Performance of the Algebraic Multilevel Iteration Method on massively parallel computer architectures

Owe Axelsson and Maya Neytcheva
Faculty of Mathematics and Informatics
Catholic University, Toernooiveld, 6525 ED Nijmegen
axelsson@sci.kun.nl   neytchev@sci.kun.nl
Krassimir Georgiev
Center for Informatics and Computer Technology
Bulgarian Academy of Sciences
georgiev@iscbg.acad.bg

Plan of the talk:

❶ Introduction: Iterative solution methods of optimal order - the necessary choice when solving systems with large and sparse matrices

❷ The AMLI method

❸ The short AMLI method or how to balance the $fine/coarse$ level ratio for optimal computational complexity

❹ Numerical experiments

❺ Scalability issues of iterative solution methods or how to balance the problem size and the number of processors in order to achieve optimal parallel efficiency

❻ Conclusions

Introduction:

Given:

(G1) A system to solve ($A\mathbf{x} = \mathbf{b}$)
where A is a large and sparse matrix.

(G2) A massively parallel machine;

The task:

(T1) Solve $A\mathbf{x} = \mathbf{b}$ as fast as possible
(without sacrificing the accuracy);

(T2) Use the machine to its full capacity.

Possible solution methods:

(S1) Direct solvers

(S2) Iterative solvers

- When very large scale numerical models are considered, the iterative solution methods are the only viable choice.
- Preconditioned methods...
- The preconditioned methods used on a parallel machine have to be of *optimal order*.

	Nonoptimal methods (considered to be easily parallelizable)	Optimal methods (considered not to be easily parallelizable)
which are those	Diagonally preconditioned CG Incomplete factorizations Domain Decomposition (some versions) SSOR	Multigrid Algebraic Multilevel Domain Decomposition (some versions)

The main differences between the two groups of methods are:

- the absence and, respectively, the presence of *global exchange of information* during each iteration;
- the *hierarchical* structure of all multilevel methods.

- The global information exchange is an essential ingredient for the optimal rate of convergence of preconditioned iterative solution methods. In such a way they capture the sharing of global information inherent in the physical methods being modelled.

But_1: The global information exchange for a parallel machine entails communication overhead.

But_2: The hierarchical structure of the optimal methods implies that the work is distributed over a range of meshes (levels, subdomains), some of which are "finer" and others are "coarser". If the degrees of freedom on the "coarsest" ones are less than the number of processors, some of the processors will be idle.

Consequences of implementing multilevel methods on massively parallel machines:

- communication overhead dominating the computation;
- idleness of processors.

$\{\mu,\nu\}$	Grid	PCG iterations	CPU busy time IBM RS/6000	CM-200
{1,3}	32^2	4	0.16	4.07
	64^2	4	0.73	29.38
	128^2	4	3.26	304.08
	256^2	4	14.92	no exper.

Table 1: The behavior of a straightforward implementation of the algebraic multilevel iteration method on two computers. Very disappointing!

Q: Can an optimal (or almost optimal) order method be implemented efficiently on a massively parallel computer?

An idea: Let the global spread of information take place on a coarse but not very coarse mesh.

⇒ As an illustration, an optimal order algorithm, a short version of the Algebraic Multilevel Iterations (AMLI) method will be considered, used as a solver for selfadjoint elliptic problems of second order, discretized using finite differences or finite elements.

Briefly: what is AMLI

Let $A^{(\ell)} = A$ be s.p.d. of order n_ℓ and ℓ be the number of levels. At each level k there is an active set of nodes, N_k:

$$N_0 \subset \ldots \subset N_{k-1} \subset N_k \subset \ldots \subset N_\ell.$$

We recursively construct a sequence of matrices

$$A^{(\ell-1)}, \ldots, A^{(k)}, A^{(k-1)}, \ldots, A^{(0)}$$

of geometrically decreasing orders $n_{\ell-1}, \ldots, n_k$, n_{k-1}, $\ldots$, n_0, such that

$$\frac{n_k}{n_{k-1}} = \rho_k \geq \rho > 1.$$

Each level matrix $A^{(k)}$ is purely algebraically determined by the previous level matrix $A^{(k+1)}$. The procedure for constructing $A^{(k)}$, $k = \ell - 1, \ldots, 1, 0$ is as follows:

$m1$: The matrix $A^{(k+1)}$ is split into a two-by-two block form

$$A^{(k+1)} = \begin{array}{cc} \begin{bmatrix} A_{11}^{(k+1)} & A_{12}^{(k+1)} \\ & \\ A_{21}^{(k+1)} & A_{22}^{(k+1)} \end{bmatrix} & \begin{array}{l} \} \; N_{k+1} \backslash N_k \\ \\ \} \; N_k \end{array} \end{array}$$

by reordering the unknowns according to a permutation rule chosen in advance and satisfying (0.1).

