A SIMPLE AND EFFICIENT PARALLEL IMPLEMENTATION OF THE ILU PRECONDITIONER FOR POISSON EQUATIONS

CHOHONG MIN AND FREDERIC GIBOU

1. Introduction

Incomplete factorization preconditioners such as ILU and MILU are robust techniques for solving linear systems, and they are well-known to be very effective for M-matrices, discrete analalog of the Laplace operator. However they are difficult to parallelize efficiently. Various techniques have been developed to parallelize these preconditioners, such as multi-colour ordering and subdomain partitioning. In this paper, we consider the standard central finite differences for the Poisson equation $\Delta u = f$, and focus on a diagonal ordering that naturally enables parallel computations.

Consider the standard 5-point finite difference method for solving the Poisson equation $-\Delta u = f$

$$\frac{-u_{i+1,j} - u_{i,j+1} + 4u_{i,j} - u_{i,j-1} - u_{i-1,j}}{h^2} = f_{i,j}.$$

The variables $[u_{ij}]$ form a linear system $Au = f$. The matrix A, which is a discretization of the negative Laplace operator, satisfy many useful properties: its diagonal elements are all positive, its off-diagonal elements are all negative, and it is symmetric positive definite and diagonally dominant. In short, it is an M-matrix and its ILU factorization is guranteed to exist **?**]. We take the standard lexicographic ordering for the variables as depicted in the left picture of figure **??**. With the ordering and the regular stencil structure of the finite difference method, the ILU preconditioner for the matrix takes a simple form $M = (L + D) D^{-1} (D + U)$, where L and U are the stric lower and upper triangular matrices of A and D is a diagonal matrix that is determined as

$$\begin{aligned}
&D := D_0 \\
&\text{for } (i,j) = (1,1), (1,2), \cdots, (imax, jmax) \\
&\quad D_{(i+1,j),(i+1,j)} := D_{(i+1,j),(i+1,j)} - \left(A_{(i,j),(i+1,j)}\right)^2 / D_{(i,j),(i,j)} \\
&\quad D_{(i,j+1),(i,j+1)} := D_{(i,j+1),(i,j+1)} - \left(A_{(i,j),(i,j+1)}\right)^2 / D_{(i,j),(i,j)},
\end{aligned}$$

where D_0 is the diagonal matrix of A and the index increases in the lexicographic ordering **? ?**]. Consider the ILU preconditioned conjugate gradient method, one of the most efficient ways to solve generic Poisson problems together with multi-grid method **?**].

	memory accusess	arithematic operations	total
inner product $(r \cdot z)$	$2N^3$	$2N^3$	$4N^3$
matrix-by-vector multiplication (Ap)	$15N^3$	$13N^3$	$28N^3$
vector linear sum $(z + \beta p)$	$2N^3$	$2N^3$	$4N^3$
preconditioning $\left(M^{-1}r\right)$	$18N^3$	$16N^3$	$34N^3$

TABLE 1. Operation counts of the PCG routine in N^3 grid

$u := 0$, $r := f$, $z := M^{-1}r$, $p := z$
Until $r \cdot z$ is small enough
  $\alpha := r \cdot z / p \cdot Ap$, $rz^{old} := r \cdot z$
  $u := u + \alpha p$
  $r := r - \alpha Ap$
  $z := M^{-1}r$
  $\beta := r \cdot z / rz^{old}$
  $p := z + \beta p$

One iteration spends one inner product, one matrix-by-vector multiplication, three vector linear summations, and one preconditioning. All the calculations but the preconditioing are trivial to parallelize, and the preconditioning consumes the largest of computation time, as shown in table **??**.

The preconditioner $M = \left(I + LD^{-1}\right)(D + U)$ is the product of a lower triangular matrix and a upper triangular matrix, and the preconditioning $z := M^{-1}r$ is implemented by forward substitution followed by backward substitution. In either sustitution, the vector elements are sequentially updated following their ordering, thus its direct parallelization is not possible in the general setting. Using the specific stencil structure of the finite difference method, however, its parallization becomes possible. We introduce a simple and efficient parallelization algorithm of the preconditioning in the next section.

