

CS 689-001
Parallel Numerical Computation

Lecture 7*
Preconditioned Iterations

Professor Jun Zhang
Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046
March 22, 1999

*Reference: *Iterative Methods for Sparse Linear Systems*, by Yousef Saad, 1996.

Preconditioned Conjugate Gradient

Suppose M in an incomplete Cholesky form

$$M = LL^T,$$

then the preconditioner can be split as

$$L^{-1}AL^{-T}x = L^{-1}b, \quad u = L^{-1T}b$$

to preserve symmetry so CG can be applied

The M inner product is

$$(u, v)_M = (Mu, v) = (u, Mv)$$

and $M^{-1}A$ is self-adjoint with this norm

$$(M^{-1}Au, v)_M = (Au, v) = (u, Av) = (u, M(M^{-1}A)v)$$

We can redefine CG using M inner product

Preconditioning

When directly applied, iterative method are not robust for solving realistic problems

Preconditioning techniques transform the original system into a one that is easier to solve with an iterative method. Instead of solving

$$Au = b$$

We are solving

$$M^{-1}Au = M^{-1}b, \quad \text{or}$$

$$AM^{-1}x = b, \quad \text{and} \quad u = M^{-1}x$$

They are called left and right preconditionings respectively. We first consider the case that A and M are symmetric positive definite

Rewrite PCG

Denote residual and preconditioned residual

$$r^{(j)} = b - Au^{(j)}, \quad z^{(j)} = M^{-1}r^{(j)}$$

components of PCG can be reformulated

$$\begin{aligned} \alpha_j &= (z^{(j)}, z^{(j)})_M / (M^{-1}Ap_j, p_j)_M \\ u^{(j+1)} &= u^{(j)} + \alpha_j p_j \\ r^{(j+1)} &= r^{(j)} - \alpha_j p_j \quad \text{and} \quad z^{(j+1)} = M^{-1}r^{(j+1)} \\ \beta_j &= (z^{(j+1)}, z^{(j+1)})_M / (z^{(j)}, z^{(j)})_M \\ p_{j+1} &= z^{(j+1)} + \beta_j p_j \end{aligned}$$

Note that $(z^{(j)}, z^{(j)})_M = (r^{(j)}, z^{(j)})$ and $(M^{-1}Ap_j, p_j)_M = (Ap_j, p_j)$. the M inner product does not have to be computed explicitly

Left Preconditioning PCG Algorithm

1. $r^{(0)} = b - Au^{(0)}$, $z^{(0)} = M^{-1}r^{(0)}$, $p_0 = z^{(0)}$
2. For $j = 0, 1, \dots$, until convergence, do:
 3. $\alpha_j = (r^{(j)}, z^{(j)}) / (Ap_j, p_j)$
 4. $u^{(j+1)} = u^{(j)} + \alpha_j p_j$
 5. $r^{(j+1)} = r^{(j)} - \alpha_j Ap_j$
 6. $z^{(j+1)} = M^{-1}r^{(j+1)}$ (preconditioning)
 7. $\beta_j = (r^{(j+1)}, z^{(j+1)}) / (r^{(j)}, z^{(j)})$
 8. $p_{j+1} = z^{(j+1)} + \beta_j p_j$
9. End do

If $M = LL^T$, the preconditioner can be implemented in split and nonsplit forms. In can be shown that the two implementations are equivalent. $M^{-1}A$ is also self-adjoint with respect to the A inner product

