Krylov Subspace Methods

Overview : Lab #1

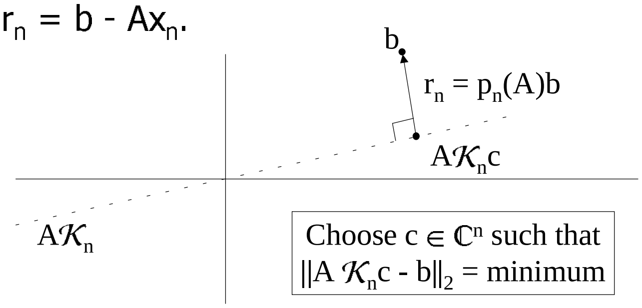
- Krylov Subspace Methods
  - GMRES and BICGSTAB
- Preconditioners
  - Diagonal and SPAI
- Sparse Matrix Storage
- Overview of Lab Software
  - ISIS++
- “Hands-On” Experiments

Krylov Subspace Methods

- Let $\mathcal{K}_n = \langle b, Ab, \dots, A^{n-1}b \rangle$, the n^{th} Krylov subspace
- Let $A \in \mathbb{C}^{m \times m}$,
- Let x^* be the solution to $Ax=b$ (so, $x^*=A^{-1}b$)
- At each step n, we would like to find the "best" $x_n \in \mathcal{K}_n$.

GMRES : Basic Idea

- At each step n, approximate x^* by $x_n \in \mathcal{K}_n$ that minimizes the norm of the residual $r_n = b - Ax_n$.



GMRES : Basic Idea

- GMRES uses an n+1 term recurrence.
- Storage (memory) costs and computational costs increase with n.

- Alternative: Restarted GMRES : GMRES(k)
  - After k steps, restart the GMRES iteration anew with x_n as the initial guess

GMRES : Convergence #1

- (Nonrestarted) GMRES converges monotonically: $|r_{n+1}| \leq |r_n|$.

- Why? $|r_n|$ is as small as possible for $\mathcal{K}_n$. By enlarging $\mathcal{K}_n$ to $\mathcal{K}_{n+1}$, we decrease $|r_n|$ or at worst leave it unchanged.

GMRES : Convergence #2

- GMRES must converge after m steps:
  $|r_m|=0$
- We hope that it converges in n << m steps!
- Restarted GMRES may not converge.

GMRES : Polynomial Approx.

- $|r_n|=|p_n(A)b| \leq |p_n(A)||b| =$ minimal
- The convergence rate is determined by

$$\frac{\|r_n\|}{\|b\|} \leq \inf_{p_n \in P_n} \|p_n(A)\|$$

- So how small can $|p_n(A)|$ be?

GMRES : Polynomial Approx.

- Suppose A is diagonalizable. Then,

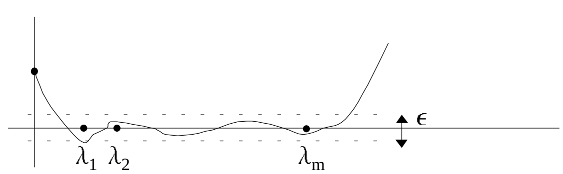
  $$\|p(A)\| \le \|V\| \|p(\Lambda)\| \|V^{-1}\| = \kappa(V) \|p\|_{\Lambda(A)}$$

- So, at step n, the residual r_n satisifies

  $$\frac{\|r_n\|}{\|b\|} \le \inf_{p_n \in P_n} \|p_n(A)\| \le \kappa(V) \inf_i \|p(\lambda_i)\|$$

GMRES : Polynomial Approx.

- If A is not too far from normal (in the sense that $\kappa(V)$ is not too large) and the size of p_n on the spectrum of $\Lambda(A)$ decreases quickly with n, then GMRES converges quickly.