2. METHOD

With the lexicographic ordering, the support of the lower triangular matrix $\left(I + LD^{-1}\right)$ is the two points to lower left direction and that of the upper triangular matrix $(D + U)$ is the other two points to upper right direction, as depicted in figure **??**. The forward substitution of the lower triangular matrix updates grid nodes in the increasing lexicographic ordering, so that a grid node should be updated after all the grid nodes with smaller indices.

The two orderings in figure **??** are equivalent to the forward substitution, because each grid node is updated after its two neighboring to lower left direction. Notably in the diagonal ordering the grid nodes of each group are indenpendent to each other according to the binary relation defined by the graph of the matrix, and can be updated in parallel.

- things to say :
- diagonal groups vary in sizes
- many forks and merging in multi-threading
- in 3d the shape is hexagon
- not restricted to rectangular domains,

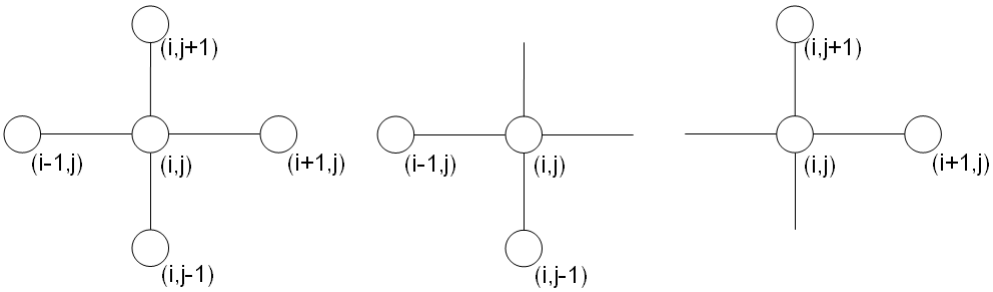

FIGURE 2.1. stencil points of the matrices A(left), $\left(I + LD^{-1}\right)$(middle), and $(D + U)$(right)

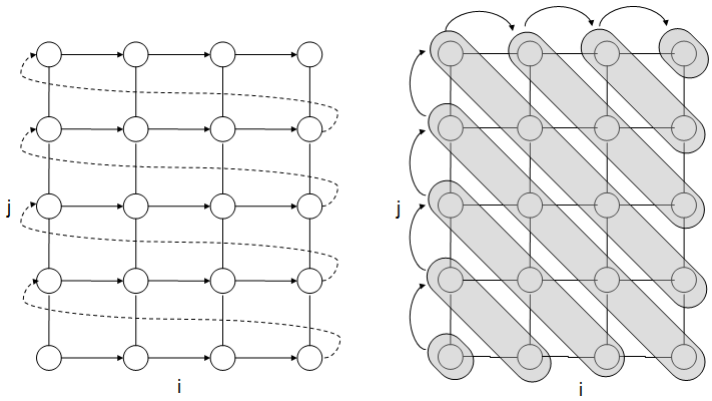

FIGURE 2.2. the lexicographic ordering and a digonal-scan ordering

In three dimensions, the Poisson equation $-\Delta u = f$ is approximated by the standard 7-point finite difference equation,

$$\frac{\begin{array}{l}-u_{i-1,j,k} - u_{i,j-1,k} - u_{i,j,k-1} \\ -u_{i+1,j,k} - u_{i,j+1,k} - u_{i,j,k+1}\end{array} + 6u_{i,j,k}}{h^2} = f_{i,j,k}.$$

We assume the Dirichlet boundary condition, so that the value of u_{ijk} is given whenever each index i, j, k is zero or the maximum of the index. In the lexicographic ordering, the forward substitution precedes as the following iterations.

for $i = 1 : imax$
for $j = 1 : jmax$
for $k = 1 : kmax$
$u_{i,j,k} := f\left(u_{i,j,k},\, u_{i-1,j,k},\, u_{i,j-1,k},\, u_{i,j,k+1}\right)$