Left Preconditioned GMRES

1. $r^{(j)} = M^{-1}(b - Au^{(j)})$, $\beta = \|r^{(0)}\|_2$, $v_1 = r^{(0)}/\beta$
 2. For $j = 1, 2, \dots, m$, do:
 3. Compute $w = M^{-1}Av_j$ (preconditioning)
 4. For $i = 1, 2, \dots, j$, Do:
 5. $h_{ij} = (w, v_i)$, $w = w - h_{ij}v_i$
 7. End do
 8. $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
 9. End do
 10. $V_m = [v_1, \dots, v_m]$, $\bar{H}_m = \{h_{ij}\}_{1 \leq i \leq j+1, 1 \leq j \leq m}$
 11. Compute $y_m = \operatorname{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$
and $u^{(m)} = u^{(0)} + V_m y_m$
 12. If satisfied stop, else restart with $u^{(0)} = u^{(m)}$
- $\operatorname{span}\{r^{(0)}, M^{-1}Ar^{(0)}, \dots, (M^{-1}A)^{m-1}r^{(0)}\}$

Right Preconditioned PCG

AM^{-1} is SPD with respect to M^{-1} inner product, PCG with right preconditioning is

1. $\alpha_j = (r^{(j)}, r^{(j)})_{M^{-1}} / (AM^{-1}p_j, p_j)_{M^{-1}}$
 2. $x^{(j+1)} = x^{(j)} + \alpha_j p_j$
 3. $r^{(j+1)} = r^{(j)} - \alpha_j AM^{-1}p_j$
 4. $\beta_j = (r^{(j+1)}, r^{(j+1)})_{M^{-1}} / (r^{(j)}, r^{(j)})_{M^{-1}}$
 5. $p_{j+1} = r^{(j+1)} + \beta_j p_j$
- $u = M^{-1}x$, put $q_j = M^{-1}p_j$, $z^{(j)} = M^{-1}r^{(j)}$
1. $\alpha_j = (z^{(j)}, r^{(j)}) / (Aq_j, q_j)$
 2. $u^{(j+1)} = u^{(j)} + \alpha_j q_j$
 3. $r^{(j+1)} = r^{(j)} - \alpha_j Aq_j$, $z^{(j+1)} = M^{-1}r^{(j+1)}$
 4. $\beta_j = (z^{(j+1)}, r^{(j+1)}) / (z^{(j)}, r^{(j)})$
 5. $q_{j+1} = z^{(j+1)} + \beta_j q_j$

Left Preconditioned GMRES (II)

All residual vectors and their norms computed correspond to preconditioned residual $z^{(m)} = M^{-1}(b - Au^{(m)})$, not to the original residual $r^{(m)} = b - Au^{(m)}$. The true residual can be obtained by applying M to $z^{(m)}$. This is inconvenient for testing stopping criterion.

If A is almost symmetric positive definite, an SPD preconditioner M can be constructed and a version of split preconditioned GMRES or a preconditioned GMRES with M inner product may be developed

The spectra of the left, right, and split preconditioned matrices are the same

Right Preconditioned GMRES

$$AM^{-1}x = b, \quad u = M^{-1}x$$

The new variable x never needs to be used explicitly. The preconditioned and original residuals are the same

$$r^{(m)} = b - Au^{(m)} = b - AM^{-1}x^{(m)}$$

If the preconditioned variable is

$$x^{(m)} = x^{(0)} + \sum_{i=1}^m v_i \eta_i$$

the corresponding original variable

$$u^{(m)} = u^{(0)} + M^{-1} \left[\sum_{i=1}^m v_i \eta_i \right]$$

The essential difference between left and right preconditioned GMRES is the direct availability of residual and residual norms

Parallel Implementation of PCG

PCG has five types of main operations

1. preconditioner setup
2. matrix vector multiply
3. vector updates
4. inner product
5. preconditioning operations

Potential problems with parallel PCG lie in the preconditioner setup and the preconditioning operation steps. Traditional preconditioners are based on incomplete LU factorizations and are inherently sequential

