INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING, *SERIES B*
Volume 2, Number 4, Pages 281–297

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, $h_{i+1}, h_{i+2}, \ldots, h_{j-1}$. Loosely speaking, this jumping multigrid method solves a linear system of $O(N)$ equations by a memory of $O(\sqrt{N})$, and by a parallel computation of $O(\sqrt{N})$.

Key words. elliptic equation, finite element, extrapolation, uniform grid, superconvergence

1. Introduction

The multigrid method is an optimal order solver which solves a linear system of N unknowns in $O(N)$ arithmetic operations, cf. [1, 16, 18, 19, 20]. 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 to 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. [6, 14, 5]. Thus, via such finite element extrapolation, we propose a new multigrid method, named jumping multigrid method. In the method, after 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 some 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. [3, 4]. 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

Received by the editors October 25, 2011 and, in revised form, November 22, 2011.
2000 *Mathematics Subject Classification.* 65M60, 65N30.

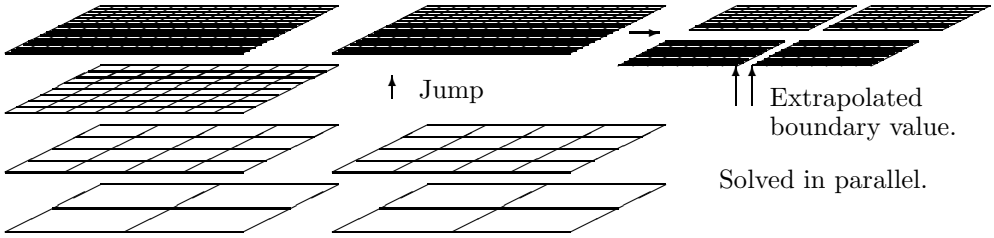

FIGURE 1. The multigrid method, and jumping multigrid.

each such a square, cf. Figure 1. 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 requirement 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.

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

		Multigrid			Jumping multigrid			
h	N	$\lvert u-u_h\rvert_{l^\infty}$	$\frac{e_i}{e_{i+1}}$	cpu	$\lvert u-u_h\rvert_{l^\infty}$	$\frac{e_i}{e_{i+1}}$	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.

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.

We should comment that the jumping multigrid method is similar to the cascadic multigrid method, cf. references in [6]. However, in the cascadic multigrid method, the fine-level smoothing destroys the low-frequency components of the solution