An algebraic multigrid method for solving the Laplacian equations used in image analysis

Xin He *

1 Introduction

The inhomogeneous Laplace equation with internal Dirichlet boundary conditions has recently appeared in many applications arising from image segmentations, image colorization, image filtering and so on. Efficient solutions of (anisotropy) Laplacian equations have been studied intensively in numerical analysis world. In this project paper, I apply the known algorithms, especially an algebraic multigrid method, to solve Laplacian equations used in image analysis.

2 General setting and preliminaries

Some notations of a graph need to be fixed firstly. A graph consists of a pair $G = (V, E)$ with vertices (nodes) $v \in V$ and edges $e \in E \subseteq V \times V$. An edge e, spanning two vertices v_i and v_j, is denoted by e_{ij}. A weighted graph assigns a value (also called a weight) to each edge. The weight of an edge e_{ij} is denoted by w_{ij}. The degree of a vertex is $d_i = \sum w_{ij}$ for all edges e_{ij} incident on v_i. Here, I assume that our graph is connected and undirected (i.e., $w_{ij} = w_{ji}$).

For generality, I will consider finding a solution to the equation

$$(L + D)\mathbf{x} = \mathbf{f},$$

where D is a diagonal matrix with arbitrary nonnegative function on the diagonal, and L represents the Laplacian matrix defined as

$$L_{ij} = \begin{cases} d_i & if\ i = j \\ -w_{ij} & if\ v_i\ and\ v_j\ are\ adjacent\ nodes \\ 0 & otherwise \end{cases}$$

where L_{ij} is indexed by vertices v_i and v_j. In the various imaging applications, the edge weights are used to encode the image structure. Although in this paper the weight of an

*Department of Information Technology, Uppsala University.Email: he.xin@it.uu.se

edge e_{ij} is chosen to be its length, another very common weighting function is given by

$$w_{ij} = e^{-\beta(g_i - g_j)^2},$$

where g_i indicates the image intensity at pixel i and β is a parameter.

Additionally, I will permit internal Dirichlet boundary conditions at a set of nodes $V_B \in V$ where the solutions on these nodes have been prescribed in advance. Among the several ways to enforce the boundary conditions, in this paper I use the following method. For example, if $v_k \in V_B$, I zero the kth row and column of the coefficient matrix $L + D$ and then specify the (k, k) element to be unit. Corresponding, the kth component of the right hand vector, i.e., $\mathbf{f}(k)$, should be given the prescribed solution on the node v_k.

3 Numerical solutions

How to efficiently solve the discrete Laplacian equation is the kernel of this project paper. The Laplacian coefficient matrix is large but sparse. Since each node is connected via an edge to its neighboring nodes with a 4-connected structure in 2D, or 6-connected structure in 3D, there are at most 4 nonzero elements per row in 2D and 6 nonzero elements per row in 3D. Due to the preservation of sparsity, direct solution specified for large and sparse linear systems is suitable. Because the Laplacian matrix is symmetric and positive definite, conjugate gradient (CG) iterative solution and preconditioned conjugate gradient (PCG) with incomplete Cholesky decomposition factors as preconditioner are also choices. Additionally, the algebraic multigrid (AMG) method is another very efficient approach. The AMG method I use here is an aggregation-based algebraic multigrid method (AGMG), see [1, 2, 3]. The implementation is in **Fortran** and **Matlab** interface is provide. Therefore, its performance in time is comparable with that of the 'backslash' direct solver in **Matlab**.

The image graph used here is the homogeneous Cartesian mesh in 2D. The distance between any two adjacent nodes is the same and the length of an edge is h. The regular refinement is applied, namely, the edge is equally cut into two edges by one refinement. In my computation the diagonal matrix L is chosen to be $L = h^2 I$, where I denotes the identity matrix. In this paper, I choose the weight function of an edge to be its length. Therefore, in the homogeneous Cartesian mesh, for generality reason the weight function w_{ij} is set to be unit. The Dirichlet boundary can be specified randomly within the graph. Without loss of generality, in this paper I consider the whole graph boundary as the Dirichlet boundary.

To compare the efficiency of direct and iterative solutions, i.e., CG, PCG and AGMG when solving the discrete Laplacian equation, I check the CPU time (in seconds) for all methods and the iteration number of iterative solutions. The results are presented in Table 1. The CPU time of the direct solver and multigrid AGMG method increase by 4 times when refining the mesh once, i.e., the size of Laplacian matrix increasing by 4 times. Meanwhile, the iterations of AGMG keep the same. This observation show that the AGMG is of the optimal computational complexity and is independent of the mesh refinement. Another important conclusion based on Table 1 is that the AGMG is the most efficient approach among the several options. The high efficiency of AGMG is due to its

Table 1: The comparison between direct solver, CG, PCG and multigrid AGMG methods

size(L)	Direct solver	CG		PCG		AGMG	
	CPU	iter.	CPU	iter.	CPU	iter.	CPU
16641	0.61	264	0.48	113	0.48	19	0.14
66049	3.25	530	3.51	226	3.14	19	0.35
263169	14.88	1071	25.13	456	22.72	19	1.48
1050625	62.45	2150	219.34	917	215.34	20	6.81
4198401	264.76	4754	1927.14	2024	1812.08	20	28.97

original design for large systems arising from discretization of scalar second order elliptic PDEs, e.g., the Laplacian equations.