$m2$: ${A_{11}^{(k+1)}}^{-1}$ is approximated by another matrix $B_{11}^{(k+1)}$ which is symmetric, sparse, nonnegative and positive definite but of simpler structure and which preserves the action of ${A_{11}^{(k+1)}}^{-1}$ on a chosen positive vector. In order to do this, the matrix A is assumed to be an M-matrix - i.e., A is a Stieltjes matrix.

$m3:$ The matrix $A^{(k)}$ is then defined as

$$A^{(k)} = A_{22}^{(k+1)} - A_{21}^{(k+1)} B_{11}^{(k+1)} A_{12}^{(k+1)},$$

and is an approximation of the true Schur complement of $A^{(k+1)}$.

Having constructed the sequence of matrices $\{A^{(k)}\}_{k=0}^{\ell}$ *top-to-bottom* we define a corresponding sequence of preconditioners $\{M^{(k)}\}_{k=0}^{\ell}$ *bottom-to-top* as follows:

$$M^{(k+1)} = \begin{bmatrix} B_{11}^{(k+1)^{-1}} & 0 \\ & \\ A_{21}^{(k+1)} & I \end{bmatrix} \begin{bmatrix} I & B_{11}^{(k+1)} A_{12}^{(k+1)} \\ & \\ 0 & \widetilde{S}^{(k)} \end{bmatrix} \text{ for} k = 0, 1, \ldots \ell - 1,$$

where $M^{(0)} = A^{(0)}$. $\widetilde{S}^{(k)}$ is a matrix polynomial of degree ν_k which can vary with the level number:

$$\widetilde{S}^{(k)} = A^{(k)} \left[I - P_{\nu_k}(M^{(k)^{-1}} A^{(k)}) \right]^{-1}.$$

This particular choice of $\widetilde{S}^{(k)}$ and the structure of $M^{(k)}$ makes it possible to solve systems with $\{M^{(\ell)}\}$ performing only matrix-vector multiplications and vector additions. The degree of the polynomials P_{ν_k} can be chosen so that the spectral condition number $\kappa_\ell = \kappa(M^{(\ell)^{-1}} A^{(\ell)}) = O(1),\ \ell \to \infty$.

