and vector-vector addition is
\[
e_b = \sum_{r=0}^{k-1} (2^{k-r-1} - 2) \cdot 4m + 4m + \sum_{r=0}^{k-1} 2^{k-r-1} \cdot [(2^{r+1} - 2) \cdot 2m]
= [k2^{k+1} - 4k]m.
\]

In summary, the computational complexity is
\[
F_e = c_r + e_r + c_b + e_b = [(40k - 101)2^k + 28k + 103]m.
\]

Then, the complexity of the fast direct solver for the system \((4.17)\) consists of three parts:

1. The operation count of obtaining the right-hand side \((I_{N/2} \otimes Q)^{-1}F(i,j,k)\) in \((4.19)\) is
\[
c_1 = \frac{N}{2} \times F_e = \frac{N}{2} \left[ 10N^2 \log_2 N - \frac{141}{4} N^2 \right],
\]
where \(n = N/2, m = N/2\) in \(F_e\).

2. In the step of solving the system \((4.13)\) by the generalized cyclic reduction, the sum of the operation count for solving tridiagonal systems with \(M^{(e)} + \eta/\zeta I \in \mathbb{R}^{m \times m}\) and tridiagonal matrix-matrix multiplication is
\[
c_2 = 9m \left[ (4k - 11)(n + 1) + 4k + 11 \right] + 6m \left[ 4 \cdot 2^k - 4k - 4 \right].
\]

In addition, the operation count for scalar-vector multiplication and vector-vector addition is
\[
e_g = m \left[ (8k - 7)2^k + 18 \right].
\]

3. The operation count of solving \(N/2\) times of the system \((4.19)\) by the extended cyclic reduction algorithm is
\[
c_3 = \frac{N}{2} \left[ (4k - 11)N/2 + 4k + 11 \right] \left[ c_2 + e_g + 9(N/2)^2 \right]
+ (N/2)^2 \left[ (4k - 2)N/2 + 4 \right].
\]
Taking \(n = N/2, m = N/2\), the computational complexity is
\[
\mathcal{F} = 8(c_1 + c_3) \approx 176N^3(\log_2 N)^2 - 1204N^3 \log_2 N + 2058N^3.
\]

**Remark 4.2** Taking \(n = N/2, m = N/2\) in \(F_e\), the complexity of the fast direct solver for the system resulting from the Poisson equation in two dimensions is
\[
\mathcal{F} = 4F_e \approx 40N^2 \log_2 N - 141N^2.
\]

### 4.5 Complexity of the PCG Method with Preconditioner \(M\)

The four arithmetic operations of each component in implementation process of the PCG method with preconditioner \(M\) is listed as below:

1. The operation count of performing the backward discrete Legendre transform in \((2.4)\) is
\[ T_1 = \log_2 N / \log_2(\log_2 N) \left[ \left( 10 + \frac{4}{\pi} \right) N \log_2 N + (26 + 4m)N \right] \]

\[ + N \left[ \frac{l^2 + 3l}{4} + 10 \log_2 N + 26 \right], \]

where \( T_1 \) consists of the transform from the coefficients of Legendre polynomials to the coefficients of Chebyshev polynomials and a non-uniform discrete cosine transform [15]. In our implementation, \( m=5 \) is the number of terms of (3.3) in [14] approximating \( L_n(\cos \theta) \), and \( l = 7 \) is the number of the truncated series of (2.5) in [15]. The discrete cosine transform and the discrete sine transform are done by means of FFT and FFT is in \( \frac{4}{\pi} N \log_2 N + 4N \) operations (cf. P.502 in [4]).

2. The operation count of performing the forward discrete Legendre transform in (2.5) is

\[ T_2 = \log_2 N / \log_2(\log_2 N) \left[ \left( 20 + \frac{8}{\pi} \right) N \log_2 N + \left( \frac{8}{\pi} + 72 + 5m \right)N \right] \]

\[ + N \left[ \frac{l^2 + 3l}{4} + 20 \log_2 N + 72 \right], \]

where \( T_2 \) consists of two transforms, which are a slight modification of those in \( T_1 \) [15].

