Symmetric Positive Definite (SPD) Matrix

- An SPD matrix acts sort of like a positive number.
- A is *symmetric* if $a_{ij} = a_{ji}$, for all i and j.
- Several equivalent conditions for A to be *positive definite*:
  - All eigenvalues are > 0
  - LU factorization without pivoting succeeds, and all pivots are > 0
  - For every nonzero vector x , the number $x^T A x > 0$
- SPD matrices come up a lot in scientific computing & data analysis!
- The temperature matrix is SPD.

Conjugate gradient iteration

- An *iterative* algorithm to solve $Ax = b$.
- Start with a guess $x^{(0)}$, then compute $x^{(1)}$, $x^{(2)}$, ...
- Stop when you think you're close enough.
- In theory, CG can be used to solve *any* system $Ax = b$, provided only that A is SPD.
- In practice, how well CG works depends on specifics of A in subtle ways, involving eigenvalues and condition number.

Conjugate gradient iteration for Ax = b

$x^{(0)} = 0$ approximate solution

$r^{(0)} = b$ residual = b - Ax

$d^{(0)} = r^{(0)}$ search direction

for $k = 1, 2, 3, \ldots$:

$x^{(k)} = x^{(k-1)} + \ldots$ new approx solution

$r^{(k)} = \ldots$ new residual

$d^{(k)} = \ldots$ new search direction

Conjugate gradient iteration for Ax = b

$x^{(0)} = 0$ approximate solution

$r^{(0)} = b$ residual = b - Ax

$d^{(0)} = r^{(0)}$ search direction

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

$\alpha^{(k)} = \ldots$ step length

$x^{(k)} = x^{(k-1)} + \alpha^{(k)} d^{(k-1)}$ new approx solution

$r^{(k)} = \ldots$ new residual

$d^{(k)} = \ldots$ new search direction

Conjugate gradient iteration for Ax = b

$x^{(0)} = 0$ approximate solution

$r^{(0)} = b$ residual = b - Ax

$d^{(0)} = r^{(0)}$ search direction

for $k = 1, 2, 3, \ldots$:

$\alpha^{(k)} = (r^{(k-1)\mathbf{T}} r^{(k-1)}) / (d^{(k-1)\mathbf{T}} A \, d^{(k-1)})$ step length

$x^{(k)} = x^{(k-1)} + \alpha^{(k)} d^{(k-1)}$ new approx solution

$r^{(k)} = \ldots$ new residual

$d^{(k)} = \ldots$ new search direction

Conjugate gradient iteration for Ax = b

$x^{(0)} = 0$ approximate solution

$r^{(0)} = b$ residual = b - Ax

$d^{(0)} = r^{(0)}$ search direction

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

$\alpha^{(k)} = (r^{(k-1)T} r^{(k-1)}) / (d^{(k-1)T} A\, d^{(k-1)})$ step length

$x^{(k)} = x^{(k-1)} + \alpha^{(k)} d^{(k-1)}$ new approx solution

$r^{(k)} = \ldots$ new residual

$\beta^{(k)} = (r^{(k)T} r^{(k)}) / (r^{(k-1)T} r^{(k-1)})$

$d^{(k)} = r^{(k)} + \beta^{(k)} d^{(k-1)}$ new search direction

Conjugate gradient iteration for Ax = b

$x^{(0)} = 0$ approximate solution

$r^{(0)} = b$ residual = b - Ax

$d^{(0)} = r^{(0)}$ search direction

for $k = 1, 2, 3, \ldots :$

$\alpha^{(k)} = (r^{(k-1)T} r^{(k-1)}) / (d^{(k-1)T} A\, d^{(k-1)})$ step length

$x^{(k)} = x^{(k-1)} + \alpha^{(k)} d^{(k-1)}$ new approx solution

$r^{(k)} = r^{(k-1)} - \alpha^{(k)} A d^{(k-1)}$ new residual

$\beta^{(k)} = (r^{(k)T} r^{(k)}) / (r^{(k-1)T} r^{(k-1)})$

$d^{(k)} = r^{(k)} + \beta^{(k)} d^{(k-1)}$ new search direction

Conjugate gradient iteration to solve A*x=b

$x^{(0)} = 0,\ r^{(0)} = b,\ d^{(0)} = r^{(0)}$ (these are all vectors)