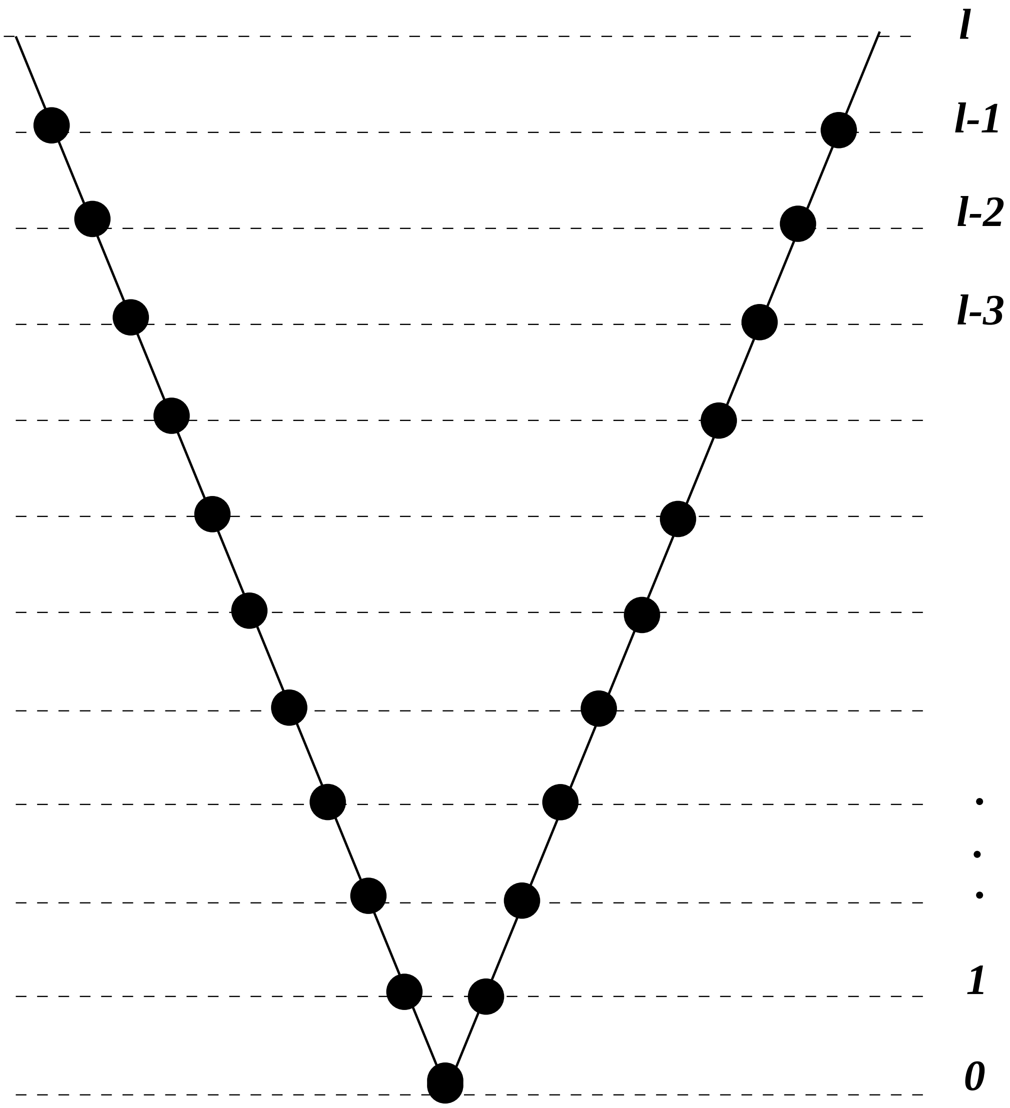
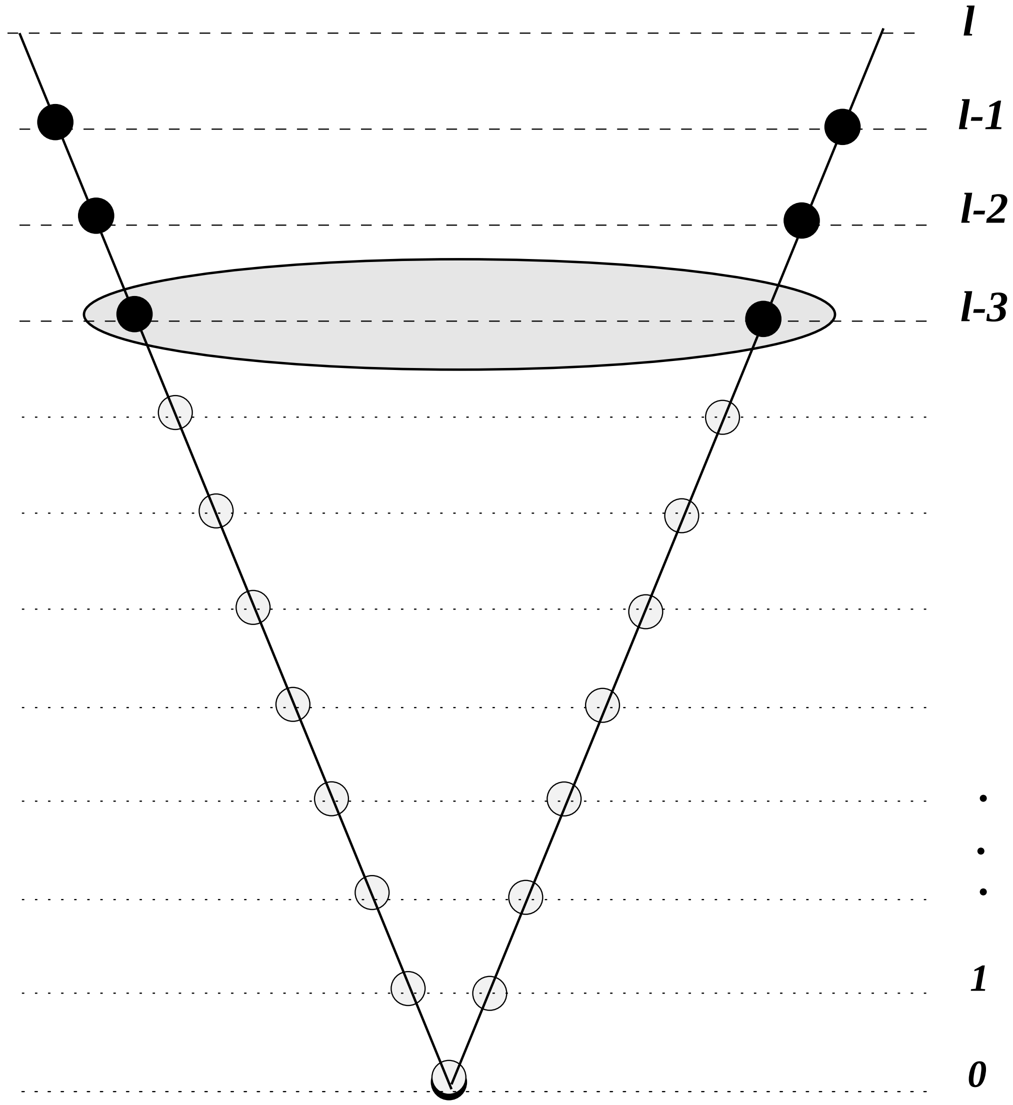