BICGSTAB: Basic Idea

- At each step n, approximate x^* by $x_n \in \mathcal{K}_n$ where $r_n = \varphi_n(A)\psi_n(A)r_0$
- $\varphi_n(A)$ is the residual polynomial for BICG
- $\psi_{n+1}(A) = (1-\omega_n t)\psi_n$ is designed to stabilize the convegence behavior of BiCG.

BICGSTAB: Basic Idea

- The residual norm $|r_n|_2$ is not minimal.
- BICGSTAB is not optimal in that it does not minimize the number of iterations.
- BICGSTAB uses a three-term recurrence (rather than the (n+1)-term recurrance of GMRES) and so requires less storage.
- BICGSTAB is "transpose-free", unlike BICG.

Preconditioners : General

- Want to solve $n \times n$ nonsingular system $Ax=b$
- Instead, we solve $M^{-1}Ax=M^{-1}b$, where M has been chosen to speed the iterative solution process.
- NOTE: We do not form M^{-1} explicitly in general, but instead solve $My=c$.

Preconditioners : General

- "Ideal" Preconditioner : $M^{-1} = A^{-1}$.
- What makes a "good" preconditioner?
  - 1) M easy to compute
  - 2) $My=c$ easy to solve
  - 3) $M^{-1}A$ approximates I
  - 4) Eigenvalues of $M^{-1}A$ are clustered

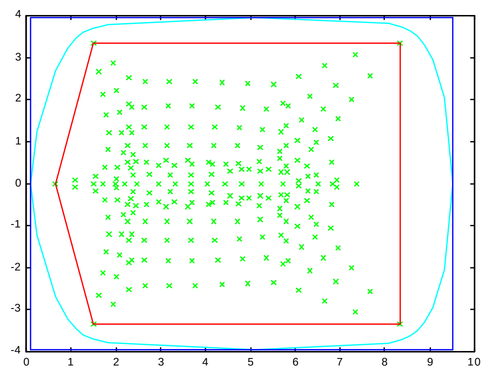
Preconditioners : Diagonal

- Take M = diag(A)   (be sure M^{-1} exists!)
- Also called "Jacobi" preconditioner
- Solving My=c has O(n) cost.

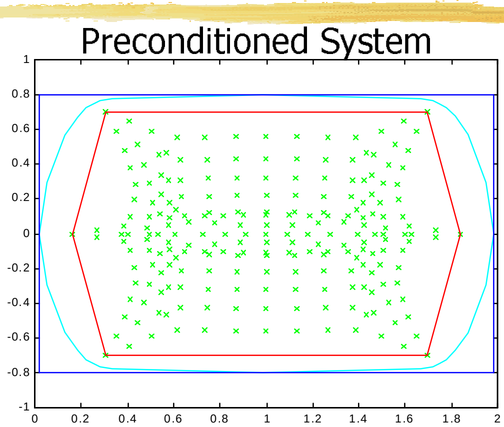
- If M is diagonally dominant, $M^{-1}A$ may be a good approximation to I.
- (Could also try an ILU-type preconditioner.)

Preconditioners : Diagonal

Unpreconditioned System



Preconditioners : Diagonal



Preconditioners : SPAI

- **SPAI** = **SP**arse **A**pproximate **I**nverse
- Significantly more expensive than ILU-type on one processor, but will parallelize
- Can succeed where ILU-types fails

Preconditioners : SPAI

- Approximate A^{-1} in the Frobenius norm:

$$\min_M \|AM - I\|_F^2 = \sum_{k=1}^{n} \min_{m_k} \|Am_k - e_k\|_2^2$$

where m_k is the k^{th} column of M.

- This produces n independent least-squares problems, which is the source of parallelism in SPAI.

Preconditioners : SPAI

- Assume we know the sparsity pattern of M.
- Let $\mathcal{J} = \{j \mid m_k(j) \neq 0\}$
- The set of rows of A that could affect MA is $\mathcal{I} = \{i \mid A(i,\mathcal{J}) \neq 0\}$.
- Construct the full matrix $A' = A(\mathcal{I},\mathcal{J})$ where A′ has $|\mathcal{I}|$ rows and $|\mathcal{J}|$ columns.