3. The operation cont of performing one matrix-vector multiplication by \( A \) in (3.3) is

\[ T_A = \begin{cases} 4N(T_1 + T_2) + 4N^2, & \text{for the 2D case,} \\ 9N^2(T_1 + T_2) + 12N^3, & \text{for the 3D case.} \end{cases} \]

The operation count of performing one matrix-vector multiplication by \( B \) in (3.3) is

\[ T_B = \begin{cases} 2N(T_1 + T_2) + 4N^2, & \text{for the 2D case,} \\ 3N^2(T_1 + T_2) + 12N^3, & \text{for the 3D case.} \end{cases} \]

4. The operation count of ILU(0) factorization process is

\[ \mathcal{R} = (2\mathcal{L}^2 + \mathcal{L})N^d, \quad d = 2, 3 \]

where

\[ \mathcal{L} = \max \{ (T_1 + 2)(2T_1 + 1), (T_2 + 2)(2T_2 + 5) \}, \quad \text{for the 2D case,} \]

\[ \mathcal{L} = \max \{ (2T_1 + 1)(2T_1 + 5)(T_1 + 2), (2T_2 + 5)(2T_2 + 5)(T_2 + 2) \}, \quad \text{for the 3D case,} \]

\( \mathcal{L} \) is the half bandwidth of the matrix \( M \) and \( M \) is constructed by using a \( (T_1+1) \)-term Legendre series and a \( (T_2+1) \)-term Legendre series in each direction to approximate the coefficient function \( \beta(x) \) and \( \alpha(x) \), respectively.

5. The operation count of forward substitution is

\[ (2\mathcal{L} + 1)N^d, \quad d = 2, 3. \]

6. The operation count of backward substitution is

\[ 2\mathcal{L}N^d, \quad d = 2, 3, \]

In summary, the total computational complexity of the PCG method is

\[ \text{number of PCG iterations} \times \left( T_A + T_B + (4\mathcal{L} + 1)N^d \right) + \mathcal{R}. \]
Table 1  Computational cost in each component of the method

| Forming preconditioner $M$ | Performing one matrix-vector product | ILU(0) factorization | Inverting $M$ approximately based on ILU(0) |
|---------------------------|-------------------------------------|----------------------|------------------------------------------|
| $O(L^{2d}N^d)$             | $O(N^d(\log_2 N)^2)$                | $O(L^{2d}N^d)$       | $O(L^dN^d)$                              |

Table 2  CPU time for fast multiplication of matrix $A \in \mathbb{R}^{(N-1)\times(N-1)}$ by any vector

| $N$   | 320 | 640 | 1280 | 2560 | 5120 | 10240 |
|-------|-----|-----|------|------|------|-------|
| $\beta(x) = (2x^2 + 1)^4$ | Time(s) | 0.0157 | 0.0211 | 0.0306 | 0.0514 | 0.1130 | 0.2349 |
| $\beta(x) = e^{2x}$       | Time(s) | 0.0184 | 0.0207 | 0.0301 | 0.0522 | 0.1128 | 0.2376 |

Table 3  CPU time for fast multiplication of matrix $B \in \mathbb{R}^{(N-1)\times(N-1)}$ by any vector

| $N$   | 320 | 640 | 1280 | 2560 | 5120 | 10240 |
|-------|-----|-----|------|------|------|-------|
| $\alpha(x) = (2x^2 + 1)^4$ | Time(s) | 0.0178 | 0.0215 | 0.0320 | 0.0529 | 0.1140 | 0.2414 |
| $\alpha(x) = e^{2x}$       | Time(s) | 0.0202 | 0.0236 | 0.0300 | 0.0518 | 0.1188 | 0.2404 |

Besides, the operation count of forming preconditioner $M$ is

$$S = dL^{2d}N^d + d^2L^2N^2 + T_1^{2d} + T_2^{2d}.$$  

Finally, Table 1 shows the computational cost spend on different parts of the method.

5 Numerical Results

In this section, some numerical experiments are provided to demonstrate the effectiveness of both matrix-vector multiplications and the proposed preconditioner $M$. Meanwhile, numerical results of the one-step process based on the ILU(0) factorization and the classic AMG method to solve the sparse system are compared. In particular, a class of coefficient functions with high variations are test. In all numerical experiments, the stopping criterion $\varepsilon = 10^{-12}$. The code is in MATLAB 2016b.

5.1 Numerical Results for Fast Matrix-Vector Multiplications

The first test investigates the time taken to compute one matrix-vector multiplication of a vector by the discretization matrix resulting from the Legendre-Galerkin method. The vector is generated randomly by the rand() command. For this purpose, the numerical experiments are carried out for different coefficients $\beta(x)$ and $\alpha(x)$ in one, two and three dimensions.