Figure 1: A picture of a V-cycle (occurring in Multigrid as well as in AMLI, when a polynomial of degree one os used on all levels)



Now, we want to "cut" the cycle on some level $k_0 > 0$.
For $k_0 = 0$ we reffer the method to as the *full length* AMLI and for some $k_0 > 0$, as the *short length* AMLI.
What we are left with on level k_0 is to solve systems with a matrix $A^{(k_0)} \equiv M^{(k_0)}$, which might be fairly large.
Two questions are to be answered now:

- Do we loose the optimal convergence rate?
- Can we preserve the optimal computational complexity?

Computational complexity

The computational complexity of the multilevel preconditioner using a polynomial of fixed degree ν, depends on ν in the following way. Assume that we start at the finest level ℓ and descend down to the coarsest level k_0. Then the computational complexity of one action of the preconditioner is

$$w_\ell \leq C(n_\ell - n_{\ell-1} + \nu(n_{\ell-1} - n_{\ell-2} + \cdots + n_{k_0+1} - n_{k_0})) + \nu^{\ell-k_0} C_{k_0} n_{k_0},$$

where n_s is the degree of freedoms at level s.

$\boxed{C}$ is independent of ℓ and k,

$\boxed{C}$ depends on the sparsity of the matrices.

$\boxed{C_{k_0}}$ depends on the solution method used when solving systems with the coarse mesh matrix $M^{(k_0)}$.

Consider for simplicity $n_s = 4^s = \frac{1}{4} n_{s+1}$, which holds for a regular mesh. Then, if $\nu < 4$,

$$w_\ell \leq n_\ell \left[\frac{3}{4} C \frac{1 - \left(\frac{\nu}{4}\right)^{\ell-k_0}}{1 - \left(\frac{\nu}{4}\right)} + C_{k_0} \left(\frac{\nu}{4}\right)^{\ell-k_0} \right],$$

or, for the work per meshpoint,

$$\frac{w_\ell}{n_\ell} \leq \frac{3C}{4-\nu} + \left(C_{k_0} - \frac{3C}{4-\nu} \right) \left(\frac{\nu}{4}\right)^{\ell-k_0}. \tag{1}$$

As is well known, the condition number of $M^{(\ell)^{-1}}A^{(\ell)}$ increases when k_0 decreases. Also, the communication overhead increases. We want, therefore, to choose k_0 as the largest number for which the cost for each action of the preconditioner is bounded, independent of ℓ.