By one refinement, the CPU time increase by around 8 times and the iterations increase by 2 times for the CG and PCG iterative solutions. Although the iterations of PCG are almost the half of that of CG method, the CPU time for the two methods are almost the same. The reason is that compared to the CG method, at each iteration the PCG method need to solve a linear system with the preconditioner, which takes time. The convergence histories of the CG, PCG and AGMG solutions are plotted in Figure 1.

All the results are computed using **Matlab** implementation on a server with 16 cores AMD Opteron 6274 CPU provided by SNIC through Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX).

4 Conclusion and future work

Based on the numerical experiments I conclude that the aggregation-based algebraic multigrid method (AGMG) used in this paper is highly efficient for solving the Laplacian equations used in image analysis. The AGMG solver is used here as a black box and purely algebraic, that is, no information has to be supplied besides the system matrix and the right hand vector. The test of the efficiency of AGMG algorithm for solving the inhomogeneous Laplacian equations, namely, the weights are not uniform, is considered as a future work.

References

[1] A. Napov, Y. Notay, An algebraic multigrid method with guaranteed convergence rate, Report GANMN 10-03, Université Libre de Bruxelles, Brussels, Belgium, 2010 (Revised 2011).

[2] Y. Notay, An aggregation-based algebraic multigrid method, *Electron. T. Numer. Ana.*, **37** (2010), 123-146.

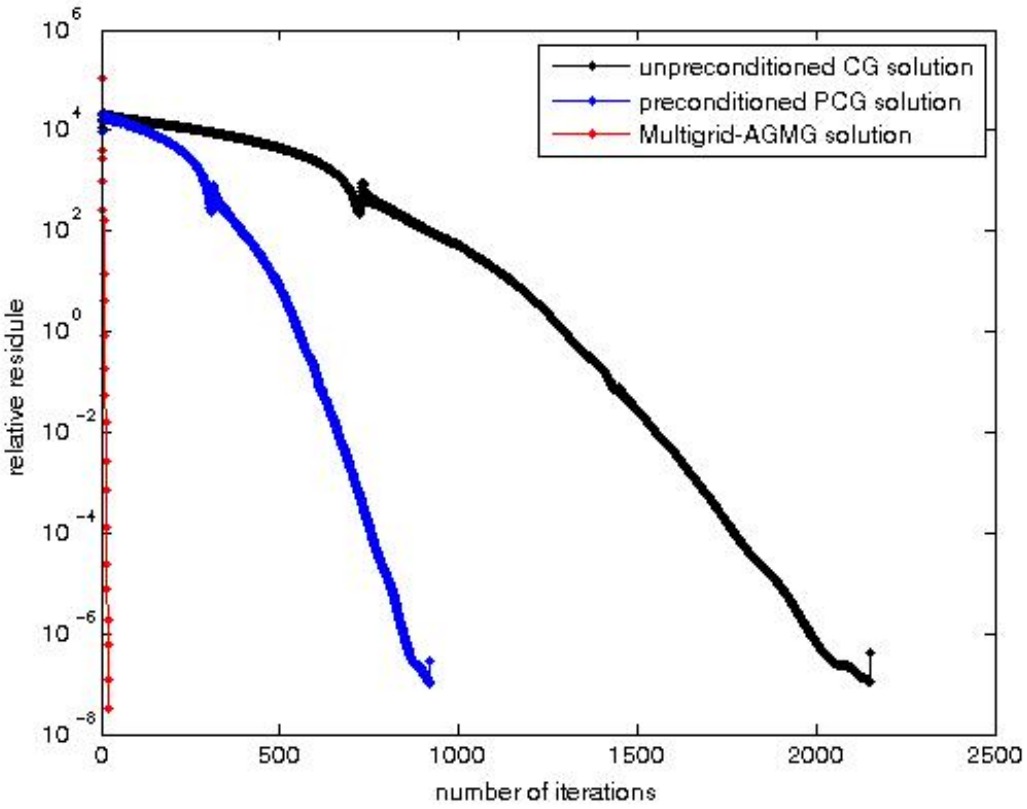

Figure 1: The convergence of CG, PCG and AGMG solutions, $Size(L) = 1050625$

[3] Y. Notay, Aggregation-based algebraic multigrid for convection-diffusion equations, Report GANMN 11-01, Université Libre de Bruxelles, Brussels, Belgium, 2011.