A JUMPING MULTIGRID METHOD VIA FINITE ELEMENT EXTRAPOLATION

By

Chuanmiao Chen

Hongling Hu

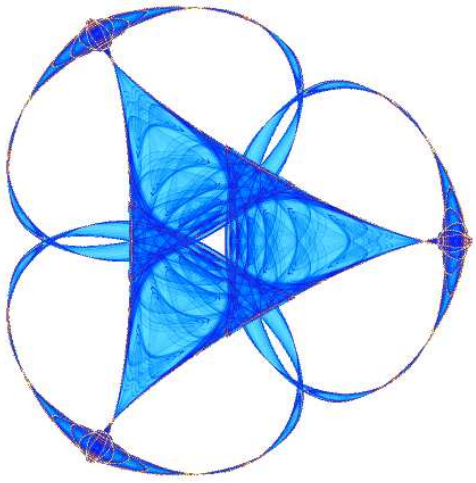
Ziqing Xie

and

Shangyou Zhang

IMA Preprint Series # 2355

(December 2010)



INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

UNIVERSITY OF MINNESOTA

400 Lind Hall

207 Church Street S.E.

Minneapolis, Minnesota 55455–0436

Phone: 612-624-6066 Fax: 612-626-7370

URL: http://www.ima.umn.edu

A jumping multigrid method via finite element extrapolation

Chuanmiao Chen*, Hongling Hu*,†,
Ziqing Xie*,‡ and Shangyou Zhang§

Abstract

The multigrid method solves the finite element equations in optimal order, i.e., solving a linear system of $O(N)$ equations in $O(N)$ arithmetic operations. Based on low level solutions, we can use finite element extrapolation to obtain the high-level finite element solution on some coarse-level element boundary, at an higher accuracy $O(h_i^4)$. Thus, we can solve higher level ($h_j, j \lesssim 2i$) finite element problems locally on each such coarse-level element. That is, we can skip the finite element problem on middle levels, $h_{i+1}, h_{i+2}, \ldots, h_{j-1}$. Roughly speaking, such a jumping multigrid method solves an order $O(N) = O(2^{2di})$ linear system of equations by a memory of $O(\sqrt{N}) = O(2^{di})$, and by a parallel computation of $O(\sqrt{N})$, where d is the space dimension.

Keywords. elliptic equation, finite element, extrapolation, uniform grid, superconvergence.

AMS subject classifications (2000). 65M60, 65N30.

1 Introduction

The multigrid method is an optimal solver which solves a linear system of N unknowns in $O(N)$ arithmetic operations, cf. [1, 15, 17, 18, 19]. To be precise, it solves finite element equations, arising from elliptic boundary value problems, up to the order of truncation error by a work proportional to the number of unknowns. In the multigrid method, we solve a sequence of finite element problems on a sequence of refined grids. In the method, coarse level problems provide initial solutions for finer level problems, and later, correct effectively the low-frequency errors of the iterative solutions on finer levels. The coarse level solutions also contain information of fine level solutions and the exact solution. The traditional extrapolation, based on finite element solutions of a few levels, provides a better approximation of the exact solution. The newly discovered finite element extrapolation, however, provides high order approximation to the fine level finite element solutions, *but not* to the exact solution, cf. [5, 13, 4]. Thus, via such finite element extrapolation, we propose a new multigrid method. In the method, after

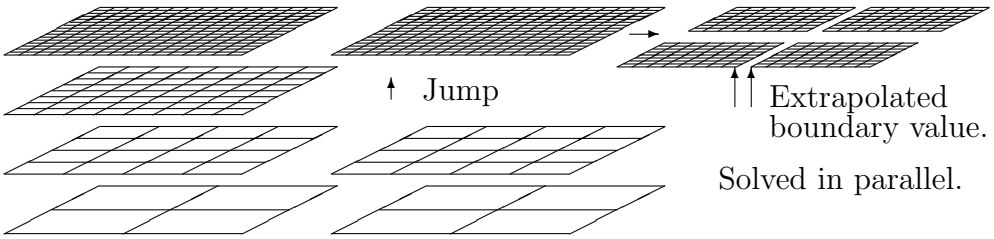


Figure 1: The multigrid method, and jumping multigrid.

solving a few coarse level finite element equations, we skip some levels and jump to the finest level finite element equation. An illustration is shown in Figure 1.

The method is based on the asymptotic expansion of finite element solution at the nodes:

$$u_h(x_j) = u(x_j) + c_1(x_j)h^{2p} + c_2(x_j)h^{2p+2} + o(h^{2p+2}), \quad u \in C^{4+\epsilon},\ p = 1, 2. \tag{1.1}$$

(1.1) requires, in addition, a uniform or locally symmetric grid, cf. [2, 3]. Based on a few low level solutions of the finite element problems, we use a high-order extrapolation to get the nodal value of finite element solutions on a high level. The height of the jump depends on the levels of coarse-level solutions and on the smoothness of the exact solution. After such an extrapolation to get the finite element solution on the inter-element boundary of a low-level grid, we then solve a local problem within each such a square. Different from traditional multigrid method, we do not update the inner-boundary value of the high order finite element solution. Thus, we solve many local problems, on each coarse-level element, in parallel, on the high level. This would significantly reduce the requirement on computer memory. In the best situation, the memory is reduced from $O(N)$ to $O(\sqrt{N})$ for a finite element problem of $O(N)$ unknowns. An example is shown in Table 1, where the $h = 2^{-12}$ solution is obtained from the $h = 2^{-6}$ and $h = 2^{-7}$ solutions. We computed the middle level solutions in the jumping multigrid method for a comparison only, with the traditional multigrid method.

A variation of the jumping multigrid method is to extrapolate the whole fine level solution completely, not only on the inter-element boundary. This way, we do not even solve local finite element problems any more. It would reduce the workload further by a factor of 10 or so, i.e., about 1/10 of the cpu time shown in the last but one column of Table 1. However, this method works less effective when the exact solution is less smooth. In fact, both jumping multigrid methods work less well for problems with singularities. We will show the deficit of the new methods for various singular problems in the numerical test section. In fact, even for an L-shape domain problem, cf. Table 6, where the solution is of $H^{5/2}$ and the asymptotic expansion (1.1) would not hold globally, we can still use the solutions of 2^{-7} and 2^{-8} levels to get a jump solution on the 2^{-13} level reasonably well. This suggests a further research problem. In fact, this is a primary reason for us to study the extrapolation for finite element solutions. By comparing the finite element solution and the extrapolated solution, it gives us a refinement strategy. Such a comparison was done by Fulton [9] before. This would lead to an adaptive finite

Table 1: The Convergence and cpu time, cf. (4.1).