Use a diagonally preconditioned conjugate gradient method to solve systems with $M^{(k_0)}$. As the condition number of the system is $O(n_{k_0})$, the number of iterations required is $C_0 n_{k_0}^{\frac{1}{2}}$ for some constant C_0, which does not depend on k_0. Since $n_{k_0} = 4^{k_0}$, to balance the two terms in (12) best for any value of ℓ, we must then let $n_{k_0}^{\frac{1}{2}} = \left(\frac{4}{\nu}\right)^{\ell-k_0}$, that is $2^{k_0} = \left(\frac{4}{\nu}\right)^{\ell-k_0}$ or

$$k_0 = \hat{k}_0 \equiv \left\lceil \frac{2-\log_2 \nu}{3-\log_2 \nu}\ell \right\rceil . \qquad (2)$$

From (13), the following values of $\hat{k}_0$ as a function of ν are found:

ν	3	2	1
$\hat{k}_0$	$\lceil \frac{2}{7}\ell \rceil$	$\lceil \frac{1}{2}\ell \rceil$	$\lceil \frac{2}{3}\ell \rceil$

Table 2: Values of coarse level number for the diagonally preconditioned method

(The exact value of $\hat{k}_0$ for $\nu = 3$ is $\approx 0.2933\ell$.) The work per meshpoint is then bounded by

$$\frac{w_\ell}{n_\ell} \leq \frac{3C}{4-\nu} + C_0.$$

Use the conjugate gradient method, preconditioned with a modified incomplete factorization or a block diagonal domain decomposition method.

For these methods the condition number of the preconditioned system grows as $C_0\, 2^{k_0}, k_0 \to \infty$.

In a similar way we find now that the optimal value to minimize the work per meshpoint of one action of the preconditioner satisfies $2^{\frac{k_0}{2}} = \left(\frac{4}{\nu}\right)^{\ell - k_0}$ or $k_0 = 2(\ell - k_0)(2 - \log_2 \nu)$, i.e.,

$$k_0 = \left\lceil \frac{4 - 2\log_2 \nu}{5 - 2\log_2 \nu}\, \ell \right\rceil ,$$

which leads to the following values for k_0 as a function of ν:

ν	3	2	1
k_0	$\lceil \frac{4}{9}\ell \rceil$	$\lceil \frac{2}{3}\ell \rceil$	$\lceil \frac{4}{5}\ell \rceil$

Table 3: Values of the coarse level number for the modified incomplete factorization method

The above results for k_0 as a function of ℓ can be further improved using polynomials of cyclically varying degrees. (For further details, cf. [3], [4], [5].)

★ ★ ★

Rate of convergence

The rate of the convergence of the short-AMLI method is studied in detail in [5]. The following cases are analyzed:

- block-diagonally preconditioned HBF matrices;
- full block-matrix factorization preconditioning for the multilevel HBF matrices;
- full block-matrix factorization preconditioning for Stieltjes matrices.

The analysis shows that, as for the classical formulation of the AMLI method, there exist a choice of the degree of the polynomials (usually some function of the so-called Cauchy-Bunyakowski-Schwarz (CBS) constant γ), for which the condition number $\varkappa({M^{(\ell)}}^{-1}A^{(\ell)})$ and, respectively, the rate of convergence is independent of the problem size. If, in addition, k_0 is chosen as shown before, then the short AMLI method is optimal both with respect to rate of convergence and computational complexity.

Remark: Using certain interpolation error estimates for the special model problem, $-\Delta u = f$, the following estimates of the condition numbers $\varkappa_\ell$ have been derived by Yserentant, Vassilevski, namely

$\varkappa_\ell \leq O(\ell^2)$ for two-dimensional problems,

$\varkappa_\ell \leq O(2^\ell)$ for three-dimensional problems.

However, when a short AMLI method is used we can get even smaller (and even optimal) condition numbers, in particular for three-dimensional problems. The upper bound derived in [5] shows that the condition number grows with ℓ as $\varkappa_{\ell,k_0}^{\frac{1}{2}} \leq const \left(\frac{1}{\sqrt{1-\gamma^2}}\right)^{\ell-k_0}$. Let $\gamma^2 = \frac{1}{2}$. If $k_0 = \frac{2}{3}\ell$, then $\varkappa_{\ell,k_0} \leq const\ 2^{\frac{\ell}{3}} = const\ h_\ell^{-\frac{1}{3}}$, where $h_\ell = 2^{-\ell}$ is the mesh-size on level ℓ. If $k_0 = \frac{4}{5}\ell$, then $\varkappa_{\ell,k_0} \leq const\, 2^{\frac{\ell}{5}} = const\, h_\ell^{-\frac{1}{5}}$. Hence, the condition number depends only weakly on the level number ℓ for the short-length V-cycle method. It is interesting to compare this with the above cited condition numbers. Although $2^{\frac{\ell}{3}}$ grows asymptotically faster than ℓ^2, note that for all practically viable values of ℓ, $2^{\frac{\ell}{3}}$ is much smaller than ℓ^2. The 'hidden' constants in the order symbols are about equal.

