An Efficient Newton Multiscale Multigrid Method for 2D Semilinear Poisson Equations

Ming Li\textsuperscript{1,2}, Zhoushun Zheng\textsuperscript{2}, Kejia Pan\textsuperscript{2,*} and Xiaoqiang Yue\textsuperscript{3}

\textsuperscript{1}Department of Mathematics, Honghe University, Yunnan 661100, China.
\textsuperscript{2}School of Mathematics and Statistics, Central South University, Changsha 410083, China.
\textsuperscript{3}Hunan Key Laboratory for Computation and Simulation in Science and Engineering, Key Laboratory of Intelligent Computing & Information Processing of Ministry of Education, School of Mathematics and Computational Science, Xiangtan University, Xiangtan 411105, China.

Received 9 January 2020; Accepted (in revised version) 26 March 2020.

Abstract. An efficient Newton multiscale multigrid (Newton-MSMG) for solving large nonlinear systems arising in the fourth-order compact difference discretisation of 2D semilinear Poisson equations is presented. The Newton-MG method is employed to calculate approximation solutions on coarse and fine grids and then a completed Richardson extrapolation is used to construct a sixth-order extrapolated solution on the entire fine grid directly. The method is applied to two nonlinear Poisson-Boltzmann equations and numerical simulations show that the Newton-MSMG method is a cost-effective approach with the sixth-order accuracy.

AMS subject classifications: 65N30, 65N55

Key words: Semilinear Poisson equation, Richardson extrapolation, sixth-order accuracy, Newton’s method, multiscale multigrid.

1. Introduction

Numerical solution of Poisson equations plays an important role in electrostatics and mechanical engineering. In particular, the multigrid (MG) methods combined with Richardson extrapolation strategies are fast and provide high-accuracy solutions of Poisson equations. Thus Chen et al. [2] and Hu et al. [9,10] considered an extrapolation cascadic multigrid (EXCMG) algorithm based on finite element (FE) error expansions. In the EXCMG method, an extrapolation operator for the conjugate gradient (CG) method is developed to construct an accurate initial guess — i.e. a high-order approximation of an FE solution on the next finer grid. This greatly accelerate the convergence of the original cascadic multigrid algorithm. Later on, Pan et al. [14,15] and Li et al. [11,12] extended the EXCMG al-

*Corresponding author. Email address: kejiapan@csu.edu.cn (K.J. Pan)
algorithm to high-order compact difference schemes for 2D and 3D Poisson equations. Wang and Zhang [23] proposed a multiscale multigrid (MSMG) method for 2D Poisson equations. The MSMG method employs a multigrid V-cycle procedure to compute fourth-order accurate solutions on the fine and coarse grids. Using an iterative refinement procedure with the Richardson extrapolation technique, it generates a sixth-order accurate solution on the fine grid. Numerical simulations show the efficacy and high accuracy of the MSMG method for finding solutions of Poisson and convection-diffusion equations [3, 5, 6, 24]. Recently, Dai et al. [4] applied the EXCMG method to determine an initial guess for the original MSMG method. They also introduced an EXCMG accelerated multiscale multigrid (EXCMG-MSMG) method, which is more efficient than the MSMG method.

Although such extrapolation multigrid methods can simultaneously achieve high accuracy and high efficiency, they are only applicable to linear problems. On the other hand, the existing solvers for nonlinear problems, such as Newton multigrid (Newton-MG) method [1], cascadic multigrid (CMG) methods [18, 21, 26, 29] and adaptive multigrid methods [8, 25], cannot achieve the sixth-order accuracy. Therefore, it is important to generalise the extrapolation multigrid algorithms to nonlinear problems.

In this work, we employ fourth-order compact difference schemes and develop a Newton multiscale multigrid (Newton-MSMG) method, which achieves the sixth-order accuracy in solving 2D semilinear Poisson equations. Starting with the Newton-MG method, we determine fourth-order accurate solutions on coarse and fine grids, and then use the completed Richardson extrapolation to generate sixth-order accurate solutions on the entire fine grid directly.

The outline of our presentation is as follows. Section 2 introduces fourth-order compact difference schemes for 2D semilinear Poisson equations. A completed Richardson extrapolation operator is considered in Section 3. A Newton-MSMG method for the semilinear Poisson equation is described in Section 4. Numerical results in Section 5 show the efficiency and the accuracy of the method. Some conclusions are given in the final section.

2. Fourth-Order Compact Difference Schemes

We consider the following 2D semilinear Poisson equation

\[ u_{xx}(x, y) + u_{yy}(x, y) = g(u, x, y), \quad (x, y) \in \Omega, \]  

where the nonlinear forcing function \( g(u, x, y) \) and unknown function \( u(x, y) \) are assumed to be continuously differentiable and have required partial derivatives. We impose the Dirichlet boundary condition on \( \partial \Omega \).

For simplicity, we assume that \( \Omega \) is a rectangular domain \([L_a, L_b] \times [L_c, L_d]\) subdivided into uniform grid \( \Omega_h \) with mesh sizes

\[ h_x = \frac{1}{N_x} (L_b - L_a), \quad h_y = \frac{1}{N_y} (L_d - L_c), \]

where \( N_x \) and \( N_y \) are the numbers of uniform intervals in the \( x \) and \( y \) directions, respectively. Let \( U_{i,j} \) denote an approximation of \( u \) at the mesh point \((x_i, y_j)\) with \( x_i = L_a + ih_x \) and \( y_j = L_c + jh_y \).
and \(y_j = L_c + jh_y\). Let \(G_{i,j} = G(U_{i,j}, x_i, y_j)\) and \(\gamma = h_x/h_y\). We now consider the following nine-point fourth-order compact difference schemes for the nonlinear equation (2.1):

Hoc4a (cf. Zhai et al. [27]).