		Multigrid			Jumping multigrid			
h	N	$\|u-u_h\|_{l^\infty}$	$\frac{e_i}{e_{i+1}}$	cpu	$\|u-u_h\|_{l^\infty}$	$\frac{e_i}{e_{i+1}}$	cpu	cpu*
2^{-1}	9	0.21292031						
2^{-2}	25	0.07833587	2.7180		Same as			
2^{-3}	81	0.01768645	4.4291		multigrid			
2^{-4}	289	0.00432559	4.0887					
2^{-5}	1,089	0.00108084	4.0020	0.1				
2^{-6}	4,225	0.00027034	3.9979	0.4				
2^{-7}	16,641	0.00006757	4.0008	2.5	0.00006757	4.0008	2.5	
2^{-8}	66,049	0.00001689	4.0000	10	0.00001687	4.0044	7	
2^{-9}	263,169	0.00000422	4.0000	42	0.00000420	4.0186	21	
2^{-10}	1,050,625	0.00000105	4.0000	175	0.00000103	4.0738	80	.3
2^{-11}	4,198,401	0.00000026	4.0000	761	0.00000024	4.3017	304	1.4
2^{-12}	16,785,409				0.00000005	5.2658	1225	4.8

* Parallel computing.

element method where local grids are uniform and local errors are smoothing. This overcomes the problems of other adaptive methods, where grids are irregular locally, and nonsmooth errors cause excessive but unnecessary local refinements. Previously, implicit extrapolation was introduced to the multigrid method dealing with irregular grids [14]. But we prefer local uniform grids.

It is not new applying extrapolation in the multigrid method. A similar idea, cf. [11, 16], τ-extrapolation, was used in the multigrid method from the very beginning. In the τ-extrapolation multigrid method, an extrapolation is applied to obtain a better right hand side function in the coarse-level correction. The extrapolation is done to produce a better approximation of the fine-level solution at the next coarse level problem. In the jumping multigrid method, the extrapolation is used to produce an high order solution at a grid several levels higher.

Finally, we need to comment that the jumping multigrid method is similar to the cascadic multigrid method, cf. references in [5]. However, in the cascadic multigrid method, the fine-level smoothing destroys the low-frequency components of the solution while reducing its error in H^1-norm. This allows the cascadic multigrid method to proceed only a level or two ($i+2$ level) from the level i where exact solution is obtained. The jumping multigrid method would proceed roughly from level i to level $2i$.

The rest of the paper is organized as follows. In Section 2, we provide the finite element setting and the jumping multigrid methods. In Section 3, we present some finite element extrapolation (not the traditional extrapolation for the exact solution) formulae. In Section 4, we present four numerical test problems.

2 The jumping multigrid method

We solve a model Poisson equation:

$$\begin{aligned} -\Delta u &= f \qquad \text{in } \Omega \subset R^2, \\ u &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{2.1}$$

The bounded polygonal domain Ω is initially subdivided by $\mathcal{T}_0$, where the $\mathcal{T}_0$ is either a quadrilateral or a triangular grid. By connecting the mid-edge points, we have a sequence of nested grids:

$$\mathcal{T}_0 \subset \mathcal{T}_1 \subset \cdots,$$

where the grid sizes are $h_j = h_{j-1}/2$. The multi-level finite element spaces are

$$V_j = \{v \in C_0(\Omega) \mid v|_K \in Q_1 \text{ or } P_1 \text{ for all } K \in \mathcal{T}_j\}, \tag{2.2}$$

where Q_1 is the bilinear polynomial space on quadrilateral grids, while P_1 is the space of linear polynomials on triangular grids. The finite element discretization of (2.1) is: Find $u_j \in V_j$ such that

$$(\nabla u_j, \nabla v_h) = (f, v_h) \qquad \forall v_h \in V_j. \tag{2.3}$$

Usually, we only want the finite element solution u_h on the highest level V_j. A standard multigrid V-cycle is defined as follows.

Algorithm 2.1 *(One j-th level V-cycle iteration.) Given an initial guess $u_j^{(0)}$ of the finite element problem with the right hand side function g, we generate $V(u_j^{(0)}, g, j) = u_j^{(2m+1)}$ by one j-th level V-cycle iteration.*

1. *(Pre-smoothing) We do m pre-smoothing iterations*

$$u_j^{(i)} = S(u_j^{(i-1)}, g), \quad i = 1, 2, ..., m$$

where the smoothing is usually the conjugate gradient iteration and m is 3 or 4.

2. *(Coarse-level correction) We do next one V-cycle correction: If $j > 0$,*

$$u_j^{(m+1)} = V(0, g - A_j u_j^{(m)}, j-1),$$

where $(A_j u_j^{(m)}, v_h) = (\nabla u_j^{(m)}, \nabla v_h)$ for all $v_h \in V_j$. And if $j = 0$,

$$u_j^{(m+1)} = u_j^{(m)}.$$

3. *(Post-smoothing) We then do m post-smoothing iterations*

$$u_j^{(i)} = S(u_j^{(i-1)}, g), \quad i = m+2, m+3, ..., 2m+1.$$

To solve the finite element equation (2.3), we do enough V-cycle multigrid iterations as follows.

Algorithm 2.2 *(The multigrid iteration.) Let $u_{j,l} = 0$ or $u_{j,l} = u_{j-1}$ of the lower level solution. We compute $u_{j,l}$ until convergence for the problem (2.3) by*

$$u_{j,l} = V(u_{j,l-1}, f, j), \quad l = 1, 2, \ldots.$$

The jumping multigrid method differs from the multigrid method only at higher levels.

Algorithm 2.3 *(The jumping multigrid method) Find two consecutive level multigrid solutions u_{i-1} and u_i for the problem (2.3). To find the jumping multigrid solution of (2.3) on level j (usually $j \le i + 5$ and $j < 2i$), we first interpolate u_{i-1} and u_i to get $\tilde{u}_j^{(b)} \in V_j$ on the boundary of each $K \in \mathcal{T}_{i-2}$ element by*

- *either the formula (3.13) below, $p = 1$ and $m = j - i + 1$,*
- *or the formula (3.7) below with P_{2p+2} interpolation, $p = 1$ and $m = j - i + 1$.*

Then we apply the multigrid method on each element to find $u_j|_K$ inside each element:

$$(\nabla \tilde{u}_j^{(i)}, v_h) = (f, v_h) - (\nabla \tilde{u}_j^{(b)}, v_h) \quad \forall v_h \in C_0(K) \cap V_j, \tag{2.4}$$

and let $\tilde{u}_j = \tilde{u}_j^{(i)} + \tilde{u}_j^{(b)}$, where $\tilde{u}_j^{(b)}$ is the zero-extension of the boundary value $\tilde{u}_j^{(b)}$ into an element K.

We note that the jumping multigrid method does not attempt to produce the exact solution of the original finite element equation (2.3). However, by the finite element extrapolation, we have

$$|\tilde{u}_j^{(b)}(\mathbf{x}_k) - u_j(\mathbf{x}_k)| \le Ch_i^4 \quad \text{for all } \mathbf{x}_k \in \partial K,\ K \in \mathcal{T}_{i-2}. \tag{2.5}$$

Subtracting (2.4) from (2.3), we have

$$(\nabla(u_j - \tilde{u}_j), \nabla v_h) = 0 \quad \forall v_h \in C_0(K) \cap V_j. \tag{2.6}$$

Thus, by the maximum principle for the linear finite element (cf. [8, 6, 12, 10]), we have the following optimal order convergence for the jumping multigrid method.