As in two dimensions, we group grid nodes (i, j, k) by their index sum $i + j + k$, and perform the forward substitution in the following iterations.

for $s = 3 : imax + jmax + kmax$
for each (i, j, k) such that $i + j + k = s$
$u_{i,j,k} := f\left(u_{i,j,k},\, u_{i-1,j,k},\, u_{i,j-1,k},\, u_{i,j,k+1}\right)$

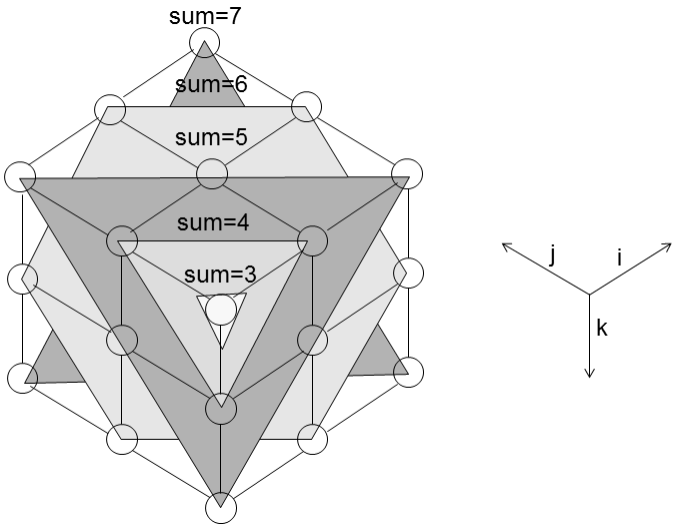

FIGURE 2.3. diagonal ordering for three dimensional grid

Lemma 1. *The Gauss-Seidel updates in the lexicographic and diagonal ordering are equivalent.*

Proof. Note that each iteration visits every grid node, and once for each. We prove it by mathematical indeuction on the sum. When the sum $s = 3$, u_{111} is updated from its value and given boundary values, so the calculations in both orderings are the same. Assume that the argument is true with index sum s. Now consider a grid index (i, j, k) with $i + j + k = s + 1$. The update of u_{ijk} depends on itself, which is not updated yet, and the three grid nodes whose index sum is s. By the assumption, all the arguments of the update function are the same, and the update of u_{ijk} are the same. Since the index is arbitrary with index sum s+1, the argument is true for the index sum $s + 1$.

□

2.1. **Irregular domain.** The parallel implementaion of ILU preconditioner is not restricted to rectangular domains. As depicted in figure **??**, the diagonal ordering and grouping is taken to all the grid nodes while grid nodes inside the domain are maked black. In the process of forward and backward subtitutions, only marked nodes are updated in parallel. Since marked nodes are irregularly distributed, it is not a simple matter how to divide the band into several subbands of equal workload for parallel process. We simply divide the band into equally sized subbands. Surely the efficiency depends on the diagonal alignment of the domain. Since the Laplace operator is rotational invariant, we may take diagonal or anti-diagonal ordering depending on the shape of the irregular domain.

3. EXAMPLES

3.1. **Dirichlet boundary condition.** domain : $\left\{(x, y, z) \in [-1, 1]^3 \,|\, \sqrt{x^2 + y^2 + z^2} \leq .7\right\}$
exact solution : $u(x, y, z) = \sin(x)\cos(y)\, e^{-x^2 - z^2}$
weight : $w(x, y, z) = 1 + \frac{1}{2}\sin(x + y + z)$