\[
(5 - \gamma^2)(U_{i+1,j} + U_{i-1,j}) + (5\gamma^2 - 1)(U_{i,j+1} + U_{i,j-1}) - 10\left(1 + \gamma^2\right)U_{i,j} \\
+ \frac{1 + \gamma^2}{2}(U_{i+1,j+1} + U_{i+1,j-1} + U_{i-1,j+1} + U_{i-1,j-1}) \\
= \frac{h_x^2}{2}\left[\frac{1}{12}(G_{i+1,j+1} + G_{i-1,j+1} + G_{i+1,j-1} + G_{i-1,j-1}) \\
+ \frac{5}{6}(G_{i+1,j} + G_{i,j+1} + G_{i,j-1} + \frac{25}{3}G_{i,j})\right]. \quad (2.2)
\]

Hoc4b (cf. Refs. [13, 19, 20, 23, 28]).

\[
(5 - \gamma^2)(U_{i+1,j} + U_{i-1,j}) + (5\gamma^2 - 1)(U_{i,j+1} + U_{i,j-1}) - 10\left(1 + \gamma^2\right)U_{i,j} \\
+ \frac{1 + \gamma^2}{2}(U_{i+1,j+1} + U_{i+1,j-1} + U_{i-1,j+1} + U_{i-1,j-1}) \\
= \frac{h_x^2}{2}\left[\frac{1}{2}G_{i+1,j+1} + \frac{1}{2}G_{i-1,j+1} + \frac{1}{2}G_{i+1,j-1} + \frac{1}{2}G_{i-1,j-1} + 10G_{i,j}\right]. \quad (2.3)
\]

Hoc4c (cf. Manohar and Stephenson [13]).

\[
(5 - \gamma^2)(U_{i+1,j} + U_{i-1,j}) + (5\gamma^2 - 1)(U_{i,j+1} + U_{i,j-1}) - 10\left(1 + \gamma^2\right)U_{i,j} \\
+ \frac{1 + \gamma^2}{2}(U_{i+1,j+1} + U_{i+1,j-1} + U_{i-1,j+1} + U_{i-1,j-1}) \\
= \frac{h_x^2}{2}\left[\frac{1}{2}G_{i+1,j+1} + \frac{1}{2}G_{i-1,j+1} + \frac{1}{2}G_{i+1,j-1} + \frac{1}{2}G_{i-1,j-1} + 10G_{i,j}\right]. \quad (2.4)
\]

Hoc4d (cf. Zhai et al. [27]).

\[
(5 - \gamma^2)(U_{i+1,j} + U_{i-1,j}) + (5\gamma^2 - 1)(U_{i,j+1} + U_{i,j-1}) - 10\left(1 + \gamma^2\right)U_{i,j} \\
+ \frac{1 + \gamma^2}{2}(U_{i+1,j+1} + U_{i+1,j-1} + U_{i-1,j+1} + U_{i-1,j-1}) \\
= \frac{h_x^2}{6}\left[G_{i+1,j+1} + G_{i-1,j+1} + G_{i+1,j-1} + G_{i-1,j-1} \\
+ G_{i+1,j} + G_{i-1,j} + G_{i,j+1} + G_{i,j-1} + 28G_{i,j}\right]. \quad (2.5)
\]

Arranging all \(U_{i,j}\) into unknown solution vector \(u_h\) and employing one of the above schemes, we obtain a nonlinear system of equations with mesh size \(h = \max\{h_x, h_y\}\) as follows:

\[
F_h(u_h) = 0, \quad (2.6)
\]

where the notation \(F_h(u_h)\) indicates that the corresponding operator is nonlinear.
Extrapolation technique is an efficient tool that helps to eliminate certain error terms, thus enhancing the precision of computed solutions. Classical Richardson extrapolation \[16\] can get a fourth-order extrapolated solution on the coarse grid by combining two second-order accurate solutions on the fine and coarse grids. Recently, the extrapolation technique has been used to achieve the sixth-order accuracy on entire fine grid for linear elliptic equations with the fourth-order difference solutions on different scale grids with mesh sizes \(h\) and \(h/2\), cf. Refs. \([3, 5, 6, 11, 14, 15, 20, 23]\).

To avoid using too many grid points and to reduce computational work on the auxiliary coarse grid, the mesh size of the fine grid is set to one third of the coarse grid — cf. Fig. 1. For \(d\)-dimensional problems, the ratio of cells on the coarse to fine grids is \(1/3^d\). Thus, the number of nonlinear equations on the coarse grid is much smaller than for usual coarse grids, so that the corresponding nonlinear system on the coarse grid is easier to solve.

For simplicity, here we only consider the completed Richardson extrapolation strategy on the set \(I_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]\) and construct extrapolated solutions
\[
\{Eu_{s,t}^h, Eu_{i+k/3,t}^h, Eu_{s,j+v/3}^h, Eu_{i+k/3,j+v/3}^h; s = i, i + 1, t = j, j + 1, k, v = 1, 2\} \quad \text{on } \Omega_h
\]
from the coarse grid solutions
\[
\{u_{s,t}^H; s = i, i + 1, t = j, j + 1\} \quad \text{on } \Omega_H
\]
and the fine grid solutions
\[
\{u_{s,t}^h, u_{i+k/3,t}^h, u_{s,j+v/3}^h, u_{i+k/3,j+v/3}^h; s = i, i + 1, t = j, j + 1, k, v = 1, 2\} \quad \text{on } \Omega_h,
\]
where \(u_{s,t}^H\) and \(u_{s,t}^h\) are fourth-order accurate solutions \(u^H\) and \(u^h\) at the points \((x_s, y_t)\), respectively.

![Figure 1: Coarse and refined grids.](image)
3.1. Extrapolation formula at the coarse grid points

According to the fourth-order compact schemes (2.2)-(2.5), we assume that the error expansions can be written as

\[ u - u^H = A(x, y)H^4 + O(H^6), \]  
\[ u - u^h = \frac{1}{81}A(x, y)H^4 + O(H^6), \]  

(3.1)  
(3.2)

where \( A(x, y) \) is a sufficiently smooth function independent of \( H \).