Theorem 2.1 *At the convergence, the jumping multigrid method defined in Algorithm 2.3 produces a solution of optimal order*

$$|u - \tilde{u}_j|_{l^\infty} \le Ch_j^2,$$

assuming the linear triangle or the bilinear square element on uniform grids are used ($p = 1$), $i > j/2$, and the solution $u \in H^4(\Omega)$ in (2.1).

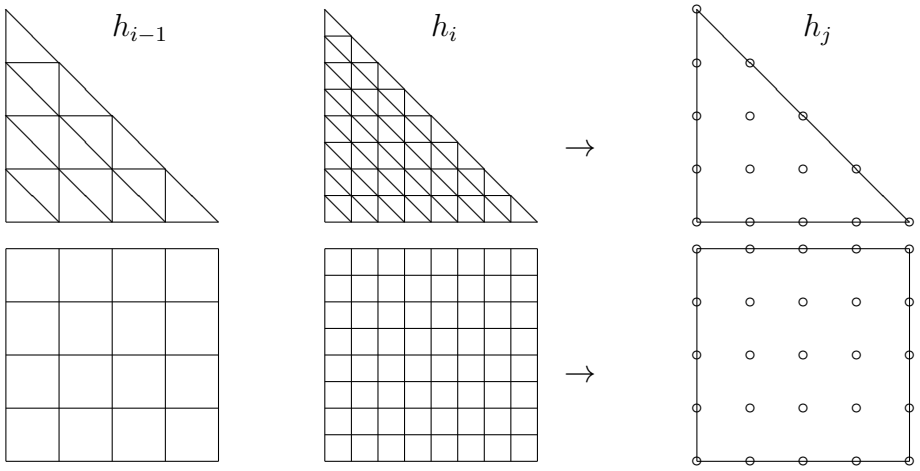


Figure 2: The P_4 and Q_4 extrapolation for linear and bilinear elements, respectively.

Proof. By the discrete maximal principle for P_1 and Q_1 elements, cf. [8, 6], and (2.6), it follows that

$$|\tilde{u}_j^{(b)} - u_j|_{l^\infty(K)} \leq |\tilde{u}_j^{(b)} - u_j|_{l^\infty(\partial K)} \leq Ch_i^4 \leq Ch_j^2 \quad \forall K \in \mathcal{T}_{i-3}.$$

∎

There might be a superconvergence between the jumping multigrid solution and multigrid solution. But both have only $O(h_j^2)$ nodal convergence to the exact solution u. We extrapolate the fine level solutions on some inter-element boundary in the jumping multigrid method. We can also do such an extrapolation on the whole domain as well.

Algorithm 2.4 *(A variation of jumping multigrid method) Find two consecutive level multigrid solutions u_{i-1} and u_i for the problem* (2.3)*. To find the jumping multigrid solution of* (2.3) *on level j (usually $j \leq i+5$ and $j < 2i$), we interpolate u_{i-1} and u_i to get $\tilde{u}_j^{(b)} \in V_j$ at all finite element nodes by the formula* (3.7) *below (to get the nodal values at P_{2p+2}/Q_{2p+2} nodes on grid u_{i-3}) and followed by the P_{2p+2}/Q_{2p+2} interpolation (to get all nodal values at the h_j grid), cf. Figure 2.*

The variation of jumping multigrid method also provides an optimal order solution on level h_j. But if the exact solution is not smooth, for certain cases, the extrapolation on inter-element boundary may work well but not work inside an element where the singularity occurs. For such a case, the jumping multigrid method may work while its variation does not.

3 Extrapolation for finite element solution

We provide the extrapolation formulae for $p = 1$ and $p = 2$. This includes and slightly extends the work in [5, 13, 4].

3.1 Extrapolation at a common node of all levels

Let u_h be a finite element solution of polynomial degree p in d-dimensional space. Assuming the grid is symmetric locally and the solution is smooth enough, cf. (1.1) and comments there, we have the following asymptotic expansions at a vertex x_j:

$$(u_h - u)(x_j) = c_1(x_j)h^{2p} + c_2(x_j)h^{2p+2} + o(h^{2p+2}), \tag{3.1}$$

$$(u_{h/2} - u)(x_j) = c_1(x_j)\frac{h^{2p}}{2^{2p}} + c_2(x_j)\frac{h^{2p+2}}{2^{2p+2}} + o(h^{2p+2}). \tag{3.2}$$

Here $p = 1$ and $p = 2$. For $p = 2$, we have a superconvergence at a vertex x_j. The classical extrapolation produces an approximation to the exact solution, i.e., eliminating the $O(h^{2p})$ terms in (3.1) and (3.2).

$$\left(2^{2p}u_{h/2} - u_h - (2^{2p} - 1)u\right)(x_j) = c_2(x_j)h^{2p+2}(-\frac{3}{4}) + o(h^{2p+2}),$$

$$u(x_j) = \frac{2^{2p}u_{h/2}(x_j) - u_h(x_j)}{2^{2p} - 1} + c_2(x_j)\frac{3h^{2p+2}}{2^{2p+2} - 4} + o(h^{2p+2}). \tag{3.3}$$

The finite element extrapolation is to produce an high-order approximation to the next solution only, not to the exact solution. By (3.1), we also have

$$(u_{h/4} - u)(x_j) = c_1(x_j)\frac{h^{2p}}{4^{2p}} + c_2(x_j)\frac{h^{2p+2}}{4^{2p+2}} + o(h^{2p+2}). \tag{3.4}$$

We would eliminate both the low-order terms h^{2p} and the exact solution $u(x_j)$ this time, by three equations, (3.1), (3.2) and (3.4). Repeating the elimination for (3.3), we have

$$u(x_j) = \frac{2^{2p}u_{h/4}(x_j) - u_{h/2}(x_j)}{2^{2p} - 1} + c_2(x_j)\frac{3h^{2p+2}}{2^{2p+2} - 4} + o(h^{2p+2}). \tag{3.5}$$

Eliminating $u(x_j)$ by (3.3) and (3.5), we would get an extrapolation formula for the finite element solution

$$\frac{2^{2p}u_{h/2}(x_j) - u_h(x_j)}{2^{2p} - 1} = \frac{2^{2p}u_{h/4}(x_j) - u_{h/2}(x_j)}{2^{2p} - 1} + O(h^{2p+2}).$$

That is,

$$u_{h/4}(x_j) = \frac{2^{2p} + 1}{2^{2p}}u_{h/2}(x_j) - \frac{1}{2^{2p}}u_h(x_j) + O(h^{2p+2}). \tag{3.6}$$

For higher level solutions, (3.5) would be

$$u(x_j) = \frac{2^{(2m-2)p}u_{h/2^m}(x_j) - u_{h/2}(x_j)}{2^{(2m-2)p} - 1} + O(h^{2p+2}).$$

Then, (3.6) is extended to

$$u_{h/2^m}(x_j) = \frac{(2^{2mp} - 1)u_{h/2}(x_j) - (2^{(2m-2)p} - 1)u_h(x_j)}{2^{2mp} - 2^{(2m-2)p}} + O(h^{2p+2}). \tag{3.7}$$