The average time of 10 tests of matrix-vector multiplications by matrix $A$ is reported in Tables 2, 4, 6, and by matrix $B$ in Tables 3, 5, 7 respectively. It can be observed that the time scales roughly linearly in the dimension of matrices $A$ and $B$, which is consistent with the
Table 4 CPU time for fast multiplication of matrix $A \in \mathbb{R}^{(N-1)^2 \times (N-1)^2}$ by any vector

| $N$   | 16   | 32   | 64   | 128  | 256  | 512  |
|-------|------|------|------|------|------|------|
| $\beta(x) = (2x^2 + 2y^2 + 1)^4$ |      |      |      |      |      |      |
| Time(s) | 0.0596 | 0.2052 | 0.7670 | 2.9913 | 12.2643 | 50.4063 |
| $\beta(x) = e^{2(x+y)}$ |      |      |      |      |      |      |
| Time(s) | 0.0575 | 0.2077 | 0.7650 | 2.9453 | 12.0757 | 49.7519 |

Table 5 CPU time for fast multiplication of matrix $B \in \mathbb{R}^{(N-1)^2 \times (N-1)^2}$ by any vector

| $N$   | 16   | 32   | 64   | 128  | 256  | 512  |
|-------|------|------|------|------|------|------|
| $\alpha(x) = (2x^2 + 2y^2 + 1)^4$ |      |      |      |      |      |      |
| Time(s) | 0.0297 | 0.1046 | 0.3723 | 1.5264 | 6.2888 | 26.0985 |
| $\alpha(x) = e^{2(x+y)}$ |      |      |      |      |      |      |
| Time(s) | 0.0281 | 0.1049 | 0.3794 | 1.4797 | 6.0668 | 25.1772 |

Table 6 CPU time for fast multiplication of matrix $A \in \mathbb{R}^{(N-1)^3 \times (N-1)^3}$ by any vector

| $N$   | 4    | 8    | 16   | 32   | 64   |
|-------|------|------|------|------|------|
| $\beta(x) = (2x^2 + 2y^2 + 2z^2 + 1)^4$ |      |      |      |      |      |
| Time(s) | 0.0778 | 0.3989 | 2.5246 | 20.4493 | 166.0483 |
| $\beta(x) = e^{2(x+y+z)}$ |      |      |      |      |      |
| Time(s) | 0.0770 | 0.3982 | 2.2168 | 16.3830 | 132.7023 |

discussions in Sect. 4.3. As is shown in [15], the fast FFT-based discrete Legendre transform becomes more efficient than the direct approach when $N$ is larger for an $N$-point transform. However, this case is not considered in the current numerical results of this subsection.

5.2 Numerical Results for the Number of Iterations

The second test is to firstly demonstrate the effectiveness of proposed preconditioner $M$. To this end, the iteration steps of the PCG method with a constant-coefficient preconditioner (PCG-I) and the PCG method with the proposed preconditioner $M$ (PCG-II) are compared. The preconditioner $M$ is constructed by approximating $\beta(x)$ and $\alpha(x)$ with a $(t_1+1)$-term Legendre series and a $(t_2+1)$-term Legendre series in each direction respectively. Here, two cases that $t_1$ and $t_2$ are constants and of order $O(\log_2 N)$ are tested. In each iteration of PCG-I, the system with the constant-coefficient preconditioner as the coefficient matrix is solved by direct methods in $O(N)$ operations for $d = 1$ [23], in $O(N^2 \log_2 N)$ operations for $d = 2$ [23,28] and in $O(N^3 (\log_2 N)^2)$ operations for $d = 3$ (see Sect. 4.4).

Secondly, the test is to show the effectiveness of the one-step algorithm introduced in Sect. 4.2. To make a comparison, a classic V-cycle algebraic multigrid (AMG) method is used to solve the sparse system with the proposed preconditioner $M$ as the coefficient matrix in the PCG method (PCG-III). The classic algebraic multigrid consists of a C/F-splitting in the coarsening algorithm and the standard interpolation in the construction of prolongation matrix. In the V-cycle scheme, the Gauss-Seidel type smoother is chosen and the number of presmoothing and postsmoothing is denoted by $\mu$. Let $l = \lceil \frac{1}{2} \log_2 (N - 1)^d - 4 \rceil$, the
Table 7 CPU time for fast multiplication of matrix \( B \in \mathbb{R}^{(N-1)^3 \times (N-1)^3} \) by any vector

| \( N \) | 4   | 8   | 16  | 32  | 64  |
|-------|-----|-----|-----|-----|-----|
| \( \alpha(x) = (2x^2 + 2y^2 + 2z^2 + 1)^4 \) |     |     |     |     |
| Time(s) | 0.0251 | 0.1404 | 0.7706 | 6.2419 | 50.8091 |
| \( \alpha(x) = e^{2(x+y+z)} \) |     |     |     |     |
| Time(s) | 0.0255 | 0.1268 | 0.7127 | 5.7871 | 47.1070 |

