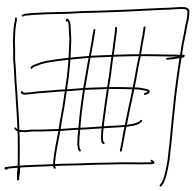
Preconditioning

Wednesday, December 04, 2013
11:28

General Krylov Picture

- Generate Krlov Subspace from A, b
- Extract "optimal" approximation
  - Galerkin
  - energy min
  - residual min

In the mesh problem



if b has one non-zero

$K_k(A,b) \subseteq$ { vectors w/ non-zeros at most distance k from non-zero in b

Favorites

- CG if symmetric, positive definite
- GMRES if non-symmetric

Preconditioning: Convert $Ax=b$ to $M^{-1}Ax = M^{-1}b$

where M^{-1} is "close to" A^{-1}

Really M^{-1} tries to capture physics that would slow us down.

Preconditioning Strategies

- Use splitting ideas OR approximate direct factorizations (this works OK)
- Use "physics" of problem
  - e.g. different discretizations of PDE (multigrid)
  - approx to a PDE (use an FFT preconditioner)
  - simplify physics (fine mesh → coarse mesh, for example)

Pro: Fast solves possible. Con: Huge amount of work

Preconditioners based on splittings

- the simplest is Jacobi $M = \text{diag}(A)$

  pro: easy con: really not that useful

  Note: Model problem: $(T_{N\times N}) \Rightarrow D = 4I$

  $\Rightarrow K_k(D^{-1}T_{N\times N}, D^{-1}b) = K_k(T_{N\times N}, b)$

  Hey look Jacobi is useless

  Burnt... block Jacobi is useful because I take the block diagonal of A

- Solve D_{jj} systems exactly
- approx solution of D_{jj} systems

Blocking also eases parallelization.
→ Petse uses block Jacobi to
do local + processor preconditioning
to reduce synchronization traffic

- one could also do Gauss-Seidel or block Gauss-Seidel
  - but we lose some opportunity for parallelism in the precondition
  - we also must symmetrize to use CG (which we want to do fairly often just because $A = A^T \not\Rightarrow M^{-1}A = (M^{-1}A)^T$)

Preconditioning for CG

$$M^{-1}Ax = M^{-1}b \quad \text{vs} \quad \left(M^{-1/2} A M^{-1/2}\right)\left(M^{1/2} x\right) = M^{-1/2} b$$
$$\left(R^{-T} A R^{-1}\right)(Rx) = R^{-T} b$$
$$M = RR^T$$

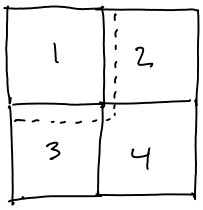
Another approach - S.P.D. M, no factorization
Think CG = Lanczos + Tridiagonal Solve (mixed up)
Lanczos = interlaced mult by A & orth. w.r.t. inner product
Std? M? M^{-1}?

We really need M to be SPD to use CG. You need to sanity check M yourself, the CG routine won't save you from yourself

For Jacobi / block Jacobi — no worries.
For Gauss-Seidel or SOR - symmetrize w/ forward then backward sweep

Other approaches:
- Sparse approximate factorization (many variants)
  - you can pre-specify where non-zeros go
  - can specify drop tolerance
- Sparse approximate inverse methods

Schwartz method ideas



4 processors have their data

what if processor one looks at a narrow band in the other domains?

What you get from this is a little bit of information transfer, but this accelerates convergence dramatically

- Additive Schwartz - Block Jacobi + Schwartz
- Multiplicative Schwartz: Gauss Seidel + Schwartz

Really what's going on is you shuffle error around into the internals of the other processor, and you keep smoothing it out ⇒ whack-a-mole style

but the error is more each CPU's local domain

⇒ easier to "whack" or "smooth out"