For example, when $p = 1$, we have the following extrapolation formulae,

$$\begin{aligned}
u_{h/4}(x_j) &= \frac{5u_{h/2}(x_j) - u_h(x_j)}{4}, \\
u_{h/8}(x_j) &= \frac{21u_{h/2}(x_j) - 5u_h(x_j)}{16}, \\
u_{h/16}(x_j) &= \frac{85u_{h/2}(x_j) - 21u_h(x_j)}{64}, \\
u_{h/32}(x_j) &= \frac{341u_{h/2}(x_j) - 85u_h(x_j)}{256}, \\
u_{h/2^m}(x_j) &= \frac{(2^{2m} - 1)u_{h/2}(x_j) - (2^{(2m-2)} - 1)u_h(x_j)}{2^{(2m-2)}(3)}, \\
u(x_j) &= \frac{4u_{h/2}(x_j) - u_h(x_j)}{3}, \qquad \text{when } m \to \infty.
\end{aligned}$$

If $p = 2$, the extrapolation formulae are

$$\begin{aligned}
u_{h/4}(x_j) &= \frac{17u_{h/2}(x_j) - u_h(x_j)}{16}, \\
u_{h/8}(x_j) &= \frac{273u_{h/2}(x_j) - 17u_h(x_j)}{256}, \\
u_{h/16}(x_j) &= \frac{4469u_{h/2}(x_j) - 273u_h(x_j)}{4096}, \\
u(x_j) &= \frac{16u_{h/2}(x_j) - u_h(x_j)}{15}, \qquad \text{when } m \to \infty.
\end{aligned}$$

3.2 Extrapolation at a mid-node of all levels

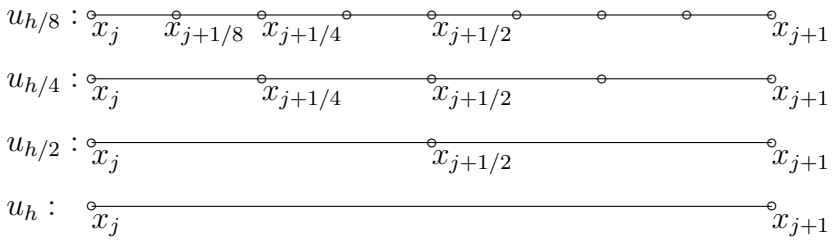


Figure 3: Four levels of nested grids.

Except at the first level, i.e. u_h level, the mid-point $x_{j+1/2}$ is a finite element node, cf. Figure 3. Thus (3.2) and (3.4) hold. On the first level, we use such an asymptotic

expansion to avoid the unknown mid-value $c_1(x_{j+k/2^n})$:

$$\begin{aligned}
&\frac{(2^n-k)c_1(x_j)+kc_1(x_{j+1})}{2^n}+O(h^2)\\
=&\frac{(2^n-k)[c_1(x_{j+\frac{k}{2^n}})-\frac{k}{2^n}hc_1'(x_{j+k/2^n})]+k(c_1(x_{j+\frac{k}{2^n}})+(1-\frac{k}{2^n})hc_1'(x_{j+\frac{k}{2^n}})]}{2^n}\\
=&c_1(x_{j+\frac{k}{2^n}}),\qquad k=0,1,2,\ldots,2^n. \qquad (3.8)
\end{aligned}$$

In particular,

$$c_1(x_{j+1/2})=\frac{c_1(x_j)+c_1(x_{j+1})}{2}+O(h^2).$$

This time, to get the extrapolated value $u_{h/4}(x_{j+1/2})$, we have only one finite element solution at the point $u_{h/2}(x_{j+1/2})$, cf. Figure 3. We use the four vertex values below it, $u_h(x_j)$, $u_{h/2}(x_j)$, $u_h(x_{j+1})$ and $u_{h/2}(x_{j+1})$. Subtracting (3.1) from (3.2), it follows that

$$(u_{h/2}-u_h)(x_{j+i})=\frac{1-2^{2p}}{2^{2p}}c_1(x_{j+i})h^{2p}+O(h^{2p+2}),\qquad i=0,1. \qquad (3.9)$$

In same fashion, we have

$$(u_{h/2^m}-u_{h/2})(x_{j+i})=\frac{1-2^{(2m-2)p}}{2^{2mp}}c_1(x_{j+i})h^{2p}+O(h^{2p+2}),\qquad i=0,\frac{1}{2},1. \qquad (3.10)$$

At the middle point $x_{j+1/2}$, we get, by (3.8), (3.9) and (3.10),

$$\begin{aligned}
&u_{h/2^m}(x_{j+\frac{1}{2}})\\
=&u_{h/2}(x_{j+\frac{1}{2}})++\frac{1-2^{(2m-2)p}}{2^{2mp}}\frac{c_1(x_j)+c_1(x_{j+1})}{2}h^{2p}+O(h^{2p+2})\\
=&u_{h/2}(x_{j+\frac{1}{2}})+\frac{1-2^{(2m-2)p}}{2^{(2m-2)p}-2^{2mp}}\frac{(u_{h/2}-u_h)(x_j)+(u_{h/2}-u_h)(x_{j+1})}{2}+O(h^{2p+2}). \qquad (3.11)
\end{aligned}$$

This provides a formula for each higher level mid-value $u_{h/2^m}(x_{j+1/2})$. In particular, for $p=1$,

$$\begin{aligned}
u_{\frac{h}{4}}(x_{j+\frac{1}{2}})=&u_{\frac{h}{2}}(x_{j+\frac{1}{2}})+\frac{u_{\frac{h}{2}}(x_j)+u_{\frac{h}{2}}(x_{j+1})}{8}-\frac{u_h(x_j)+u_h(x_{j+1})}{8},\\
u_{\frac{h}{8}}(x_{j+\frac{1}{2}})=&u_{\frac{h}{2}}(x_{j+\frac{1}{2}})+5\frac{u_{\frac{h}{2}}(x_j)+u_{\frac{h}{2}}(x_{j+1})}{32}-5\frac{u_h(x_j)+u_h(x_{j+1})}{32},\\
u_{\frac{h}{8}}(x_{j+\frac{1}{2}})=&u_{\frac{h}{2}}(x_{j+\frac{1}{2}})+21\frac{u_{\frac{h}{2}}(x_j)+u_{\frac{h}{2}}(x_{j+1})}{128}-21\frac{u_h(x_j)+u_h(x_{j+1})}{128},\\
u(x_{j+\frac{1}{2}})=&u_{\frac{h}{2}}(x_{j+\frac{1}{2}})+\frac{u_{\frac{h}{2}}(x_j)+u_{\frac{h}{2}}(x_{j+1})}{6}-\frac{u_h(x_j)+u_h(x_{j+1})}{6},\\
&\text{when}\ \ m\to\infty.
\end{aligned}$$