Eliminating \( A(x, y) \) in (3.1) and (3.2) yields

\[ u_{s,t} = \frac{81}{80}u^h_{s,t} - \frac{1}{80}u^H_{s,t} + O(H^6). \]

We temporarily omit the high-order term \( \theta(H^6) \) and write the classical Richardson extrapolation formula [16, 17] at the coarse grid nodes \( (x_s, y_t) \) as

\[ Eu_{s,t} = \frac{81}{80}u^h_{s,t} - \frac{1}{80}u^H_{s,t}, \quad s = i, i + 1, \quad t = j, j + 1. \]  

(3.3)

3.2. Extrapolation formula at the trisection points

Now we consider extrapolation formulas for the following trisection points

\((x_s, y_{j+i/3}), (x_{i+k/3}, y_t), (x_{i+k/3}, y_{j+i/3})\), \( s = i, i + 1, \quad t = j, j + 1, \quad k, v = 1, 2. \)

Similar to the representations (3.1) and (3.2), we have

\[ u^h_{s,t} - u^H_{s,t} = \frac{80}{81}A(x_s, y_t)H^4 + O(H^6) \]

and

\[ A(x_s, y_t) = \frac{81}{80H^4}(u^h_{s,t} - u^H_{s,t}) + O(H^2). \]  

(3.4)

It follows from (3.4) and linear interpolation theory that

\[ A_{i,j+1/3} = \frac{2}{3}A_{i,j} + \frac{1}{3}A_{i,j+1} + O(H^2) \]

\[ = \frac{27}{80H^4}\left[2(u^h - u^H)_{i, j} + (u^h - u^H)_{i, j+1}\right] + O(H^2). \]  

(3.5)

Substituting (3.5) into (3.2) gives

\[ u_{i,j+1/3} = u^h_{i,j+1/3} + \frac{1}{240}\left[2(u^h - u^H)_{i, j} + (u^h - u^H)_{i, j+1}\right] + O(H^6). \]

Ignoring the high-order term \( \theta(H^6) \), we obtain the following trisection point extrapolation formula for the fine grid points \( (x_s, y_{j+i/3}) \):

\[ Eu_{s,j+1/3} = \frac{81}{80}u^h_{s,j+1/3} - \frac{1}{80}u^H_{s,j+1/3} + \frac{1}{240}\left[2(u^h - u^H)_{s, j} + (u^h - u^H)_{s, j+1}\right], \quad s = i, i + 1. \]  

(3.6)
Similar procedure produces other trisection point extrapolation formulas — viz.

\[
Eu_{s,j+2/3} = u_{s,j+2/3}^h + \frac{1}{240} \left[ (u^h - u^H)_{i,j} + 2(u^h - u^H)_{i,j+1} \right], \quad s = i, i + 1, \\
Eu_{t,i+1/3} = u_{i+1/3,t}^h + \frac{1}{240} \left[ 2(u^h - u^H)_{i,t} + (u^h - u^H)_{i+1,t} \right], \quad t = j, j + 1, \\
Eu_{t,j+2/3} = u_{i+2/3,t}^h + \frac{1}{240} \left[ (u^h - u^H)_{i,t} + 2(u^h - u^H)_{i+1,t} \right], \quad t = j, j + 1.
\]

Besides, since \((x_{i+1/3}, y_{j+1/3}), (x_{i+2/3}, y_{j+2/3})\) and \((x_{i+1/3}, y_{j+2/3}), (x_{i+2/3}, y_{j+1/3})\) can be, respectively, regarded as the trisection points of the line-segments \(AB\) and \(CD\) — cf. Fig. 1(b), we also obtain the following extrapolation formulas:

\[
Eu_{i+1/3,j+2/3} = u_{i+1/3,j+2/3}^h + \frac{1}{240} \left[ 2(u^h - u^H)_{i,j+1} + (u^h - u^H)_{i+1,j} \right], \\
Eu_{i+2/3,j+1/3} = u_{i+2/3,j+1/3}^h + \frac{1}{240} \left[ (u^h - u^H)_{i+1,j} + 2(u^h - u^H)_{i,j+1} \right], \\
Eu_{i+1/3,j+1/3} = u_{i+1/3,j+1/3}^h + \frac{1}{240} \left[ 2(u^h - u^H)_{i,j} + (u^h - u^H)_{i+1,j+1} \right], \\
Eu_{i+2/3,j+2/3} = u_{i+2/3,j+2/3}^h + \frac{1}{240} \left[ (u^h - u^H)_{i,j} + 2(u^h - u^H)_{i+1,j+1} \right].
\]

Incorporating the extrapolation formulas \((3.6)\) and \((3.7)\) with the classic Richardson extrapolation technique \((3.3)\), we derive a completed Richardson extrapolation operator \(Ex\) for finding an extrapolated solution \(Eu^h\) on the entire fine grid \(\Omega_h\), viz.

\[
Eu^h \leftarrow Ex \left( u^H, u^h \right).
\]

**Remark 3.1.** \(Ex\) is a locally linear operator, which can be performed very fast.

### 4. Newton Multigrid Methods

#### 4.1. Classical Newton-MG method

The Newton-MG method \([1, 22]\) proposed by Dembo et al. \([7]\) is an efficient approach for solving systems of nonlinear equations \((2.6)\). The main feature of the Newton-MG method is the use of classical V-cycle multigrid for solving the Jacobian systems in Newton iterations. To be more precise, given an initial guess \(x_0\), the Newton-MG method generates the following sequence of approximate solutions \(\{x_k\}\) of the nonlinear system \(F(x) = 0\).

