The Landscape of Sparse Ax=b Solvers

	Direct $A = LU$	Iterative $y_t = Ay_{t-1}$
Non-symmetric	**Pivoting LU**	**GMRES, BiCGSTAB, …**
Symmetric positive definite	**Cholesky**	**Conjugate gradient**

More General ↕ **More Robust**

More Robust ↔ **Less Storage**

D

Conjugate gradient iteration for Ax = b

$x_0 = 0$ approx solution

$r_0 = b$ residual = b - Ax

$d_0 = r_0$ search direction

<u>for</u> $t = 1, 2, 3, \ldots$

$x_t = x_{t-1} + \ldots$ new approx solution

$r_t = \ldots$ new residual

$d_t = \ldots$ new search direction

Conjugate gradient iteration for Ax = b

$x_0 = 0$ approx solution

$r_0 = b$ residual = b - Ax

$d_0 = r_0$ search direction

for $t = 1, 2, 3, \ldots$

$\alpha_t = \ldots$ step length

$x_t = x_{t-1} + \alpha_t d_{t-1}$ new approx solution

$r_t = \ldots$ new residual

$d_t = \ldots$ new search direction

Conjugate gradient iteration for Ax = b

$x_0 = 0$ approx solution

$r_0 = b$ residual = b - Ax

$d_0 = r_0$ search direction

<u>**for**</u> $t = 1, 2, 3, \ldots$

$\alpha_t = (r^T_{t-1} r_{t-1}) / (d^T_{t-1} A d_{t-1})$ step length

$x_t = x_{t-1} + \alpha_t \, d_{t-1}$ new approx solution

$r_t = \ldots$ new residual

$d_t = \ldots$ new search direction

Conjugate gradient iteration for Ax = b

$x_0 = 0$ approx solution

$r_0 = b$ residual = b - Ax

$d_0 = r_0$ search direction

for $t = 1, 2, 3, \ldots$

$\alpha_t = (r^T_{t-1} r_{t-1}) / (d^T_{t-1} A d_{t-1})$ step length

$x_t = x_{t-1} + \alpha_t \, d_{t-1}$ new approx solution

$r_t = \ldots$ new residual

$\beta_t = (r^T_t r_t) / (r^T_{t-1} r_{t-1})$

$d_t = r_t + \beta_t \, d_{t-1}$ new search direction

Conjugate gradient iteration for Ax = b

$x_0 = 0$ approx solution

$r_0 = b$ residual = b - Ax

$d_0 = r_0$ search direction

for $t = 1, 2, 3, \ldots$

$\alpha_t = (r^T_{t-1} r_{t-1}) / (d^T_{t-1} A d_{t-1})$ step length

$x_t = x_{t-1} + \alpha_t d_{t-1}$ new approx solution

$r_t = r_{t-1} - \alpha_t A d_{t-1}$ new residual

$\beta_t = (r^T_t r_t) / (r^T_{t-1} r_{t-1})$

$d_t = r_t + \beta_t d_{t-1}$ new search direction

Conjugate gradient iteration

$x_0 = 0, \quad r_0 = b, \quad d_0 = r_0$

for $t = 1, 2, 3, \ldots$

$\alpha_t = (r^T_{t-1} r_{t-1}) / (d^T_{t-1} A d_{t-1})$	step length
$x_t = x_{t-1} + \alpha_t d_{t-1}$	approx solution
$r_t = r_{t-1} - \alpha_t A d_{t-1}$	residual
$\beta_t = (r^T_t r_t) / (r^T_{t-1} r_{t-1})$	improvement
$d_t = r_t + \beta_t d_{t-1}$	search direction

- One matrix-vector multiplication per iteration
- Two vector dot products per iteration
- Four n-vectors of working storage

Conjugate gradient: Orthogonal sequences

- Krylov subspace: $K_t(A, b) = \text{span}(b, Ab, A^2b, \ldots, A^{t-1}b)$
- Conjugate gradient algorithm:

  $\underline{\text{for}}\ t = 1, 2, 3, \ldots$

  $\quad\quad \text{find } x_t \in K_t(A, b)$

  $\quad\quad \text{such that } r_t = (b - Ax_t) \perp K_t(A, b)$

- Notice $r_t \in K_{t+1}(A, b)$, so $r_t \perp r_k$ for all $k < t$
- Similarly, the "directions" are A-orthogonal:

  $$(x_t - x_{t-1})^T \cdot A \cdot (x_k - x_{k-1}) = 0$$

- The magic: Short recurrences. . .