Table 8 Iteration counts for Example 1 in the 1D case

| \( N \) | 320 | 640 | 1280 | 2560 | 5120 | 10240 |
|-------|-----|-----|------|------|------|-------|
| Example 1 (a) | | | | | | |
| PCG-I | \( t_1 = 0, t_2 = 0 \) | 130 | 134 | 138 | 142 | 145 | 148 |
| PCG-II | \( t_1 = 4, t_2 = 2 \) | 16 | 17 | 17 | 17 | 18 | 18 |
|        | \( t_1 = 6, t_2 = 2 \) | 7 | 8 | 8 | 8 | 8 | 8 |
| Example 1 (b) | | | | | | |
| PCG-I | \( t_1 = 0, t_2 = 0 \) | 108 | 110 | 113 | 116 | 119 | 121 |
| PCG-II | \( t_1 = 4 \) | 11 | 11 | 11 | 12 | 12 | 12 |
|        | \( t_1 = 5 \) | 7 | 7 | 7 | 8 | 8 | 8 |

number of levels is set as the following

\[
\text{number of levels} = \begin{cases} 
2, & l < 2, \\
8, & l > 8, \\
l, & \text{otherwise}.
\end{cases}
\]

Numerical results are presented in Tables 8, 9 and 10. Test problems are considered as follows:

**Example 1** (1D) The problem (3.1) in one dimension takes the following analytic coefficients:
(a) \( \beta(x) = (2x^2 + 1)^4 \) and \( \alpha(x) = \cos(x) + 10 \).
(b) \( \beta(x) = e^{2x} \) and \( \alpha(x) = 0 \).

**Example 2** (2D) The problem (3.1) in two dimensions takes the following analytic coefficients:
(a) \( \beta(x) = (2x^2 + 2y^2 + 1)^4 \) and \( \alpha(x) = \cos(x + y) + 10 \).
(b) \( \beta(x) = e^{2(x+y)} \) and \( \alpha(x) = 0 \).

**Example 3** (3D) The problem (3.1) in three dimensions takes the following analytic coefficients:
(a) \( \beta(x) = (2x^2 + 2y^2 + 2z^2 + 1)^4 \) and \( \alpha(x) = \cos(x + y + z) + 10 \).
(b) \( \beta(x) = e^{x+y+z} \) and \( \alpha(x) = 0 \).

Table 8 reports the results for the one-dimensional problem. Note that the PCG method with the proposed preconditioner \( M \) exhibits excellent performance in terms of iteration step over the PCG method with a constant-coefficient preconditioner. Besides, the iteration steps of PCG-II only increase slightly as the discretization parameter \( N \) increases. As \( \beta(x) \) and
Table 9 Iteration counts for Example 2 in the 2D case

| N   | 32   | 64   | 128  | 256  | 512  |
|-----|------|------|------|------|------|
| Example 2 (a) PCG-I | \( t_1 = 0, t_2 = 0 \) | 215  | 288  | 297  | 300  | 300  |
|      PCG-II          | \( t_1 = 4, t_2 = 3 \) | 15   | 18   | 23   | 24   | 24   |
|      PCG-III (\( \mu = 4 \)) | \( t_1 = 2 \log_2 N - 8, t_2 = \log_2 N \) | 29   | 16   | 10   | 6    | 4    |
| Example 2 (b) PCG-I | \( t_1 = 0, t_2 = 0 \) | 495  | 569  | 589  | 593  | 594  |
|      PCG-II          | \( t_1 = 5 \) | 12   | 18   | 30   | 30   | 30   |
|      PCG-III (\( \mu = 3 \log_2 N + 30 \)) | \( t_1 = 5 \) | 36   | 52   | 55   | 53   | 65   |

\( \alpha(x) \) are approximated by a finite number of Legendre series to a higher accuracy, the iteration steps decrease which indicates that the PCG method converges more quickly. Tables 9 and 10 list the numerical results for the two- and three-dimensional problems respectively. The iteration steps of PCG-II for the 2D and 3D cases behave similarly as the 1D case. All of the examples show that the proposed preconditioner \( M \) is very effective for problems with large variations in coefficient functions. Besides, the number of iterations of the PCG-II method with \( t_1 \) and \( t_2 \) being of order \( O(\log_2 N) \) decrease when the discretization parameter \( N \) is large, which indicate that the condition on \( t_1 \) and \( t_2 \) is not sharp.