The numerical results in \([1]\) show that the Newton-MG method is much more efficient than Newton’s method. Applying it to the nonlinear discretisation systems \((2.6)\) in fourth-order compact difference schemes \((2.2)-(2.5)\), we obtain a fourth-order accurate solution. In order to enhance the accuracy, one can employ the completed Richardson extrapolation.
4.2. Newton multiscale multigrid method

We already obtained a sixth-order accurate extrapolation solution $Eu^h$ on fine grid $\Omega_h$ by applying the operator $Ex$ to the fourth-order accurate solutions $u^H$ and $u^h$. But it is difficult to derive accurate numerical solutions $u^H$ and $u^h$ of the nonlinear equations. Therefore, we use Newton-MG method (Algorithm 4.1) to find approximate solutions $u^{H,\ast}$ and $u^{h,\ast}$.

**Algorithm 4.1** Newton-MG method for $F(x) = 0$, cf. Refs. [1, 22].

1: For given $\epsilon$, $k_{\max}$ and $x_0$, set $k = 0$.
2: if $\|F(x_k)\| \geq \epsilon$ then
3: go to Step 7
4: else
5: stop
6: end if
7: Use a few MG V-cycles to get an approximation $\tilde{s}_k$ to Jacobian system $F'(x_k)s_k = -F(x_k)$.
8: Update $x_{k+1} \leftarrow x_k + \tilde{s}_k$, $k \leftarrow k + 1$.
9: if $k < k_{\max}$ then
10: go to Step 2
11: else
12: stop
13: end if

The structure of Algorithm 4.2 is shown in Fig. 2. The hollow circle denotes the initial guess, the green diamond the Newton-MG solver, the blue circle the fourth-order accurate solution computed by the Newton-MG method, and the red square the sixth-order accurate extrapolated solution obtained by the completed Richardson extrapolation operator $Ex$ from approximate solutions $u^{H,\ast}$ and $u^{h,\ast}$.

**Algorithm 4.2** Newton-MSMG method.

1: Calculate the approximate solution $u^{H,\ast}$ by using Newton-MG method with an initial guess $u^{H,0}$.
2: Calculate a good initial guess:

$$u^{h,0} \Leftarrow P^h_H u^{H,\ast},$$

where $P^h_H$ is a high-order interpolation operator.
3: Calculate the approximate solution $u^{h,\ast}$ by using Newton-MG method with an initial guess $u^{h,0}$.
4: Calculate the sixth-order accurate solution by using the completed Richardson extrapolation operator $Ex$:

$$Eu^h \Leftarrow Ex \left( u^{H,\ast}, u^{h,\ast} \right).$$
5. Numerical Experiments

In order to show the accuracy and the efficiency of the Newton-MSMG method, we present numerical results for two nonlinear Poisson-Boltzmann equations on a regular domain. Here, the nonlinear forcing term and Dirichlet boundary conditions on \( \partial \Omega \) are chosen so that the problems have prescribed analytic solutions. All computations are carried out on a desktop with Inter (R) Xeon(R) CPU E5-2609 (2.50GHz) and 16GB RAM using MATLAB in double precision.

Implementing the fourth-order compact difference schemes (2.2)-(2.5), we apply the Newton-MG method \[1, 22\] and the Newton-MSMG method to the test problems. For convenience, each MG method starts with the zero initial guess and is terminated if the Euclid norm of the residual got smaller than \(10^{-10}\).

In Newton-MG method, we use a standard full weighting restriction operator to project residual vectors from the fine grid to the coarse grid and the bilinear interpolation operator to transfer corrections from the coarse grid to the fine grid. The number of pre-smoothing and post-smoothing steps are equal to 3. Besides, we adopt cubic spline interpolation in Newton-MSMG method to provide an initial guess on the fine grid from the approximate solutions on the coarse ones.

The \(L^2\)-norm error between the exact and numerical solutions obtained by Newton-MG or Newton-MSMG methods is denoted by \(\|E\|_2\) and the convergence rate is defined by

\[
\text{Rate} = \frac{\log(\|E_H\|_2/\|E_h\|_2)}{\log(H/h)},
\]

where \(E_H\) and \(E_h\) are the errors for the mesh sizes \(H\) and \(h\), respectively.

Moreover, "Time" and "Iter" refer to the calculation time in seconds and the number of iterations, respectively. We set \(N = N_x = N_y\) as the number of uniform intervals in the \(x\) and \(y\) directions. To be more specific, for Newton-MG method, the two values from left to right correspond to the outer and inner iterations, respectively. For Newton-MSMG method, the four values are the numbers of outer and inner iterations on grids with mesh sizes \(1/N\) and \(3/N\).
Example 5.1. Consider the nonlinear Poisson-Boltzmann equation

\[ u_{xx}(x, y) + u_{yy}(x, y) - \lambda^2 \sinh(u(x, y)) = f(x, y) \]

with the exact solution

\[ u(x, y) = \exp\left(-1000(x - 0.5)^2\right) - y^2 \]

on the unit domain \( \Omega = [0, 1] \times [0, 1] \).

Example 5.2. Consider the nonlinear Poisson-Boltzmann equation

\[ \lambda^2(u_{xx}(x, y) + u_{yy}(x, y)) + \exp(-u(x, y)) = f(x, y) \]

with the exact solution

\[ u(x, y) = \exp(y - x) + \frac{(1 + y)^{101}}{2^{100}} \]

on the unit domain \( \Omega = [0, 1] \times [0, 1] \).

Various data concerning the solutions of the above problems obtained by Newton-MG and Newton-MSMG methods are presented in Tables 1-8. We note that the finest grid approximate solutions obtained by Newton-MSMG method are much more accurate. Thus in order to achieve the same accuracy, the Newton-MSMG method uses less grid points than the other one. For instance, the numerical solution obtained by Newton-MSMG method on the grid with \( N = 384 \) is comparable to the solution on the grid with \( N = 768 \) obtained by Newton-MG method — cf. Table 1. Furthermore, it is easily seen that the Newton-MG method has the fourth-order accuracy, while the Newton-MSMG algorithm can achieve the sixth-order accuracy.