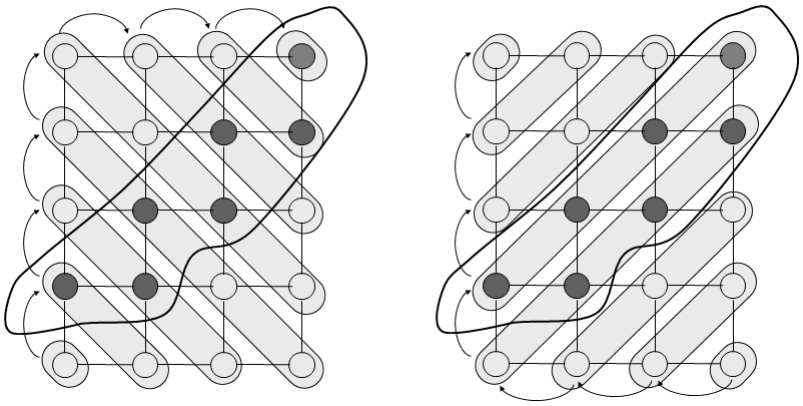

FIGURE 2.4. Grid nodes inside the domain are marked black, and the nodes outside white. The parallelization efficiency depends on the diagonal alignment of the domain.

we solve $-\nabla \cdot (w\nabla u) = f$ with Dirichlet boundary condition on the domain. PCG routine is iterated until the convergence criteria $r^n \cdot z^n < 10^{-14} r^0 \cdot z^0$.

grid	L^1 error	order	L^∞ error	order	iteration
25^3	1.25×10^{-3}		4.59×10^{-4}		17
50^3	3.29×10^{-4}	1.92	1.05×10^{-4}	2.12	33
100^3	7.53×10^{-5}	2.12	2.63×10^{-5}	1.99	60
200^3	1.95×10^{-5}	1.95	6.48×10^{-6}	2.02	107

	PC				Cluster				
# of threads	1	2	3	4	1	2	4	8	16
run time	86.9	53.7	48.2	42.7	154	77.2	40.1	23.2	15.6
speed up	1	1.62	1.80	2.03	1	1.99	3.82	6.61	9.85

3.2. **Neumann boundary condition.** We solve $-\Delta u = f$ in a circular domain of center $(0,0)$ and radius $\frac{1}{2}$. The exact solution is taken as $u = \left(r^2 - \frac{1}{4}\right)^3$, and the right hand side is taken accordingly. PCG routine is iterated until the convergence criteria $r^n \cdot z^n < 10^{-14} r^0 \cdot z^0$. The linear system of the Neumann boundary condition is nonsingular with one dimensional kernel. We solve the linear system on the vector space of sum zero which elliminates the Kernel thus guaranteeing a unique solution.

grid	L^1 error	order	L^∞ error	order	iteration
25^3	2.14×10^{-3}		6.85×10^{-4}		24
50^3	4.33×10^{-4}	2.30	1.82×10^{-4}	1.91	62
100^3	8.98×10^{-5}	2.27	3.59×10^{-5}	2.34	63
200^3	2.40×10^{-5}	1.90	9.27×10^{-6}	1.95	98

	PC				Cluster				
# of threads	1	2	3	4	1	2	4	8	16
run time	82.0	51.4	46.2	41.1	154	74.2	38.0	21.9	14.4
speed up	1	1.59	1.77	2.00	1	2.07	4.05	7.03	10.7

References

[1] M Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. *Journal of Computational Physics*, 182(2):418–477, November 2002.

[2] A Chorin. A Numerical Method for Solving Incompressible Viscous Flow Problems. *J. Comput. Phys.*, 2:12–26, 1967.

[3] Iain S. Duff and Gérard a. Meurant. The effect of ordering on preconditioned conjugate gradients. *Bit*, 29(4):635–657, December 1989.

[4] F Gibou. A Second-Order-Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains. *Journal of Computational Physics*, 176(1):205–227, February 2002.

[5] C Min and F Gibou. Geometric Integration Over Irregular Domains with Application to Level Set Methods. *J. Comput. Phys.*, 226:1432–1443, 2007.

[6] Yen Ting Ng, Chohong Min, and Frédéric Gibou. An efficient fluid–solid coupling algorithm for single-phase flows. *Journal of Computational Physics*, 228(23):8807–8829, December 2009.

[7] S Osher and R Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[8] S Osher and J Sethian. Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on {H}amilton-{J}acobi Formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[9] Y Saad. *Iterative methods for sparse linear systems*. PWS Publishing, 1996.

[10] J A Sethian. *Level set methods and fast marching methods*. Cambridge University Press, 1999.