For $p = 2$,

$$\begin{aligned}
u_{\frac{h}{4}}(x_{j+\frac{1}{2}}) =& u_{\frac{h}{2}}(x_{j+\frac{1}{2}}) + \frac{u_{\frac{h}{2}}(x_j) + u_{\frac{h}{2}}(x_{j+1})}{32} - \frac{u_h(x_j) + u_h(x_{j+1})}{32},\\
u_{\frac{h}{8}}(x_{j+\frac{1}{2}}) =& u_{\frac{h}{2}}(x_{j+\frac{1}{2}}) + 17\frac{u_{\frac{h}{2}}(x_j) + u_{\frac{h}{2}}(x_{j+1})}{512} - 17\frac{u_h(x_j) + u_h(x_{j+1})}{512},\\
u_{\frac{h}{16}}(x_{j+\frac{1}{2}}) =& u_{\frac{h}{2}}(x_{j+\frac{1}{2}}) + 273\frac{u_{\frac{h}{2}}(x_j) + u_{\frac{h}{2}}(x_{j+1})}{8192} - 273\frac{u_h(x_j) + u_h(x_{j+1})}{8192},\\
u(x_{j+\frac{1}{2}}) =& u_{\frac{h}{2}}(x_{j+\frac{1}{2}}) + \frac{u_{\frac{h}{2}}(x_j) + u_{\frac{h}{2}}(x_{j+1})}{30} - \frac{u_h(x_j) + u_h(x_{j+1})}{30},\\
& \text{when } \ m \to \infty.
\end{aligned}$$

3.3 Extrapolation at a new node of any level

We next study the extrapolation for $u_{h/4}(x_{j+1/4})$. At such a point, we have no coarse level-function value, different from the cases of $u_{h/4}(x_j)$ and $u_{h/4}(x_{j+1/2})$, where we have at least one coarse-level value. Then we cannot extrapolate the two levels of solution to get high enough order solution. Instead, we extrapolate solutions on three levels. This gives our final extrapolation formula.

Assuming $u \in C^6([x_j, x_{j+1}])$, we find the linear combination first:

$$\begin{aligned}
& au(x_j) + bu(x_{j+\frac{1}{4}}) + cu(x_{j+\frac{1}{2}}) + du(x_{j+\frac{3}{4}}) + eu(x_{j+1}) + O(h^6)\\
=& u(x_{j+\frac{k}{2^n}}) + \Big[\frac{3k}{2^n} - \frac{25k^2}{2^{2n}} + \frac{70k^3}{2^{3n}} - \frac{80k^4}{2^{4n}} + \frac{32k^5}{2^{5n}}\Big]u^{(5)}(x_{j+\frac{k}{2^n}})\frac{h^5}{2^5 \cdot 5!},
\end{aligned}$$

where

$$\begin{cases}
a =1 + \dfrac{1}{3}\Big[- \dfrac{25k}{2^n} + \dfrac{70k^2}{2^{2n}} - \dfrac{80k^3}{2^{3n}} + \dfrac{32k^4}{2^{4n}}\Big],\\
b = - \dfrac{16}{3}\Big[- \dfrac{3k}{2^n} + \dfrac{13k^2}{2^{2n}} - \dfrac{18k^3}{2^{3n}} + \dfrac{8k^4}{2^{4n}}\Big],\\
c = \quad 4\Big[- \dfrac{3k}{2^n} + \dfrac{19k^2}{2^{2n}} - \dfrac{32k^3}{2^{3n}} + \dfrac{16k^4}{2^{4n}}\Big],\\
d = - \dfrac{16}{3}\Big[- \dfrac{k}{2^n} + \dfrac{7k^2}{2^{2n}} - \dfrac{14k^3}{2^{3n}} + \dfrac{8k^4}{2^{4n}},\\
e = \quad \dfrac{1}{3}\Big[- \dfrac{3k}{2^n} + \dfrac{22k^2}{2^{2n}} - \dfrac{48k^3}{2^{3n}} + \dfrac{32k^4}{2^{4n}}\Big].
\end{cases} \tag{3.12}$$

For example, when $k = 1$, $n = 3$, we have

$$\begin{aligned}
& \frac{35}{128}u(x_j) + \frac{35}{32}u(x_{j+\frac{1}{4}}) - \frac{35}{64}u(x_{j+\frac{1}{2}}) + \frac{7}{32}u(x_{j+\frac{3}{4}}) - \frac{5}{128}u(x_{j+1}) + O(h^6)\\
=& u(x_{j+\frac{1}{8}}) - \frac{7}{8388608}u^{(5)}(x_{j+\frac{1}{8}})h^5.
\end{aligned}$$

By the same weights, we expand the coefficient $c_1(x_{j+k/2^n})$:

$$\begin{aligned} u_{h/2^m}(x_{j+k/2^n}) &= u(x_{j+k/2^n}) + c_1(x_{j+k/2^n})\frac{h^{2p}}{2^{2mp}} + O(h^{\min\{2p+2,5\}}) \\ &= [au(x_j) + bu(x_{j+\frac{1}{4}}) + cu(x_{j+\frac{1}{2}}) + du(x_{j+\frac{3}{4}}) + eu(x_{j+1})] \\ &+ \Big[ac_1(x_j) + bc_1(x_{j+\frac{1}{4}}) + cc_1(x_{j+\frac{1}{2}}) + dc_1(x_{j+\frac{3}{4}}) + ec_1(x_{j+1})\Big]\frac{h^{2p}}{2^{2mp}} \end{aligned}$$

At these 5 points, $\{x_{j+l/4}, l = 0, 1, 2, 3, 4\}$, we have high-order extrapolations in the last two subsections.

$$\begin{aligned} u_{h/2^m}(x_{j+l/4}) &= u(x_{j+l/4}) + O(h^{2p+2}) \\ &= \frac{(2^{(2m-2)p} - 1)u_{h/4}(x_{j+l/4}) - (2^{(2m-4)p} - 1)u_{h/2}(x_{j+l/4})}{2^{(2m-2)p} - 2^{(2m-4)p}}, \quad l = 0, 2, 4, \\ u_{h/2^m}(x_{j+l/4}) &= u(x_{j+l/4}) + O(h^{2p+2}) \\ &= u_{h/4}(x_{j+l/4}) + \frac{1 - 2^{(2m-4)p}}{2^{(2m-4)p} - 2^{(2m-2)p}} \cdot \Big[\frac{3-l}{4}(u_{h/4} - u_{h/2})(x_j) \\ &+ \frac{1}{2}(u_{h/4} - u_{h/2})(x_{j+1/2}) + \frac{l-1}{4}(u_{h/4} - u_{h/2})(x_{j+1})\Big], \quad l = 1, 3. \end{aligned}$$

Hence, we have take the following expansion formula

$$\begin{aligned} u_{\frac{h}{2^m}}(x_{j+\frac{k}{2^n}}) &= au_{\frac{h}{2^m}} + bu_{\frac{h}{2^m}}(x_{j+\frac{1}{4}}) + cu_{\frac{h}{2^m}}(x_{j+\frac{1}{2}}) \\ &+ du_{\frac{h}{2^m}}(x_{j+\frac{3}{4}}) + eu_{\frac{h}{2^m}}(x_{j+1}) + O(h^{\min\{2p+2,5\}}). \end{aligned}$$