Table 1: Example 5.1. Hoc4a scheme is used.

| \( \lambda \) | \( N \) | Newton-MG | Newton-MSMG |
|-----|-----|--------|------------|
|     |     | ||\( E \)|| Rate | Iter | Time | ||\( E \)|| Rate | Iter | Time |
| 0.01| 192 | 6.21(-6) | - | 3, 9 | 2.40 | - | 2, 6, 3, 9 | 1.98 |
|     | 384 | 3.85(-7) | 4.01 | 3, 9 | 24.3 | 1.60(-8) | 6.05 | 2, 5, 3, 9 | 21.4 |
|     | 768 | 2.40(-8) | 4.00 | 3, 9 | 299 | 2.48(-10) | 6.01 | 2, 4, 3, 9 | 257 |
|     | 1536| 1.50(-9) | 4.00 | 3, 9 | 3489 | 4.00(-12) | 5.95 | 2, 3, 3, 9 | 2899 |
| 1   | 192 | 6.20(-6) | - | 4, 14 | 2.42 | 1.06(-6) | - | 3, 7, 4, 14 | 2.09 |
|     | 384 | 3.85(-7) | 4.01 | 4, 13 | 25.2 | 1.60(-8) | 6.05 | 2, 5, 4, 14 | 20.2 |
|     | 768 | 2.40(-8) | 4.00 | 4, 14 | 311 | 2.48(-10) | 6.01 | 2, 4, 4, 13 | 214 |
|     | 1536| 3.52(-9) | 2.77 | 3, 12 | 3596 | 4.00(-12) | 5.95 | 2, 3, 4, 13 | 2854 |
| 10  | 192 | 6.21(-6) | - | 5, 19 | 2.73 | 1.05(-6) | - | 3, 8, 5, 19 | 2.32 |
|     | 384 | 3.85(-7) | 4.01 | 5, 20 | 29.4 | 1.58(-8) | 6.05 | 3, 6, 5, 19 | 21.7 |
|     | 768 | 2.40(-8) | 4.00 | 5, 19 | 343.7 | 2.45(-10) | 6.01 | 2, 4, 5, 20 | 237 |
|     | 1536| 1.50(-9) | 4.00 | 5, 19 | 3785 | 3.96(-12) | 5.95 | 2, 3, 5, 20 | 2930 |
Table 2: Example 5.1. Hoc4b scheme is used.

| λ   | N   | \(\|E\|_2\) | Rate | Iter | Time | \(\|E\|_2\) | Rate | Iter | Time |
|-----|-----|--------------|------|------|------|--------------|------|------|------|
| 0.01| 192 | 6.21(-6)     | 3, 9 | 2.55 | 1.06(-6) | 3, 9 | 2.13 |
|     | 384 | 3.85(-7)     | 4.01 | 20.3 | 1.60(-8) | 6.05 | 18.6 |
|     | 768 | 2.40(-8)     | 3.9  | 305  | 2.48(-10) | 6.01 | 274  |
|     | 1536| 1.50(-9)     | 4.00 | 3352 | 4.00(-12) | 5.95 | 2805 |
| 1   | 192 | 6.20(-6)     | 4, 14| 2.53 | 1.06(-6) | 3, 7, 4, 14| 2.04 |
|     | 384 | 3.85(-7)     | 4, 13| 23.4 | 1.60(-8) | 2, 5, 4, 14| 18.5 |
|     | 768 | 2.40(-8)     | 4, 14| 324  | 2.48(-10) | 2, 4, 13, 297 |
|     | 1536| 3.52(-9)     | 2.77 | 3479 | 4.00(-12) | 2, 3, 4, 13 | 2816 |
| 10  | 192 | 6.11(-6)     | 5, 19| 3.03 | 1.05(-6) | 3, 8, 5, 19 | 2.32 |
|     | 384 | 3.79(-7)     | 4.01 | 26.8 | 1.58(-8) | 3, 6, 5, 19 | 19.4 |
|     | 768 | 2.36(-8)     | 5, 19| 346  | 2.45(-10) | 2, 4, 5, 20 | 251  |
|     | 1536| 1.48(-9)     | 4.00 | 3662 | 3.96(-12) | 2, 3, 5, 20 | 2973 |

Table 3: Example 5.1. Hoc4c scheme is used.

| λ   | N   | \(\|E\|_2\) | Rate | Iter | Time | \(\|E\|_2\) | Rate | Iter | Time |
|-----|-----|--------------|------|------|------|--------------|------|------|------|
| 0.01| 192 | 6.21(-6)     | 3, 9 | 2.55 | 1.06(-6) | 3, 9 | 2.13 |
|     | 384 | 3.85(-7)     | 4.01 | 20.3 | 1.60(-8) | 6.05 | 18.6 |
|     | 768 | 2.40(-8)     | 3.9  | 305  | 2.48(-10) | 6.01 | 274  |
|     | 1536| 1.50(-9)     | 4.00 | 3352 | 4.00(-12) | 5.95 | 2805 |
| 1   | 192 | 6.20(-6)     | 4, 14| 2.53 | 1.06(-6) | 3, 7, 4, 14| 2.04 |
|     | 384 | 3.85(-7)     | 4, 13| 23.4 | 1.60(-8) | 2, 5, 4, 14 | 18.5 |
|     | 768 | 2.40(-8)     | 4, 14| 324  | 2.48(-10) | 2, 4, 13, 297 |
|     | 1536| 3.52(-9)     | 2.77 | 3479 | 4.00(-12) | 2, 3, 4, 13 | 2816 |
| 10  | 192 | 6.11(-6)     | 5, 19| 3.03 | 1.05(-6) | 3, 8, 5, 19 | 2.32 |
|     | 384 | 3.79(-7)     | 4.01 | 26.8 | 1.58(-8) | 3, 6, 5, 19 | 19.4 |
|     | 768 | 2.36(-8)     | 5, 19| 346  | 2.45(-10) | 2, 4, 5, 20 | 251  |
|     | 1536| 1.48(-9)     | 4.00 | 3662 | 3.96(-12) | 2, 3, 5, 20 | 2973 |