Some recent research shows that also for the V-cycle the condition number can be fully stabilized (cf. [7]).

Remark: A similar computational complexity analysis can be done for three space dimensional problems. One finds that $\frac{w_\ell}{n_\ell} \leq const$ if $k_0 = \left\lceil \frac{3-\log_2 \nu}{4-\log_2 \nu} \ell \right\rceil$ and, for $\nu = 1$, the condition number grows as $\varkappa_{\ell,k_0} \leq const \left(\frac{1}{\sqrt{1-\gamma^2}} \right)^{\frac{\ell}{4}}$. For $\gamma^2 = \frac{5}{7}$ (which is the value of the CBS-constant for the bilinear form $a(u,v)$ corresponding to the differential operator $-\Delta u + a_0 u$ for three-dimensional problems, discretized using regular tetrahedra and piecewise quadratic basis functions) we find $\varkappa_{\ell,k_0} \leq const \left(\frac{7}{2}\right)^{\frac{\ell}{4}} \sim 1.37^\ell$, which increases much slower than 2^ℓ (recall that for the classical hierarchical basis function matrix the condition number is $O(2^\ell)$.

Numerical experiments

The following test problem was implemented on CRAY T3D with 256 processors, available at EPFL, Lausanne.

$$-\sum_{i=1}^{3} \frac{\partial}{\partial x_i} \left(a \frac{\partial u}{\partial x_i} \right) = f, \tag{3}$$

where a is a constant and $a = \varepsilon$ in a subset $\widetilde{\Omega} \in \Omega \subset \mathbf{R}^3$ and $a = 1$ elsewhere ($\varepsilon >> 1$ or $\varepsilon << 1$). The boundary conditions are a combination of homogeneous Dirichlet and Neumann boundary conditions. The problem was discretized using trilinear finite element basis functions are used on parallelepipeds. The AMLI preconditioner is based on a matrix corresponding to central finite-difference approximation of the same problem. The discretization is such that there are meshlines along the edges of $\widetilde{\Omega}$. The outer solution method is preconditioned conjugate gradient (PCG) method with a short-version AMLI preconditioner (V-cycle), for some properly chosen coarsest level number k_0. The inner solver is a diagonally preconditioned conjugate gradient method. (Detailed description of these experiments can be found in [6].)

No. of unkn. Total *no.* of levels	Coarse level *no.*	*No.* of outer iter.	Average *no.* of inner it.	Total execution time on *no.* of PEs (in *sec*)				
				16	32	64	128	256
	8	50	8	6/5/2	-	-	-	-
$(2^5)^3 = 32768$	9	39	13	5/5/2	-	-	-	-
13 levels	**10**	$\boxed{29}$	18	4/4/2	2/2/1	1/1/0	-	-
	11	29	29	6/6/4	-	-	-	-
	9	57	7	54/51/12	-	-	-	-
$(2^6)^3 = 262144$	**10**	$\boxed{49}$	10	49/46/15	25/24/8	13/12/4	7/7/2	-
16 levels	11	54	16	61/58/29	-	-	-	-
	12	38	24	52/49/30	27/26/16	16/14/9	-	-
	12	64	13	-	-	-	76/73/28	-
$(2^7)^3 = 2097152$	**13**	$\boxed{50}$	19	-	-	-	67/64/33	35/35/17
19 levels	14	54	30	-	-	-	91/88/59	-
	15	37	45	-	-	-	85/79/61	

Table 4: 3D AMLI on CRAY T3D. Trilinear finite element (27-point) stiffness matrix preconditioned by finite difference (7-point) matrix; $n_{k+1}/n_k = 2$.
Read the three times (T1/T2/T3) in the table as total execution time (T1), time for the AMLI-preconditioned PCG method (T2) and time for the inner solver (T3).

Important to observe:

- Among the choices of coarsest level number k_0 there is an optimal one for which the total execution time is minimized.
- The method scales nicely with increasing the problemsize and the number of processors used.

Do we loose something because of the short-length recursion?

No!

Do we gain anything from the short-length recursion?