Meanwhile, Tables 9 and 10 present the results of PCG-II and PCG-III for the two- and three-dimensional problems respectively. On the one hand, the number of iterations of PCG-III still increase when the discretization parameter \( N \) is large, while iteration counts of PCG-II remains constant. On the other hand, the number of presmoothing and postsmoothing \( \mu \) is set to be larger when the coefficient function \( \beta(x) \) has more high variations such as the cases (b) of Examples 2 and 3, which leads to higher costs.

**Example 4** The PCG method with the proposed preconditioner \( M \) can be applied to more general second order problems:

\[
\begin{align*}
-\nabla \cdot (C \nabla u) + \alpha(x)u &= f, \quad x \in \Omega = (-1, 1)^d, \quad d = 2, 3, \\
|u|_{\partial \Omega} &= 0,
\end{align*}
\]

(5.1)

where the coefficient \( C = \begin{pmatrix} \beta_{11} & \cdots & \beta_{1d} \\ \vdots & \ddots & \vdots \\ \beta_{d1} & \cdots & \beta_{dd} \end{pmatrix} \) is a \( d \)-by-\( d \) matrix that is uniformly elliptic on the domain \( \Omega \).

Consider the problem (5.1) with the following coefficients:

(a) \( d=2, \quad C = \begin{pmatrix} e^{x+y} & 0.05 \\ 0 & \cos(x+y) + 1 \end{pmatrix} \) and \( \alpha(x)=0 \),

(b) \( d=3, \quad C = \begin{pmatrix} e^{2(x+y+z)} & 0 \\ 0 & \cos(x+y+z) + 15 \\ 0 & \cos(x+y+z) + 15 \end{pmatrix} \) and \( \alpha(x)=0 \).
### Table 10  Iteration counts for Example 3 in the 3D case

| $N$ | 20   | 40   | 60   | 80   | 100  |
|-----|------|------|------|------|------|
| **Example 3 (a)** |      |      |      |      |      |
| PCG-I | $t_1 = 0, t_2 = 0$ | 274  | 458  | 485  | 515  | 529  |
| PCG-II | $t_1 = 4, t_2 = 3$ | 15   | 20   | 22   | 23   | 23   |
|      | $t_1 = 2 \log_2 N-4, t_2 = 2 \log_2 N-7$ | 18   | 7    | 5    | 6    | 5    |
| PCG-III ($\mu = 4$) | $t_1 = 4, t_2 = 3$ | 18   | 30   | 43   | 60   | 84   |
|      | $t_1 = 2 \log_2 N-4, t_2 = 2 \log_2 N-7$ | 18   | 16   | 23   | 30   | 40   |
| **Example 3 (b)** |      |      |      |      |      |
| PCG-I | $t_1 = 0, t_2 = 0$ | 1251 | 2048 | 2606 | 3178 | 3340 |
| PCG-II | $t_1 = 5$ | 5    | 7    | 9    | 10   | 10   |
|      | $t_1 = 2 \log_2 N-3$ | 5    | 5    | 6    | 4    | 5    |
| PCG-III ($\mu = 30 \log_2 N + 30$) | $t_1 = 5$ | 15   | 20   | 20   | 19   | 19   |
|      | $t_1 = 2 \log_2 N-3$ | 15   | 19   | 19   | 20   | 20   |
Table 11 Iteration counts for Example 4 (a) in the 2D case

| N   | 32  | 64  | 128 | 256 | 512 |
|-----|-----|-----|-----|-----|-----|
| PCG-I | $t_{11} = 0, t_{12} = 0$ | 73  | 82  | 85  | 86  | 86  |
|      | $t_{11} = 5, t_{12} = 3$ | 10  | 15  | 22  | 23  | 23  |
|      | $t_{11} = 2 \log_2 N - 7, t_{12} = 2 \log_2 N - 9$ | 18  | 15  | 14  | 5   | 5   |