Table 4: Example 5.1. Hoc4d scheme is used.

| λ   | N   | \(\|E\|_2\) | Rate | Iter | Time | \(\|E\|_2\) | Rate | Iter | Time |
|-----|-----|--------------|------|------|------|--------------|------|------|------|
| 0.01| 192 | 6.21(-6)     | 3, 9 | 2.35 | 1.06(-6) | 3, 9 | 2.06 |
|     | 384 | 3.85(-7)     | 4.01 | 21.9 | 1.60(-8) | 6.05 | 19.8 |
|     | 768 | 2.40(-8)     | 3.9  | 282  | 2.48(-10) | 6.01 | 264  |
|     | 1536| 1.50(-9)     | 4.00 | 3348 | 4.00(-12) | 5.95 | 2836 |
| $\lambda$ | $N$ | $\|E\|_2$ | Rate | Iter | Time  | $\|E\|_2$ | Rate | Iter | Time  |
|----------|-----|------------|------|------|-------|------------|------|------|-------|
| 0.01     | 192 | 1.87(-6)   | 3.99 | 7, 33| 345   | 1.87(-7)   | 5.77 | 3, 9, 8, 25 | 2.02  |
|          | 384 | 1.17(-7)   | 4.00 | 7, 35| 34.7  | 3.04(-9)   | 5.94 | 3, 7, 7, 31 | 19.3  |
|          | 768 | 7.33(-9)   | 4.00 | 7, 35| 275   | 4.86(-11)  | 5.99 | 2, 5, 7, 35 | 203   |
|          | 1536| 4.62(-10)  | 3.99 | 7, 35| 3268  | 7.91(-13)  | 5.94 | 2, 4, 7, 35 | 2704  |
| 1        | 192 | 1.22E-05   | 3.99 | 4, 15| 2.78  | 3.39(-7)   | 5.88 | 3, 8, 4, 16 | 1.89  |
|          | 384 | 7.65(-7)   | 4.00 | 4, 15| 24.9  | 5.43(-9)   | 5.96 | 2, 5, 4, 15 | 19.8  |
|          | 768 | 4.78(-8)   | 4.00 | 4, 15| 232   | 8.52(-11)  | 5.99 | 2, 5, 4, 15 | 175   |
|          | 1536| 2.99(-9)   | 4.00 | 4, 15| 3026  | 1.35(-12)  | 5.98 | 2, 4, 4, 15 | 2693  |
| 10       | 192 | 1.23E-05   | 3.99 | 3, 11| 2.12  | 3.40(-7)   | 5.88 | 2, 6, 3, 11 | 166   |
|          | 384 | 7.71(-7)   | 4.00 | 3, 11| 2.9   | 5.44(-9)   | 5.97 | 2, 5, 3, 11 | 18.2  |
|          | 768 | 4.82(-8)   | 4.00 | 3, 11| 203   | 8.55(-11)  | 5.99 | 2, 5, 3, 11 | 184   |
|          | 1536| 3.01(-9)   | 4.00 | 3, 11| 3015  | 1.34(-12)  | 6.00 | 2, 4, 3, 11 | 2713  |

Table 5: Example 5.2. Hoc4a scheme is used.

| $\lambda$ | $N$ | $\|E\|_2$ | Rate | Iter | Time  | $\|E\|_2$ | Rate | Iter | Time  |
|----------|-----|------------|------|------|-------|------------|------|------|-------|
| 0.01     | 192 | 1.87(-6)   | –    | 7, 32| 3.69  | 1.87(-7)   | –    | 3, 9, 8, 25 | 1.93  |
|          | 384 | 1.17(-7)   | 4.00 | 7, 35| 35.4  | 3.04(-9)   | 5.94 | 3, 7, 7, 31 | 18.6  |
|          | 768 | 7.33(-9)   | 4.00 | 7, 35| 238   | 4.86(-11)  | 5.97 | 2, 5, 7, 35 | 189   |
|          | 1536| 4.62(-10)  | 3.99 | 7, 35| 2991  | 7.91(-13)  | 5.94 | 2, 4, 7, 35 | 2813  |
| 1        | 192 | 1.22E-05   | –    | 4, 15| 2.71  | 3.39(-7)   | –    | 3, 8, 4, 16 | 1.92  |
|          | 384 | 7.65(-7)   | 4.00 | 4, 15| 22.4  | 5.43(-9)   | 5.96 | 2, 5, 4, 15 | 18.5  |
|          | 768 | 4.78(-8)   | 4.00 | 4, 15| 202   | 8.52(-11)  | 5.99 | 2, 5, 4, 15 | 177   |
|          | 1536| 2.99(-9)   | 4.00 | 4, 15| 2873  | 1.35(-12)  | 5.98 | 2, 4, 4, 15 | 2689  |
| 10       | 192 | 1.23E-05   | –    | 3, 11| 1.94  | 3.40(-7)   | –    | 2, 6, 3, 11 | 1.57  |
|          | 384 | 7.71(-7)   | 4.00 | 3, 11| 20.8  | 5.44(-9)   | 5.97 | 2, 5, 3, 11 | 15.6  |
|          | 768 | 4.82(-8)   | 4.00 | 3, 11| 187   | 8.55(-11)  | 5.99 | 2, 5, 3, 11 | 172   |
|          | 1536| 3.01(-9)   | 4.00 | 3, 11| 2802  | 1.33(-12)  | 6.00 | 2, 4, 3, 11 | 2667  |

Table 6: Example 5.2. Hoc4b scheme is used.
An Efficient Newton-MSMG Method for 2D Semilinear Poisson Equations

Table 7: Example 5.2. Hoc4c scheme is used.