Therefore the extrapolation formula for the finite element solution, at any level m and at any node $j + k/2^n$, cf. Figure 3, is

$$\begin{aligned} u^{\text{extrap}}_{\frac{h}{2^m}}(x_{j+\frac{k}{2^n}}) &= a_1u_{\frac{h}{4}}(x_j) + a_2u_{\frac{h}{2}}(x_j) + b_1u_{\frac{h}{4}}(x_{j+\frac{1}{4}}) + c_1u_{\frac{h}{4}}(x_{j+\frac{1}{2}}) + c_2u_{\frac{h}{2}}(x_{j+\frac{1}{2}}) \\ &+ d_1u_{\frac{h}{4}}(x_{j+\frac{3}{4}}) + e_1u_{\frac{h}{4}}(x_{j+1}) + e_2u_{\frac{h}{2}}(x_{j+1}), \end{aligned} \tag{3.13}$$

where

$$\begin{cases} a_1 = a\alpha + b\beta/2, \\ a_2 = a\beta - b\beta/2, \\ b_1 = b, \\ c_1 = c\alpha + b\beta/2 + d\beta/2, \\ c_2 = c\beta - b\beta/2 - d\beta/2, \\ d_1 = d, \\ e_1 = e\alpha + d\beta/2, \\ e_2 = e\beta - d\beta/2. \end{cases} \tag{3.14}$$

Here in (3.14), a to e are defined in (3.12) and

$$\alpha = \frac{2^{(2m-2)p} - 1}{2^{(2m-2)p} - 2^{(2m-4)p}}, \qquad \beta = \frac{1 - 2^{(2m-4)p}}{2^{(2m-2)p} - 2^{(2m-4)p}}.$$

We note that, the final formula (3.13) includes (3.7) and (3.11).

4 Numerical tests

We provide the numerical tests for four problems. For smoothing problems, the jumping multigrid method works well. But we also present many problems with singularity and some further improvements.

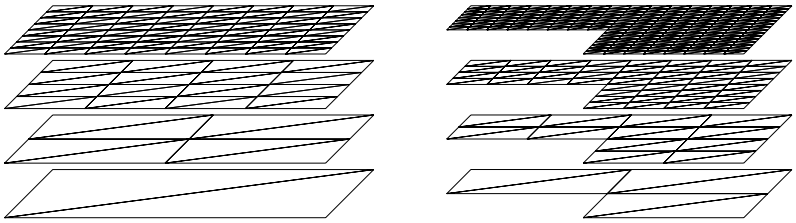


Figure 4: The P_1 grids for (4.1), (4.2) and (4.3), and then for (4.4).

4.1 A smooth problem

We compute the Poisson equation (2.1) on the unit square $[0,1]^2$ by the bilinear finite element, and by the linear triangular finite element. The domain is bisectioned again and again to generate a nested sequence of grids, of size $1/2^j$, shown in Figure 1. The exact solution is

$$u(x,y) = 2^4 x(x-x^2)e^y(y-y^2) \quad (x,y) \in [0,1]^2. \tag{4.1}$$

The convergence and cpu time are provided in Table 1. We can see the jumping multigrid method works as well as the multigrid method, but with less computation, and much less memory requirement. Again, we note that, by Algorithm 2.3, the $h = 2^{-12}$ solution is obtained directly from $h = 2^{-6}$ and $h = 2^{-7}$ solutions. But we repeatedly solve the lower-level problems on $h = 2^{-1}$ to $h = 2^{-10}$ to obtain the solution on $h = 2^{-11}$ level, in the traditional multigrid method.

We next solve the problem (4.1) again by the P_1 triangle element, on the uniform grids shown in Figure 4. We note that a superconvergence occurs in the H^1-norm. The jumping multigrid method works well for P_1 elements, shown in Table 2. However, it is slightly better for Q_1 elements on uniform square grids. It is understandable that the square grids are more symmetric than triangular grids. By Table 2, the Algorithm 2.4 jumping multigrid method works as well, but slightly worse than the method of Algorithm 2.3.

4.2 An $H^{2-\epsilon}$ solution example

We compute the Poisson equation (2.1) on the unit square $[0,1]^2$ by the P_1 finite element on grids shown in Figure 4. The exact solution is

$$u(x,y) = \frac{2xy}{\sqrt{x^2+y^2}}, \quad (x,y) \in [0,1]^2. \tag{4.2}$$

Table 2: The P_1 element for problem (4.1).

$-\log_2 h$	$\|e_h\|_{l^\infty}$	h^n	$\|e_h\|_{L^2}$	h^n	$\|e_h\|_{H^1}$	h^n
	The multigrid method					
7	0.000459210	2.0	0.000228710	2.0	0.001196	2.0
8	0.000114837	2.0	0.000057222	2.0	0.000299	2.0
9	0.000028711	2.0	0.000014308	2.0	0.000074	2.0
10	0.000007178	2.0	0.000003577	2.0	0.000018	2.0
11	0.000001795	2.0	0.000000894	2.0	0.000004	2.0
12	0.000000449	2.0	0.000000224	2.0	0.000001	2.0
	The jumping multigrid method, based on grids 7 and 8.					
9	0.000028743	2.0	0.000014325	2.0	0.000075	2.0
10	0.000007220	2.0	0.000003600	2.0	0.000019	1.9
11	0.000001839	2.0	0.000000918	2.0	0.000007	1.4
12	0.000000562	1.7	0.000000249	1.9	0.000005	0.3
	The Algorithm 2.4 jumping multigrid method.					
9	0.000028749	2.0	0.000014325	2.0	0.000075	2.0
10	0.000007239	2.0	0.000003600	2.0	0.000020	1.9
11	0.000001862	2.0	0.000000918	2.0	0.000008	1.2
12	0.000000562	1.7	0.000000249	1.9	0.000007	0.2
13	0.000000518	0.1	0.000000086	1.5	0.000007	0.0

* $e_h = I_h u - u_h$.

The singularity occurs at one point $(0, 0)$. The exact solution and the numerical error are shown in Figure 5.

We list in Table 3 the error and the order of convergence for various norms for the P_1 element in solving (4.2). Different from Table 2, the H^1-norm convergence is of order 1, not of order 2 any more. The superconvergence disappears for problems with a singularity. As the solution is not smooth, the jumping multigrid method works only for one or two levels only. The Algorithm 2.4 fails to work completely. So, solving local problem after extrapolation would make a difference when a singularity exists inside a local problem. Further research is to be done in dealing the singular problems in the jumping multigrid method.

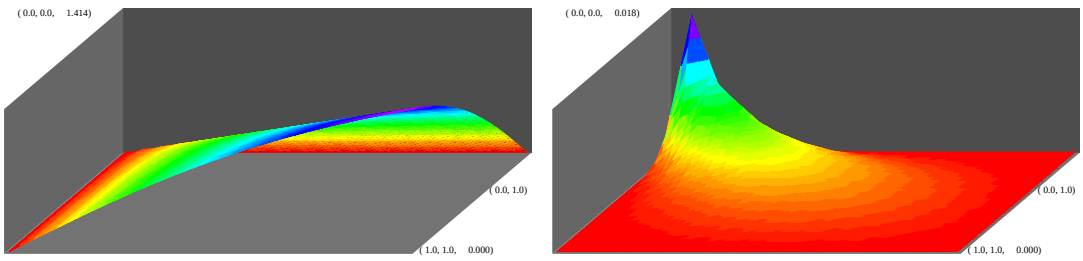


Figure 5: The solution and error for (4.2).