  A is symmetric => can get next residual and direction from the previous one, without saving them all.

Conjugate gradient: Convergence

- In exact arithmetic, CG converges in n steps
  (completely unrealistic!!)
- Accuracy after t steps of CG is related to:
  - consider polynomials of degree t that are equal to 1 at 0.
  - how small can such a polynomial be at all the eigenvalues of A?
- Thus, eigenvalues close together are good.
- Condition number: $\kappa(A) = \|A\|_2 \, \|A^{-1}\|_2 = \lambda_{max}(A) \,/\, \lambda_{min}(A)$
- Residual is reduced by a constant factor by
  $O(\varkappa^{1/2}(A))$ iterations of CG.

Preconditioners

- Suppose you had a matrix B such that:
  1. condition number $\kappa(B^{-1}A)$ is small
  2. $By = z$ is easy to solve
- Then you could solve $(B^{-1}A)x = B^{-1}b$ instead of $Ax = b$
  - Actually $(B^{-1/2}AB^{-1/2})(B^{1/2}x) = B^{-1/2}b$, but never mind…
- $B = A$ is great for (1), not for (2)
- $B = I$ is great for (2), not for (1)
- Domain-specific approximations sometimes work
- B = diagonal of A sometimes works
- Better: blend in some direct-methods ideas. . .

Preconditioned conjugate gradient iteration

$x_0 = 0, \quad r_0 = b, \quad d_0 = B^{-1} r_0, \quad y_0 = B^{-1} r_0$

for $t = 1, 2, 3, \ldots$

$\alpha_t = (y^T_{t-1} r_{t-1}) / (d^T_{t-1} A d_{t-1})$ step length

$x_t = x_{t-1} + \alpha_t d_{t-1}$ approx solution

$r_t = r_{t-1} - \alpha_t A d_{t-1}$ residual

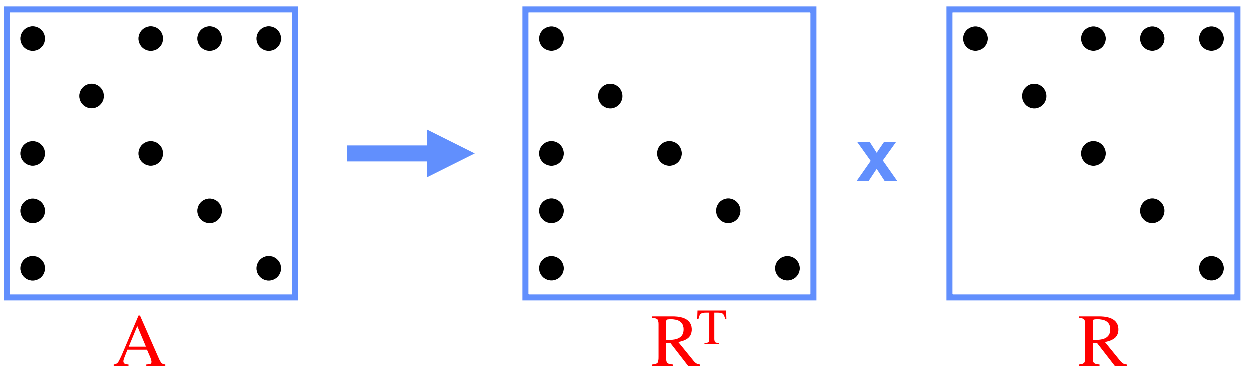
$y_t = B^{-1} r_t$ preconditioning solve

$\beta_t = (y^T_t r_t) / (y^T_{t-1} r_{t-1})$ improvement

$d_t = y_t + \beta_t d_{t-1}$ search direction

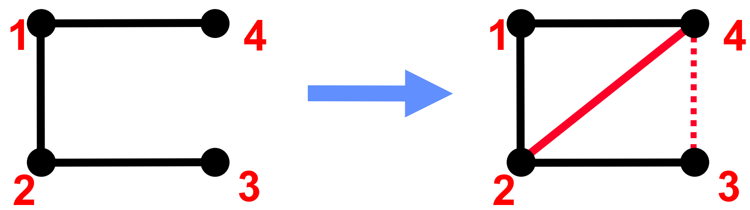
- One matrix-vector multiplication per iteration
- One solve with preconditioner per iteration