| \( \lambda \) | \( N \) | \( N \) | \( \|E\|_2 \) | Rate | Iter | Time | \( \|E\|_2 \) | Rate | Iter | Time |
|--------------|-------|-------|----------|------|------|-----|----------|------|------|-----|
| 0.01         | 192   | 1.87(-6) | – | 7, 33 | 3.23 | 1.87(-7) | – | 3, 10, 8, 25 | 1.68 |
|              | 384   | 1.17(-7) | 4.00 | 7, 35 | 33.2 | 3.04(-9) | 5.94 | 2, 5, 7, 31 | 16.4 |
|              | 768   | 7.33(-9) | 4.00 | 7, 35 | 223 | 4.86(-11) | 5.97 | 2, 5, 7, 35 | 167 |
|              | 1536  | 4.62(-10) | 3.99 | 7, 35 | 2835 | 7.91(-13) | 5.94 | 2, 4, 7, 35 | 2627 |
| 1           | 192   | 1.22E-05 | – | 4, 15 | 2.02 | 3.39(-7) | – | 3, 8, 4, 16 | 1.67 |
|              | 384   | 7.65(-7) | 4.00 | 4, 15 | 21.1 | 5.43(-9) | 5.96 | 2, 5, 4, 16 | 15.7 |
|              | 768   | 4.78(-8) | 4.00 | 4, 15 | 188 | 8.52(-11) | 5.99 | 2, 5, 4, 15 | 171 |
|              | 1536  | 2.99(-9) | 4.00 | 4, 15 | 2736 | 1.35(-12) | 5.98 | 2, 4, 4, 15 | 2674 |
| 10          | 192   | 1.23E-05 | – | 3, 11 | 1.75 | 3.40(-7) | – | 2, 6, 3, 12 | 1.40 |
|              | 384   | 7.71(-7) | 4.00 | 3, 11 | 19.1 | 5.44(-9) | 5.97 | 2, 5, 3, 11 | 15.6 |
|              | 768   | 4.82(-8) | 4.00 | 3, 11 | 179 | 8.55(-11) | 5.99 | 2, 5, 3, 11 | 169 |
|              | 1536  | 3.01(-9) | 4.00 | 3, 11 | 2704 | 1.33(-12) | 6.00 | 2, 4, 3, 11 | 2674 |

Table 8: Example 5.2. Hoc4d scheme is used.

| \( \lambda \) | \( N \) | \( \|E\|_2 \) | Rate | Iter | Time | \( \|E\|_2 \) | Rate | Iter | Time |
|--------------|-------|----------|------|------|-----|----------|------|------|-----|
| 0.01         | 192   | 1.87(-6) | – | 7, 33 | 3.15 | 1.87(-7) | – | 3, 10, 8, 25 | 1.78 |
|              | 384   | 1.17(-7) | 4.00 | 7, 35 | 29.4 | 3.04(-9) | 5.94 | 2, 5, 7, 31 | 16.5 |
|              | 768   | 7.33(-9) | 4.00 | 7, 35 | 217 | 4.86(-11) | 5.97 | 2, 5, 7, 35 | 169 |
|              | 1536  | 4.62(-10) | 3.99 | 7, 35 | 2832 | 7.91(-13) | 5.94 | 2, 4, 7, 35 | 2656 |
| 1           | 192   | 1.22E-05 | – | 4, 15 | 2.12 | 3.39(-7) | – | 3, 8, 4, 16 | 1.64 |
|              | 384   | 7.65(-7) | 4.00 | 4, 15 | 20.8 | 5.43(-9) | 5.96 | 2, 5, 4, 15 | 13.8 |
|              | 768   | 4.78(-8) | 4.00 | 4, 15 | 187 | 8.52(-11) | 5.99 | 2, 5, 4, 15 | 171 |
|              | 1536  | 2.99(-9) | 4.00 | 4, 15 | 2736 | 1.36(-12) | 5.97 | 2, 4, 4, 15 | 2665 |
| 10          | 192   | 1.23E-05 | – | 3, 11 | 1.86 | 3.40(-7) | – | 2, 6, 3, 12 | 1.44 |
|              | 384   | 7.71(-7) | 4.00 | 3, 11 | 18.7 | 5.44(-9) | 5.97 | 2, 5, 3, 11 | 14.3 |
|              | 768   | 4.82(-8) | 4.00 | 3, 11 | 179 | 8.55(-11) | 5.99 | 2, 5, 3, 11 | 168 |
|              | 1536  | 3.01(-9) | 4.00 | 3, 11 | 2703 | 1.33(-12) | 6.00 | 2, 4, 3, 11 | 2660 |

It is more important that on the finest grid, the Newton-MSMG method requires a fewer number of iterations. Consequently, it spends less time than the Newton-MG method. Fig. 3 shows the \( L^2 \)-norm error and the CPU time for two algorithms combined with Hoc4a scheme. Hence, Newton-MSMG method is a more cost-effective method.
6. Conclusion

We present an efficient Newton-MSMG method combined with fourth-order compact difference schemes for 2D semilinear Poisson equations. The Newton-MG method is employed to calculate approximation solutions on two grids with different mesh sizes. Then, a completed Richardson extrapolation is adopted to construct a sixth-order extrapolated solution on the entire fine grid directly. The method is applied to two nonlinear Poisson-Boltzmann equations and numerical simulations show that the Newton-MSMG method is a cost-effective approach with the sixth-order accuracy.

It is worth noting that our method can be extended to 3D semilinear partial differential equations with variable coefficients.

Acknowledgments

Ming Li was supported by the Natural Science Foundation of Yunnan Province of China (2017FH001-012, 2017FH001-015). Zhoushun Zheng was supported by the National Natural Science Foundation of China (51974377) and by the National Key Research and Development Program of China (2017YFB0701700, 2017YFB0305601). Kejia Pan was supported by the Science Challenge Project (TZ2016002), the National Natural Science Foundation of China (41874086), the Innovation-Driven Project of Central South University (2018CX042) and by the Excellent Youth Foundation of Hunan Province of China (2018JJ1042). Xiaoqiang Yue was supported by the Project of Scientific Research Fund of Hunan Provincial Science and Technology Department (2018WK4006).