Right Preconditioned GMRES

1. $r^{(j)} = M^{-1}(b - Au^{(0)}), \beta = \|r^{(0)}\|_2, v_1 = r^{(0)}/\beta$
2. For $j = 1, 2, \dots, m$, do:
3. Compute $w = AM^{-1}v_j$
4. For $i = 1, 2, \dots, j$, Do:
5. $h_{ij} = (w, v_i), w = w - h_{ij}v_i$
6. End do
7. $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
8. End do
9. $V_m = [v_1, \dots, v_m], \tilde{H}_m = \{h_{ij}\}_{1 \leq i \leq j+1, 1 \leq j \leq m}$
10. Compute $y_m = \operatorname{argmin}_y \|\beta e_1 - \tilde{H}_m y\|_2$,
and $u^{(m)} = u^{(0)} + M^{-1}V_m y_m$
11. If satisfied stop, else restart with $u^{(0)} = u^{(m)}$

$$\operatorname{span}\{r^{(0)}, AM^{-1}r^{(0)}, \dots, (AM^{-1})^{m-1}r^{(0)}\}$$

Reverse Communication

It is advantageous to separate the matrix operations from the Krylov subspace accelerator subroutine, so that the Krylov subspace accelerators can be independent of particular data structure

When a matrix by vector product or a preconditioning operation is needed, the subroutine is exited from the Krylov subspace accelerator and the calling program unit perform the desired operations

Such a mechanism is called *reverse communication* and it can enhance the flexibility of the Krylov subspace accelerators substantially

Reverse Communication (sample)

```

icode = 0
1 continue
  call gmres(n, im, rhs, sol, i, vv, w, wk1,
*          wk2, eps, maxits, iout, icode)
  if ( icode .eq. 1 ) then
    call precon(n, wk1, wk2, lu, ju, iu)
    go to 1
  else if ( icode .eq. 2 ) then
    call matvec(n, wk1, wk2, a, ja, ia)
    go to 1
  end if

```

Level Scheduling (5-point Matrices)

By studying data dependence of graph of the matrix, it is possible to find grid points that are mutually independent at a given step. In 5-point discretizations, grid points along the same diagonals are independent. These diagonals are scheduled level by level. The forward sweep can be performed according to these levels or wavefronts. Grid points on the same level can be processed in parallel. The maximum degree of parallelism is the minimum of the total number of grid points along different coordinate directions. The first and the last few levels have few grid points and this may result in load imbalance

Parallel Forward Sweep

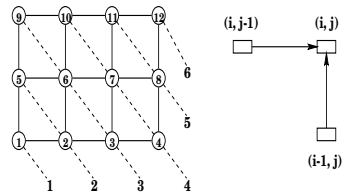
Standard preconditioners are usually constructed in the form of incomplete LU factorization. At each iteration, solutions with $Lx = b$ and $Uu = x$ are required. For compressed sparse row (CSR) format and forward sweep with L

```

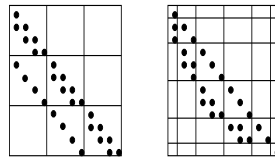
1. do i = 2, n
2.   do j = ia(i), ia(i+1)-1
3.     x(i) = x(i) - a(j)*x(ja(j))
4.   end do
5. end do

```

The vector b is overwritten. Outer loop with i is sequential, inner loop with j is parallelizable, may not have enough computations



Level scheduling for a 4 by 3 grid problem



Lower triangular matrix with level scheduling

Level Scheduling (Irregular Graph)

1. first find nodes that are not dependent on any other nodes
2. in the i th step, all nodes j that $a(i, j) \neq 0$ must be known, i.e., the nodes j are adjacent to node i must be known
3. the nodes can be marked by *levels* as the time they are visited
4. nodes with the same level are grouped together. They can be eliminated in parallel
5. permutation can be done explicitly or implicitly

Forward Sweep with Level Scheduling

```
1. do lev = 1, nlev
2.   j1 = level(lev)
3.   j2 = level(lev+1) + 1
4.   do k = j1, j2
5.     i = q(k)
6.     do j = ia(i), ia(i+1)-1
7.       x(i) = x(i) - a(j)*x(ja(j))
8.     end do
9.   end do
10. end do
```

Note that the double data structures. The first set of data is (level,q) of level scheduling, the second is the original matrix (a,ja,ia)