Incomplete Cholesky factorization (IC, ILU)



- Compute factors of A by Gaussian elimination, but ignore fill
- Preconditioner B = $R^TR \approx A$, not formed explicitly
- Compute $B^{-1}z$ by triangular solves (in time nnz(A))
- Total storage is O(nnz(A)), static data structure
- Either symmetric (IC) or nonsymmetric (ILU)

Incomplete Cholesky and ILU: Variants



- Allow one or more “levels of fill”
  - unpredictable storage requirements
- Allow fill whose magnitude exceeds a “drop tolerance”
  - may get better approximate factors than levels of fill
  - unpredictable storage requirements
  - choice of tolerance is ad hoc
- Partial pivoting (for nonsymmetric A)
- “Modified ILU” (MIC): Add dropped fill to diagonal of U or R
  - A and R^TR have same row sums
  - good in some PDE contexts
  - preserves null space for Laplacian A

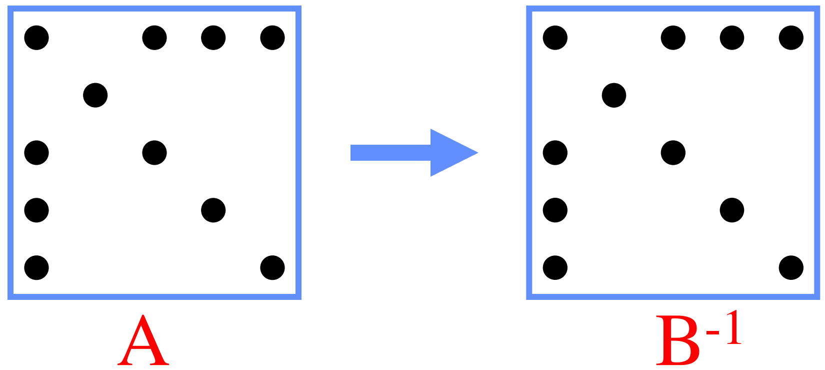
Incomplete Cholesky and ILU: Issues

- Choice of parameters
  - good: smooth transition from iterative to direct methods
  - bad: very ad hoc, problem-dependent
  - tradeoff: time per iteration (more fill => more time)
    vs # of iterations (more fill => fewer iters)

- Effectiveness
  - condition number usually improves (only) by constant factor
    - except for some PDE problems, and
    [new!] with carefully randomized dropping for Laplacians
  - still, often good when tuned for a particular class of problems

- Parallelism
  - triangular solves are not very parallel
  - reordering for parallel triangular solve by graph coloring

Sparse approximate inverses



- Compute $B^{-1} \approx A$ explicitly
- Minimize $\| A B^{-1} - I \|_F$ (in parallel, by columns)
- Variants: factored form of B^{-1}, more fill, . .
- Good: very parallel, seldom breaks down
- Bad: effectiveness varies widely

The Landscape of Sparse Ax=b Solvers

	Direct $A = LU$	Iterative $y_t = Ay_{t-1}$
Non-symmetric	**Pivoting LU**	**GMRES, BiCGSTAB, …**
Symmetric positive definite	**Cholesky**	**Conjugate gradient**

More General ↕ **More Robust**

More Robust ↔ **Less Storage**

D

Hierarchy of matrix classes (all real)

- General nonsymmetric
- Diagonalizable
- Normal
- Symmetric indefinite
- Symmetric positive (semi)definite = Factor width n
  - Factor width 2 < k < n
- Diagonally dominant SPSD = Factor width 2
- Generalized Laplacian = Symm diag dominant M-matrix
- Graph Laplacian

Other Krylov subspace methods

- Nonsymmetric linear systems:
  - GMRES:

    <u>for</u> $t = 1, 2, 3, \ldots$

    find $x_t \in K_t(A, b)$ such that $r_t = (Ax_t - b) \perp K_t(A, b)$

    But, no short recurrence => save old vectors => lots more space

    (Usually "restarted" every k iterations to use less space.)

  - BiCGStab, QMR, etc.:

    Two spaces $K_t(A, b)$ and $K_t(A^T, b)$ w/ mutually orthogonal bases

    Short recurrences => $O(n)$ space, but less robust

  - Convergence and preconditioning more delicate than CG
  - Active area of current research

- Eigenvalues: Lanczos (symmetric), Arnoldi (nonsymmetric)