References

[1] W.L. Briggs, V.E. Henson and S. F. McCormick, A Multigrid Tutorial, SIAM (2000).
An Efficient Newton-MSMG Method for 2D Semilinear Poisson Equations

[2] C.M. Chen, H.L. Hu, Z.Q. Xie and C.L. Li, Analysis of extrapolation cascadic multigrid method (EXCMG), Sci. China Ser. A-Math. 51, 1349-1360 (2008).

[3] R.X. Dai, P.P. Lin and J. Zhang, An efficient sixth-order solution for anisotropic Poisson equation with completed Richardson extrapolation and multiscale multigrid method, Comput. Math. Appl. 73, 1865-1877 (2017).

[4] R.X. Dai, P.P. Lin and J. Zhang, An EXCMG accelerated multiscale multigrid computation for 3D Poisson equation, Comput. Math. Appl. 77, 2051-2060 (2019).

[5] R.X. Dai, Y. Wang and J. Zhang, Fast and high accuracy multiscale multigrid method with multiple coarse grid updating strategy for the 3D convection-diffusion equation, Comput. Math. Appl. 66, 542-559 (2013).

[6] R.X. Dai, J. Zhang and Y. Wang, Multiple coarse grid acceleration for multiscale multigrid computation, J. Comput. Appl. Math. 269, 75-85 (2014).

[7] J.E. Dennis and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, SIAM (1996).

[8] B. Gong, J.Y. Han, J.G. Sun and Z.M. Zhang, A shifted-inverse adaptive multigrid method for the elastic eigenvalue problem, Commun. Comput. Phys. 27, 251-273 (2020).

[9] H.L. Hu, C.M. Chen and K.J. Pan, Asymptotic expansions of finite element solutions to Robin problems in $H^3$ and their application in extrapolation cascadic multigrid method, Sci. China Math. 57, 687-698 (2014).

[10] H.L. Hu, Y. Wang and J. Zhang, Multiple coarse grid acceleration for multiscale multigrid computation based on the fourth-order compact difference scheme, Math. Meth. Appl. Sci. 41, 920-928 (2018).

[11] M. Li and C.L. Li, New cascadic multigrid methods for two-dimensional Poisson problem based on the fourth-order compact difference scheme, Math. Meth. Appl. Sci. 41, 920-928 (2018).

[12] M. Li, C.L. Li, X.Z. Cui and J.E. Zhao, Cascadic multigrid methods combined with sixth-order compact scheme for Poisson equation, Numer. Algorithms 71, 715-727 (2016).

[13] R. Manohar and J.W. Stephenson, High order difference schemes for linear partial differential equations, SIAM J. Sci. Stat. Comput. 5, 69-77 (1984).

[14] K.J. Pan, D.D. He and H.L. Hu, An extrapolation cascadic multigrid method combined with a fourth-order compact scheme for 3D Poisson equation, J. Sci. Comput. 70, 1180-1203 (2017).

[15] K.J. Pan, D.D. He, H.L. Hu and Z.Y. Ren, A new extrapolation cascadic multigrid method for three dimensional elliptic boundary value problems, J. Comput. Phys. 344, 499-515 (2017).

[16] L.F. Richardson, The approximate solution of physical problems involving differential equations using finite differences, with an application to the stress in a masonry dam, Philos. Trans. Roy. Soc. London. Ser. A. 210, 307-357 (1910).

[17] L.F. Richardson and J.A. Gaunt, The deferred approach to the limit. Part I: Single lattice, Philos. Trans. Roy. Soc. London. Ser. A 226, 299-349 (1927).

[18] V.V. Shaidurov and G. Timmermann, A cascadic multigrid algorithm for semilinear indefinite elliptic problems, Computing 64, 349-366 (2000).

[19] J.C. Strikwerda, Finite Difference Schemes and Partial Differential Equations, Chapman Hall (1989).

[20] H.W. Sun and J. Zhang, A high order finite difference discretization strategy based on extrapolation for convection diffusion equations, Numer. Meth. Part. Differ. Equ. 20, 18-32 (2004).

[21] G. Timmermann, A cascadic multigrid algorithm for semilinear elliptic problems, Numer. Math. 8, 717-731 (2000).

[22] U. Trottenberg, C.W. Oosterlee and A. Schüller, Multigrid, Academic Press (2001).

[23] Y. Wang and J. Zhang, Sixth-order compact scheme combined with multigrid method and extrapolation technique for 2D Poisson equation, J. Comput. Phys. 228, 137-146 (2009).

[24] Y. Wang and J. Zhang, Integrated fast and high-accuracy computation of convection diffusion
equations using multiscale multigrid method, Numer. Meth. Part. Differ. Equ. 27, 399-414 (2011).

[25] F. Xu, Q.M. Huang, S.S. Chen and T. Bing, An adaptive multigrid method for semilinear elliptic equations, East Asian J. Appl. Math. 9, 683-702 (2019).

[26] H.X. Yu and J.P. Zeng, A cascadic multigrid method for a kind of semilinear elliptic problem, Numer. Algorithms 58, 143-162 (2011).

[27] S.Y. Zhai, X.L. Feng and Y.N. He, A new method deduce high-order compact difference schemes for two-dimensional Poisson equation, Appl. Math. Comput. 230, 9-26 (2014).

[28] J. Zhang, Multigrid method and fourth-order compact difference scheme for 2D Poisson equation with unequal meshsize discretization, J. Comput. Phys. 179, 170-179 (2002).

[29] S.Z. Zhou and H.X. Hu, On the convergence of a cascadic multigrid method for semilinear elliptic problem, Appl. Math. Comput. 159, 407-417 (2004).