Table 3: The P_1 element for problem (4.2).

$-\log_2 h$	$\lvert e_h\rvert_{l^\infty}$	h^n	$\lVert e_h\rVert_{L^2}$	h^n	$\lvert e_h\rvert_{H^1}$	h^n
	The multigrid method					
7	0.004489566	1.0	0.000205803	1.8	0.008917281	1.0
8	0.002244801	1.0	0.000057233	1.8	0.004459681	1.0
9	0.001122402	1.0	0.000015630	1.9	0.002229971	1.0
10	0.000561201	1.0	0.000004213	1.9	0.001115002	1.0
11	0.000280600	1.0	0.000001125	1.9	0.000557503	1.0
12	0.000140300	1.0	0.000000298	1.9	0.000278752	1.0
	The jumping multigrid method, based on grids 7 and 8.					
9	0.001122510	1.0	0.000017957	1.7	0.003202675	0.5
10	0.000777685	0.5	0.000009188	1.0	0.002709725	0.2
11	0.000782129	0.0	0.000007875	0.2	0.002572829	0.1
12	0.000785371	0.0	0.000007689	0.0	0.002537507	0.0
	The Algorithm 2.4 jumping multigrid method.					
9	0.003007622	-.4	0.000042717	0.4	0.010821789	***
10	0.003083738	0.0	0.000042536	0.0	0.011534462	-.1

* $e_h = I_h u - u_h$.

4.3 An $H^{1+\epsilon}$ solution example

We compute the Poisson equation (2.1) on the unit square $[0,1]^2$ by the P_1 finite element on grids shown in Figure 4. The exact solution is

$$u(x,y) = \frac{2xy}{(x^2+y^2)^{3/4}}, \quad (x,y) \in [0,1]^2. \tag{4.3}$$

The singularity occurs at one point $(0,0)$. The exact solution and the numerical error are shown in Figure 6.

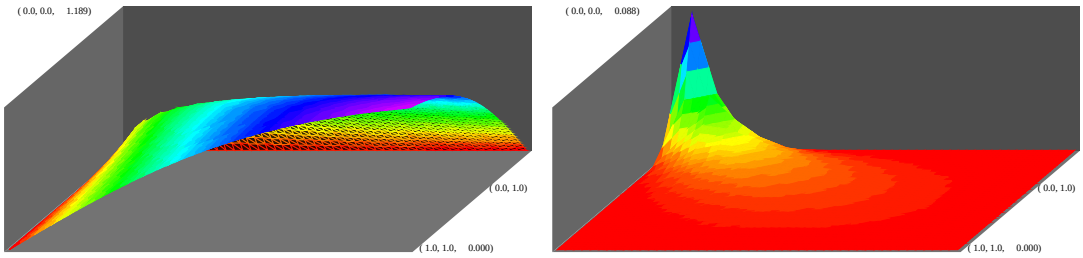


Figure 6: The solution and error for (4.3).

We list the errors and the orders of convergence for the finite element solution, in various methods, in Table 3. This time, we lose another half order of convergence, comparing to the data in Table 3. Again, the Algorithm 2.4 fails to work completely. But the deterioration of the jumping multigrid method is less significant, compared to

that for solving (4.2). This is surprising as we have a worse solution, but a better jumping multigrid method.

Table 4: The P_1 element for problem (4.3).

$-\log_2 h$	$\lvert e_h\rvert_{l^\infty}$	h^n	$\lVert e_h\rVert_{L^2}$	h^n	$\lvert e_h\rvert_{H^1}$	h^n
	The multigrid method					
6	0.062312858	0.5	0.002952996	1.4	0.116346515	0.5
7	0.044062534	0.5	0.001072977	1.5	0.082274383	0.5
8	0.031156961	0.5	0.000384708	1.5	0.058177240	0.5
9	0.022031301	0.5	0.000136994	1.5	0.041137564	0.5
10	0.015578483	0.5	0.000048613	1.5	0.029088654	0.5
11	0.011015651	0.5	0.000017219	1.5	0.020568785	0.5
12	0.007789241	0.5	0.000006094	1.5	0.014544327	0.5
	The jumping multigrid method, based on grids 7 and 8.					
9	0.022034466	0.5	0.000154339	1.3	0.044748074	0.4
10	0.015580105	0.5	0.000082330	0.9	0.034710939	0.4
11	0.011016125	0.5	0.000065678	0.3	0.028180215	0.3
12	0.007789365	0.5	0.000062714	0.1	0.024200770	0.2
	The Algorithm 2.4 jumping multigrid method.					
9	0.022317164	0.4	0.000214975	0.2	0.063180577	-.3
10	0.024392527	-.1	0.000229887	-.1	0.078892644	-.3

* $e_h = I_h u - u_h$.

4.4 An L-shape domain example

In the last numerical test, the jumping multigrid method may fail completely, if it is not amended. We solve (2.1) on an L-shape domain by the P_1 finite element on grids shown in Figure 4. The exact solution is

$$u(x,y) = r^{2/3}\sin\frac{2\theta}{3}, \quad (x,y) \in [-1,1]^2 \setminus ([0,1]\times[-1,0]), \tag{4.4}$$

where (r,θ) are the polar coordinates for point (x,y).

The zero boundary condition in (2.1) is replaced by a Dirichlet boundary condition in this case. The singularity occurs at one point $(0,0)$. The grids are shown in Figure 4, and Figure 8. The exact solution and the numerical error are shown in Figure 7.

The errors are shown in Table 5. The finite element solution loses 1/2 order of convergence. Usually, we would use graded grids for such a singular problem. This time, the jumping multigrid method fails completely to provide a solution on any higher level. What is the problem? Can we correct the jumping multigrid method? Yes.

The failure of the jumping multigrid method, reported in Table 5, is caused by the finite element extrapolation based on inaccurate coarse-level solutions on the three singular edges, see Figure 8. The coarse-level solutions are very inaccurate at the three

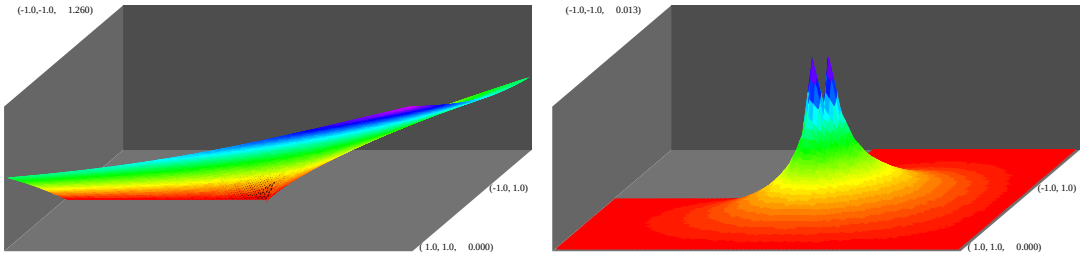


Figure 7: The solution and error for (4.4).