Table 12 Iteration counts for Example 4 (b) in the 3D case

| N   | 40  | 60  | 80  | 100 |
|-----|-----|-----|-----|-----|
| PCG-I | $t_{11} = 0, t_{12} = 0$ | 1640 | 1665 | 1680 | 1695 |
|      | $t_{11} = 7, t_{12} = t_{13} = 3$ | 11   | 27  | 70  | 135 |
|      | $t_{11} = 2 \log_2 N - 3, t_{12} = t_{13} = 2 \log_2 N - 7$ | 10   | 19  | 30  | 31  |

For problems of the form (3.1), Shen in [23] has pointed out that it is efficient to make a change of dependent variable $v = \sqrt{\beta} u$ [7] which reduces (3.1) to the following equation:

$$
\begin{align*}
-\Delta v + p(x)v &= q, \quad x \in \Omega = [-1, 1]^d, \quad d = 1, 2, 3, \\
|v|_{\partial \Omega} &= 0,
\end{align*}
$$

(5.2)

where $p(x) = \frac{\Delta(\sqrt{\beta})}{\sqrt{\beta}} + \alpha(x)$ and $q(x) = \frac{f}{\sqrt{\beta}}$, then the resulting system from the above problem (5.2) can be solved by using a preconditioned conjugate gradient method with a constant-coefficient preconditioner. However, this strategy is limited in the situation such as problem (5.1). In what follows, both PCG-I and PCG-II are performed for the linear systems arising from problem (5.1). The preconditioner $M$ is constructed by approximating $\beta_{jj}$ in case (a) with the $t_{1j}$-term Legendre polynomials, $j = 1, 2$, and $\beta_{jj}$ in case (b) with the $t_{1j}$-term Legendre polynomials, $j = 1, 2, 3$. Numerical results are shown in Tables 11 and 12.

Tables 11 and 12 show that the strategy that using a constant-coefficient problem precondition variable-coefficient problems is not effective if coefficient functions have large variation over the domain. Further, the computational complexity of PCG-I and PCG-II for Example 4 is compared, which is presented in Fig. 1. In summary, the total computational complexity of the PCG-I method is

$$
\text{number of PCG-I iterations} \times \left(T_A + \mathcal{F}\right),
$$

the total computational complexity of the PCG-II method is

$$
\text{number of PCG-II iterations} \times \left(T_A + (4L + 1)N^d\right) + \mathcal{R},
$$

where

$$
L = \begin{cases} 
\max \left\{(t_{11} + 2)(2t_{11} + 1), t_{12}(2t_{12} + 5)\right\}, & \text{in the 2D case,} \\
\max \left\{(2t_{11} + 1)(2t_{11} + 5)(t_{11} + 2), (2t_{12} + 5)(2t_{12} + 1)(t_{12} + 2), t_{13}(2t_{13} + 5)(2t_{13} + 5)\right\}, & \text{in the 3D case,}
\end{cases}
$$

and $\mathcal{F}, T_A, \mathcal{R}$ are the complexity of the fast direct solver for the Poisson equation, one matrix-vector multiplication by $A$, and ILU(0) factorization process, respectively.

Figure 1 presents that PCG-I (upper red line) costs higher than PCG-II (lower blue line) in terms of operation counts. Specifically, the computational complexity of PCG-II is 22.3%
Fig. 1 Total complexity of two algorithms for Example 4 in the log-log scale. Left: case (a). Right: case (b)

Table 13 Iteration counts for Example 5

| N  | 32  | 64  | 128 | 256 | 512 |
|----|-----|-----|-----|-----|-----|
| d 2 | γ = 1 | 9  | 15 | 23 | 22 | 22 |
|    | γ = 100 | 7 | 10 | 10 | 10 | 10 |
| d 3 | γ = 1 | 9 | 14 | 21 | 22 | 22 |
|    | γ = 100 | 8 | 10 | 10 | 10 | 10 |

Table 14 Iteration counts for Example 6 in the 2D case

| N  | 32  | 64  | 128 | 256 | 512 |
|----|-----|-----|-----|-----|-----|
| PCG-II | t1 = 5, t2 = 0 | 11 | 16 | 22 | 29 | 29 |
|      | t1 = 2 log2 N−5, t2 = 0 | 11 | 10 | 10 | 6  | 5  |

of that of PCG-I when N = 64 as is shown in the left figure. And the complexity of PCG-II is 61.4% of that of PCG-I when N = 80 as is shown in the right figure.