Yes, we have only gains:

(a) better condition numbers;

(b) more robustness with respect to problem parameters;

(c) no idleness of processors;

(d) less communication overhead required.

$\boxed{\text{Parallel efficiency}}$

or how to balance properly the problem size with the number of processors available

The fact that the complexity of the problems can grow very fast with the problem size requires a careful choice of the solution algorithm. We will advocate the strategy to look for and to use $\boxed{\textit{scalable algorithms}}$ with an $\boxed{\textit{optimal parallel complexity}}$.

Definition 1: An algorithm is scalable if the solution time remains constant when the number of computer processors increases linearly with the number of unknowns parameters (N) in the problem to be solved.

Definition 2: An algorithm is said to have optimal order parallel complexity if its efficiency index

$$E_p \to 1 \text{ when } N \to \infty.$$

$$E_p = \frac{T_1}{p\,T_p}$$

where

T_1 is the computing time of an optimal order computational complexity algorithm on a single processor, i.e., its complexity grows linearly with N,

T_p is the computing time when the same algorithm is implemented on a p-processor computer architecture.

Remarks on the definition of efficiency

It turns out that, due to the communication overhead, stated in Definition 1 is too much to ask for. Instead, we adopt the concept that for an algorithm to be scalable

- ↠ it must have an optimal order of computational complexity, i.e., for which the complexity grows linearly with the problemsize N, and
- ↠ for which it is possible to find a properly increasing (in general less than linearly) number of processors such that the efficiency approaches one with increasing problem size.

Clearly, when the efficiency index is one, all the processors are used to full efficiency and the speedup $\frac{T_1}{T_p}$ is equal to its maximum value p.

Since the optimal value of p in general grows less than linearly with N, the computing time will also grow with some power (less than one) on N. This latter holds irrespectively of the number of processors used. It turns out that T_p is minimized for a number of processors which grows slightly faster than for the number which gives asymptotically maximal efficiency (typically with a $\log N-$ or even only $\log\log N-$factor).

To start with, consider a diagonally preconditioned conjugate gradient method to solve an elliptic difference equation problem, implemented on a $\sqrt{p} \times \sqrt{p}$ mesh array computer. (Below, T_p is the time per iteration.)

$$T_p = \frac{T_1}{p} + c \sqrt{p}.$$

computing time / proportionality constant / communication time

Obviously, there exists a number of processors p^* which minimizes T_p and one finds

$$p^* = \left(\frac{2T_1}{c}\right)^{\frac{2}{3}}, \; T_{p^*} = \frac{3}{2^{\frac{2}{3}}} \left(c^2 T_1\right)^{\frac{1}{3}}.$$

For the value of p, which minimizes the computing time, there holds

$$E_{p^*} = \frac{1}{3}$$

which, although independent of p is very unsatisfactory.

Let now $p = \sigma p^*$ for some $\sigma \leq 1$. Then the corresponding formulas become:

$$T_p = \frac{1 + 2\sigma^{\frac{3}{2}}}{3\sigma} T_{p^*}, \;\; E_p = \frac{1}{1 + 2\sigma^{\frac{3}{2}}}.$$

Note: when p grows like

$$p = \sigma p^* = \sigma \left(\frac{2CN}{c} \right)^{\frac{2}{3}},$$

the efficiency E_p does not depend on the problem size (another definition of scalable algorithm).

Furthermore, if we let σ decay to zero slowly with N, such as

$$\sigma = \frac{1}{\log N} \text{ or } \sigma = \frac{1}{\log \log N},$$

then E_p approaches its optimal value one, as N increases.

However,

• the time per iteration $T_{p^*} = O(N^{\frac{1}{3}})$ grows with N.

• the number of iterations grows also with N (for d-dimensional second order elliptic problem it grows as $N^{\frac{1}{d}}$.

So, the total time for the diagonally preconditioned method on a parallel mesh array computer grows at least as

$$O(N^{\frac{1}{d}+\frac{1}{3}})$$

and this is irrespective of the number of processors used.