Table 5: The P_1 element for L-shape domain problem (4.4).

$-\log_2 h$	$\lvert e_h\rvert_{l^\infty}$	h^n	$\lVert e_h\rVert_{L^2}$	h^n	$\lvert e_h\rvert_{H^1}$	h^n
	The multigrid method					
7	0.005249495	0.7	0.000442310	1.3	0.011100790	0.7
8	0.003312514	0.7	0.000178705	1.3	0.006998354	0.7
9	0.002088137	0.7	0.000071703	1.3	0.004410006	0.7
10	0.001315789	0.7	0.000028650	1.3	0.002778459	0.7
11	0.000828982	0.7	0.000011418	1.3	0.001750402	0.7
	The jumping multigrid method, based on grids 7 and 8.					
9	0.006868472	***	0.000119763	0.6	0.014845067	***
10	0.006942016	0.0	0.000105834	0.2	0.015904440	-.1

* $e_h = I_h u - u_h$.

edges, $\{x = 0, y > 0\}$, $\{x + y = 0, x < 0\}$ and $\{y = 0, x < 0\}$, near the point $(0, 0)$. To avoid using the solutions on those three edges in extrapolation, we use a big patch of triangles at the singular point $(0, 0)$, i.e., using the boundary values of the coarse-level solutions on the 4 outside edges only, see Figure 8. Then the jumping multigrid method works well. The order of convergence is shown in Table 6.

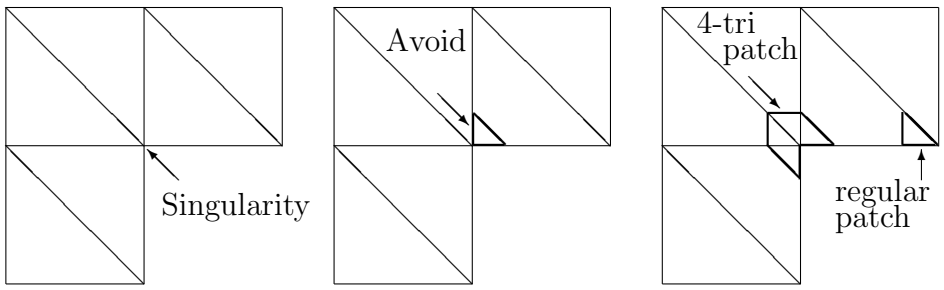


Figure 8: Using one 4-triangle patch, avoiding the singularity at $(0, 0)$.

Table 6: The P_1 element for (4.4), with Figure 8 extrapolation.

$-\log_2 h$	$\lvert e_h\rvert_{l^\infty}$	h^n	$\lVert e_h\rVert_{L^2}$	h^n	$\lvert e_h\rvert_{H^1}$	h^n
	The multigrid method					
7	0.005249495	0.7	0.000442310	1.3	0.011100790	0.7
8	0.003312514	0.7	0.000178705	1.3	0.006998354	0.7
9	0.002088137	0.7	0.000071703	1.3	0.004410006	0.7
10	0.001315789	0.7	0.000028650	1.3	0.002778459	0.7
11	0.000828982	0.7	0.000011418	1.3	0.001750402	0.7
	The jumping multigrid method, based on grids 7 and 8.					
9	0.002101275	0.7	0.000091341	1.0	0.004460345	0.6
10	0.001339321	0.6	0.000082607	0.1	0.002904556	0.6
11	0.000847952	0.7	0.000080855	0.0	0.001955927	0.6
12	0.000535260	0.7	0.000080422	0.0	0.001408952	0.5
13	0.000386822	0.5	0.000080289	0.0	0.001119360	0.3

* $e_h = I_h u - u_h$.

References

[1] R. Bank and T. Dupont, An optimal order process for solving finite element equations, Math. Comp. 36 (1981), 35–51.

[2] C.M. Chen and Y.Q. Huang, High Accuracy Theory in Finite Element Methods, (in Chinese), Hunan Science and Technology Press, 1995.

[3] C.M. Chen and Q. Lin, Extrapolation of finite element approximation in a rectangular domain, J. Comp. Math. 7 (1989), 227–233.

[4] C.M. Chen, H.L. Hu, Z.Q. Xie and C.L. Li, Analysis of extrapolation cascadic multigrid method(EXCMG). Science in China, Series A. Mathematics. 51 (2008), no. 8, 1349–1360.

[5] C.M. Chen, H.L. Hu, Z.Q. Xie and C.L. Li, L^2-error of extrapolation cascadic multigrid (EXCMG), Acta Math. Sci. Ser. B Engl. Ed. 29 (2009), no. 3, 539–551.

[6] I. Christie and C. Hall, The maximum principle for bilinear elements, Int. J. Num. Meth. Engin. 20 (1984), 549–553.

[7] P.G. Ciarlet, The Finite Element Method for Elliptic Problems, North-Holland, Amsterdam, 1978.

[8] P.G. Ciarlet and P.-A. Raviart, Maximum principle and uniform convergence for the finite element method, Comput. Methods Appl. Mech. Engrg. 2 (1973), 17-31.

[9] S.R. Fulton, On the accuracy of multigrid truncation error estimates, Elec. Trans. Num. Anal., 15 (2003), 29-37.

[10] A. Draganescu, T.F. Dupont and L.R. Scott, Failure of the discrete maximum principle for an elliptic finite element problem, Math. Comp. 74 (2005), 1–23.

[11] W. Hackbusch, Multi-Grid Methods and Applications, Springer-Verlag, 1985.

[12] W. Höhn and H.-D. Mittelmann, Some remarks on the discrete maximum-principle for finite elements of higher order, Computing 27 (1981), no. 2, 145154.

[13] H.L. Hu, C.M. Chen and Z.Q. Xie, Extrapolation cascadic multigrid method (EXCMG)—a new algorithm for solving large scale elliptic problems. (Chinese) Math. Numer. Sin. 31 (2009), no. 3, 261–274.

[14] M. Jung and U. Rüde, Implicit extrapolation methods for multilevel finite element computations, SIAM J. Sci. Comput. 17 (1996), no. 1, 156–179.

[15] L.R. Scott and S. Zhang, Higher-dimensional nonnested multigrid methods, Math. Comp. 58 (1992), 457–466.

[16] S. Schaffer, Higher order multigrid methods, Math. Comp. 43 (1984), no. 167, 89–115.

[17] S. Zhang, Optimal order non-nested multigrid methods for solving finite element equations I: On quasiuniform meshes, Math. Comp. 55 (1990), 23–36.

[18] S. Zhang, Optimal order non-nested multigrid methods for solving finite element equations II: On non-quasiuniform meshes, Math. Comp. 55 (1990), 439–450.

[19] S. Zhang, Optimal order non-nested multigrid methods for solving finite element equations III: On degenerate meshes, Math. Comp. 64 (1995), 23–49.

∗ College of Mathematics and Computer, Hunan Normal University, Changsha, 410081, China. cmchen@hunnu.edu.cn
† honglinghu@yahoo.cn
‡ zqxie@hunnu.edu.cn
§ Department of Mathematical Sciences, University of Delaware, Newark, DE 19716, U.S.A. szhang@udel.edu.