Example 5 As a special case of problem (3.1), the following Poisson equation with the Dirichlet boundary condition is solved by the PCG method with the ILU(0) preconditioning:

\[ -\Delta u + \gamma u = f, \quad x \in \Omega = (-1, 1)^d, \quad d = 2, 3, \]

\[ u|_{\partial \Omega} = 0, \]

where \( \gamma \geq 0 \) is a constant.

Table 13 reports the iteration steps of the PCG method which are the constants when the discretization parameter N is large. Besides, the ILU(0) factorization of the resulting matrices from the Legendre-Galerkin method for problem (5.3) costs \( O(N^{d^2}) \) operations, thus the total complexity of PCG method for solving the linear system is of \( O(N^d) \).

Example 6 The problem (3.1) in two dimensions takes the following coefficients:

\[ \beta(x) = [(1 - x)^{25/4} + 5][(1 - y)^{25/4} + 5], \quad \alpha(x) = 0. \]

For this case, the function \( \beta(x) \in Q^6 \). Consequently, it follows from (4.11) that \( t_1 = O(N^{8/11}) \) theoretically. Table 14 shows that the number of iterations of the PCG-II
method with $t_1$ of order $O(\log_2 N)$ decrease, and the number of iterations of the PCG-II method with the constant $t_1$ remain unchangeable when the discretization parameter $N$ is large, which indicates the constant $t_1$ still achieves the effect that the number of iterations of the PCG method is independent of the discretization parameter $N$.

6 Conclusion

An efficient preconditioner $M$ for the PCG method is proposed for the linear system arising from the Legendre-Galerkin method of second-order elliptic equations. Since the iteration steps of the PCG method increase slightly as the discretization parameter $N$ increases, matrix-vector multiplications can be evaluated in $O(N^d (\log N)^2)$ operations, and the complexity of approximately solving the system with a preconditioner $M$ is of $O(N^d (\log_2 N)^{2d})$, where the preconditioner $M$ is constructed by using the $(T+1)$-term Legendre polynomials in each direction to approximate the variable coefficient functions, $T$ is independent of $N$ in one dimension; and is set to be of order $O(\log_2 N)$ in two and three dimensions, the algorithm admits an $O(N^d (\log_2 N)^{2d})$ computational complexity for $d = 1, 2, 3$ while providing spectral accuracy. Furthermore, numerical results indicate that it is very robust.

Acknowledgements We acknowledge the support of the National Natural Science Foundation of China (NSFC 11625101 and 11421101).