Complexity analysis of the Short AMLI method on a mesh array

Similarly, for the short AMLI method we have

$$T_p = \frac{T_1}{p} + c_1\, i_{in}\sqrt{p} + c_2\, i_{in}\sqrt{\frac{N_{k_0}}{p}} + c_3\sqrt{p} + c_4\sqrt{\frac{N}{p}}.$$

global communication for the inner iterations

nearest neighbor communications for the inner iterations

outer iterations

The above formula can be simplified under some assumptions:

$$T_p = \frac{T_1}{p} + c_1\sqrt{N_{k_0}}\sqrt{p} + c_2 N_{k_0}\frac{1}{\sqrt{p}} + c_1\sqrt{p}.$$

There are two parameters involved in the above expression for T_p - p and N_{k_0}. Clearly, the best is to let N_{k_0} be as small as possible, that is $N_{k_0} = p$.

Then

$$T_p = \frac{T_1}{p} + c_1 p + (c_1 + c_2)\sqrt{p}.$$

Asymptotically, this is minimized when $\frac{T_1}{p} + c_1 p$ is minimized, i.e., for

$$p^* = \left(\frac{T_1}{c_1}\right)^{\frac{1}{2}}.$$

For $T_1 = O(N)$ we find $T_{p^*} = O(N^{\frac{1}{2}})$. Since the number of iterations for the V-cycle method is $O(\log N)$, the total computing time is $O(N^{\frac{1}{2}} \log N)$. The efficiency for $p = p^*$ is found to be

$$E_{p^*}^{(it)} = \frac{1}{2 + (w_1 + w_2)(w_1^3 T_1^{(it)})^{-\frac{1}{4}}} < \frac{1}{2}.$$

If, again, we let $p = \sigma p^*$ for some $\sigma < 1$, then

$$E_p^{(it)} = \frac{1}{1 + \sigma^2 + (w_1 + w_2)\sigma^{\frac{3}{2}}(w_1^3 T_1^{(it)})^{-\frac{1}{4}}}.$$

In this case $T_p^{(it)}$ grows like $O\left(\frac{\sqrt{N}}{\sigma}\right)$.

Hence, the total execution time is $O(N^{\frac{1}{2}} \log N)$.

★ ★ ★

To summarize: A general observation is that if we let the number of processors grow slightly slower than the number which minimizes the parallel computing time then the efficiency approaches the value 1 when the problem size increases. Since the (short) AMLI method has optimal, or for the V-cycle, close to optimal computational complexity on a single processor, we see that the method has optimal order parallel complexity. It is also seen, that the major factor contributing to the growth of the communication overhead is the amount of scalar products performed while solving the inner system.

Conclusions:

- Shortening the length of the recursion is an improvement of the original AMLI method (not only but) specially for parallel implementations.
- The idea is general and applies to other multilevel-type methods like Multigrid.
- By choosing the number of processors (slightly) smaller than the optimal number of p for smallest computer time, we can come asymptotically arbitrarily close to maximal speedup.
- To achieve the above optimal computer times we must choose the coarsest mesh Ω_{k_0} properly and such that there is at least one mesh-point mapped onto each processor, i.e., $p \leq N_{k_0}$. Furthermore, the bigger k_0 is, the smaller the condition number of the preconditioned matrix is. In practice, however, $p = N_{k_0}$ usually is not enough, the problem size has to be sufficiently large to get (close to) optimal efficiency.
- The additive version possess even better parallelization properties and is now studied in more detail.

References

[1] O. Axelsson and P. Vassilevski, Algebraic Multilevel Preconditioning Methods I, *Numer. Math.*, 56 (1989), 157-177.

[2] O. Axelsson and P. Vassilevski, Algebraic Multilevel Preconditioning Methods II, *SIAM J. Numer. Anal.*, 27 (1990), 1569-1590.

[3] P. Vassilevski, Hybrid V-cycle Algebraic Multilevel Preconditioners, *Math. Comp.*, 58 (1992), 489-512.

[4] O. Axelsson and M. Neytcheva, Algebraic Multilevel Iteration Method for Stieltjes Matrices, *Num. Lin. Alg. Appl.*, 1 (1994), 213-236.

[5] M. Neytcheva, Arithmetic and Communication Complexity of Preconditioning Methods. Ph.D. thesis, September 1995, Nijmegen.

[6] M. Neytcheva, A. Padiy, M. Mellaard, K. Georgiev and A. Axelsson, Scalable and Optimal Iterative Solvers for Linear and Nonlinear Problems. Report No. 9613 (March 1996), Department of Mathematics, University of Nijmegen.

[7] O. Axelsson, Stabilized V-cycle methods, *J. CAM*, 1996 (to appear).