for $k = 1, 2, 3, \ldots :$

$\alpha^{(k)} = (r^{(k-1)T} r^{(k-1)}) / (d^{(k-1)T} A\, d^{(k-1)})$ step length

$x^{(k)} = x^{(k-1)} + \alpha^{(k)} d^{(k-1)}$ approximate solution

$r^{(k)} = r^{(k-1)} - \alpha^{(k)} A\, d^{(k-1)}$ residual $= b - A\, x^{(k)}$

$\beta^{(k)} = (r^{(k)T} r^{(k)}) / (r^{(k-1)T} r^{(k-1)})$ improvement

$d^{(k)} = r^{(k)} + \beta^{(k)} d^{(k-1)}$ search direction

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

Vector and matrix primitives for CG

- DAXPY: $v = \alpha * v + \beta * w$ (vectors v, w; scalars α, β)
  - Time = O(n)
- DDOT: $\alpha = v^T * w = \sum_j v[j] * w[j]$ (vectors v, w; scalar α)
  - Time = O(n)
- Matvec: $v = A * w$ (matrix A, vectors v, w)
  - This is the hard part!
  - Time = $O(n^2)$ if A is a full matrix stored as a 2-D array
  - But all you need is a subroutine to compute v from w
  - If A is *sparse*, time = O(#nonzeros in A)

The Landscape of Ax=b Solvers

	Direct $A = LU$	Iterative $y' = Ay$
Non-symmetric	**Pivoting LU**	**GMRES, BiCGSTAB, …**
Symmetric positive definite	**Cholesky**	**Conjugate gradient**

More General

More Robust

More Robust ⟷ **Less Storage (if sparse)**

Optional:

Analysis of the Conjugate Gradient Algorithm

See Shewchuk's paper (linked to course web site) for details.

Conjugate gradient: Krylov subspaces

- Eigenvalues: $Av = \lambda v \qquad \{ \lambda_1, \lambda_2, \ldots, \lambda_n \}$
- Cayley-Hamilton theorem:

$$(A - \lambda_1 I)\cdot(A - \lambda_2 I) \cdots (A - \lambda_n I) = 0$$

Therefore $\sum_{0 \le i \le n} c_i A^i = 0$ for some c_i

so $$A^{-1} = \sum_{1 \le i \le n} (-c_i/c_0)\, A^{i-1}$$

- Krylov subspace:

Therefore if $Ax = b$, then $x = A^{-1} b$ and

$$x \in \text{span}(b, Ab, A^2b, \ldots, A^{n-1}b) = K_n(A, b)$$

Conjugate gradient: Orthogonal sequences

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

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

  find $x^{(i)} \in K_i(A, b)$

  such that $r^{(i)} = (b - Ax^{(i)}) \perp K_i(A, b)$

- Notice $r^{(i)} \in K_{i+1}(A, b)$, so $r^{(i)} \perp r^{(j)}$ for all $j < i$
- Similarly, the "directions" are A-orthogonal:

  $$(x^{(i)} - x^{(i-1)})^T \cdot A \cdot (x^{(j)} - x^{(j-1)}) = 0$$

- 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 k steps of CG is related to:
  - consider polynomials of degree k 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
  $O(\kappa^{1/2}(A))$ iterations of CG.

Other Krylov subspace methods

- Nonsymmetric linear systems:
  - GMRES:
    for $i = 1, 2, 3, \ldots$
    find $x^{(i)} \in K_i(A, b)$ such that $r^{(i)} = (Ax^{(i)} - b) \perp K_i(A, b)$
    But, no short recurrence => save old vectors => lots more space
    (Usually "restarted" every k iterations to use less space.)
  - BiCGStab, QMR, etc.:
    Two spaces $K_i(A, b)$ and $K_i(A^T, b)$ 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)