References

1. Auteri, F., Quartapelle, L.: Galerkin-Legendre spectral method for the 3D Helmholtz equation. J. Comput. Phys. 161, 454–483 (2000)
2. Canuto, C., Quarteroni, A.: Preconditioner minimal residual methods for Chebyshev spectral calculations. J. Comput. Phys. 60, 315–337 (1985)
3. Canuto, C., Hussaini, M.Y., Quarteroni, A., Zang, T.A.: Spectral Methods: Fundamentals in Single Domains. Springer-Verlag, Berlin, Heidelberg (2006)
4. Canuto, C., Hussaini, M.Y., Quarteroni, A., Zang, T.A.: Spectral Methods in Fluid Dynamics. Springer-Verlag, Berlin (1987)
5. Coutsias, E., Hagstrom, T., Hesthaven, J.S., Torres, D.: Integration preconditioners for differential operators in spectral $\tau$-methods. In: Proceedings of the Third International Conference on Spectral and High Order Methods, Houston, TX, pp. 21–38. (1996)
6. Carlitz, L.: The product of two ultraspherical polynomials. Proc. Glasgow Math. Assoc. 5, 76–79 (1961)
7. Concus, P., Golub, G.H.: Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations. SIAM J. Numer. Anal. 10, 1103–1120 (1973)
8. Deville, M.O., Mund, E.H.: Chebyshev pseudospectral solution of second-order elliptic equations with finite element preconditioning. J. Comput. Phys. 60, 517–533 (1985)
9. Deville, M.O., Mund, E.H.: Finite-element preconditioning for pseudospectral solutions of elliptic problems. SIAM J. Sci. Stat. Comput. 11, 311–342 (1990)
10. Fenn, M., Potts, D.: Fast summation based on fast trigonometric transforms at non-equispaced nodes. Numer. Linear Algebra Appl. 12, 161–169 (2005)
11. Fang, Z., Shen, J., Sun, H.: Preconditioning techniques in Chebyshev collocation method for elliptic equations. Inter. J. Numer. Anal. Model. 15, 277–287 (2018)
12. Gottlieb, D., Orszag, S.A.: Numerical Analysis of Spectral Methods: Theory and Applications. In: CBMS-NSF Regional Conference Series in Mathematics, 26, SIAM, Philadelphia (1977)
13. Hesthaven, J.: Integration preconditioning of pseudospectral operators. I. Basic linear operators. SIAM J. Numer. Anal. 35, 1571–1593 (1998)
14. Hale, N., Townsend, A.: A fast, simple, and stable Chebyshev-Legendre transform using an asymptotic formula. SIAM J. Sci. Comput. 36, A148–A167 (2014)
15. Hale, N., Townsend, A.: A fast FFT-based discrete Legendre transform. IMA J. Numer. Anal. 36, 1670–1684 (2016)
16. Iserles, A.: A fast and simple algorithm for the computation of Legendre coefficients. Numer. Math. 117, 529–553 (2011)
17. Kim, S.D., Parter, S.V.: Preconditioning Chebyshev spectral collocation method for elliptic partial differential equations. SIAM J. Numer. Anal. 33, 2375–2400 (1996)
18. Kim, S.D., Parter, S.V.: Preconditioning Chebyshev spectral collocation by finite difference operators. SIAM J. Numer. Anal. 34, 939–958 (1997)
19. Keimer, J., Kunis, S., Potts, D.: Using NFFT 3-a software library for various nonequispaced fast Fourier transforms. ACM Trans. Math. Softw. 36, 1–30 (2009)
20. Kunis, S.: Nonequispaced fast Fourier transforms without oversampling. PAMM 8, 10977–10978 (2008)
21. Potts, D.: Fast algorithms for discrete polynomial transforms on arbitrary grids. Linear Algebra Appl. 366, 353–370 (2003). (Special issue on structured matrices: analysis, algorithms and applications (Cortona, 2000))
22. Shen, J.: Efficient spectral-Galerkin methods III: polar and cylindrical geometries. SIAM J. Sci. Comput. 18, 1583–1604 (1997)
23. Shen, J.: Efficient Chebyshev-Legendre Galerkin methods for elliptic problems. In: Ilin, A.V., Scott, R. (eds.) Proceedings of the ICOSAHOM’95, Houston J. Math., pp. 233–240. (1996)
24. Shen, J., Wang, Y.W., Xia, J.L.: Fast structured direct spectral methods for differential equations with variable coefficients. SIAM J. Sci. Comput. 38, A28–A54 (2016)
25. Shen, J., Tang, T.: Spectral and High-Order Methods with Applications. Science Press of China, Beijing (2006)
26. Shen, J., Tang, T., Wang, L.L.: Spectral Methods: Algorithms, Analysis and Applications. Springer Series in Computational Mathematics, Springer, Berlin (2011)
27. Shen, J., Wang, F., Xu, J.: A finite element multigrid preconditioner for chebyshev collocation methods. Appl. Numer. Math. 33, 471–477 (2000)
28. Shen, J.: On fast direct Poisson solver, inf-sup constant and iterative Stokes solver by Legendre Galerkin method. J. Comput. Phys. 116, 184–188 (1995)
29. Shen, J.: Efficient spectral-Galerkin method I. Direct solvers for second- and fourth-order equations using Legendre polynomials. SIAM J. Sci. Comput. 15, 1489–1505 (1994)
30. Swarztrauber, P.N.: A direct method for the discrete solution of separable elliptic equations. SIAM J. Numer. Anal. 11, 1136–1150 (1974)
31. Saad, Y.: Iterative Methods for Sparse Linear Systems, 3rd edn. PWS Pub Co., (2000)
32. Tygert, M.: Recurrence relations and fast algorithms. Appl. Comput. Harmon. Anal. 28, 121–128 (2010)
33. Trefethen, L.N., Trummer, M.R.: An instability phenomenon in spectral methods. SIAM J. Numer. Anal. 24, 1008–1023 (1987)
34. Wang, L.L., Samson, M.D., Zhao, X.: A well-conditioned collocation method using a pseudospectral integration matrix. SIAM J. Sci. Comput. 36, A907–A929 (2014)
35. Wang, H., Xiang, S.: On the convergence rates of Legendre approximation. Math. Comput. 81, 861–877 (2011)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.