Preconditioners : SPAI

- Solve: $\min_{m'_k} \|A' m'_k - e'_k\|_2$
  where e'_k is $e_k(\mathcal{I})$.

- But how do we determine the sparsity pattern of M?

Preconditioners : SPAI

- Suppose we choose $\mathcal{J} = \{k\}$ (the diagonal) and compute m'_k as discussed above.
- Consider the residual: $r = A(.,\mathcal{J})m'_k - e_k$
- If $|r|_2 \neq 0$, m_k is not exactly $A^{-1}e_k$.

- We must augment $\mathcal{J}$ to reduce the components of the residual!

Preconditioners : SPAI

- Let $\mathcal{L} = \{l \mid r(l) \neq 0\}$.
- Let $\mathcal{J}' = \{j \mid A(\mathcal{L},j) \neq 0\} - \mathcal{J}$.

  These are candidate indices to add to $\mathcal{J}$.

- There may be many candidate indices, so we must carefully choose those that most effectively reduce $|r|_2$.

Preconditioners : SPAI

- Use the following hueristic:
- For each j $\in \mathcal{J}'$, solve the 1D problem:

$$\min_{\mu_j} \left\| r + \mu_j A e_j \right\|_2$$

- Which has the solution:

$$\mu_j = r^T A e_j \Big/ \left\| A e_j \right\|_2^2$$

- With residual:

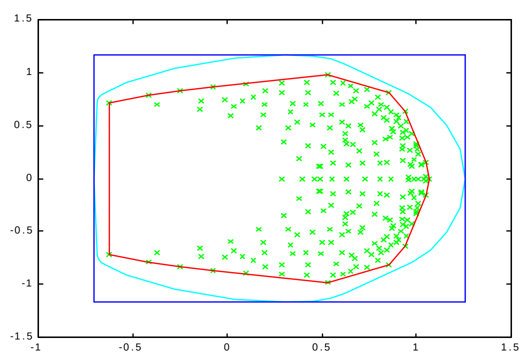
$$\rho_j = \left\| r \right\|_2^2 - \left(r^T A e_j \right)^2 \Big/ \left\| A e_j \right\|_2^2$$

Preconditioners : SPAI

1) Determine $\mathcal{J}'$
2) Determine ρ_j for all $j \in \mathcal{J}'$
3) Determine ν, the mean of $\{\rho_j\}$.
4) Retain all $j \in \mathcal{J}'$ with $\rho_j \le \nu$, up to some maximum number of indices (typically 5)

Preconditioners : SPAI

Preconditioned System



Sparse Storage Formats

- Store only the nonzero elements of a matrix

- Execute common matrix operations

Sparse Storage Formats

- Coordinate Format
  - 3 Array containing nonzero elements of A, row indices, and column indices

$$\begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix}$$

AA	12	9	7	5	1	2	11	3	6	4	8	10
JR	5	3	3	2	1	1	4	2	3	2	3	4
JC	5	5	3	4	1	4	4	1	1	2	4	3

Sparse Storage Formats

- Compressed Sparse Row (CSR)
  - 3 Array containing nonzero elements of A, column indices, and pointers to the beginning of each row

$$\begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix}$$

AA	1	2	3	4	5	6	7	8	9	10	11	12
JA	1	4	1	2	4	1	3	4	5	3	4	5
IA	1	3	6	10	12	13						

Sparse Storage Formats

- Compressed Sparse Column (CSC)
  - Like CSR, but store columns instead of rows

$$\begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix}$$

AA	1	3	6	4	7	10	2	5	8	11	9	12
JA	1	2	3	2	3	4	1	2	3	4	3	5
IA	1	4	5	7	11	13						

Sparse Storage Formats

- Harwell-Boeing Exchange Format
  - popular mechanism for text-file exchange of sparse matrix data
  - Contains header block describing matrix name, type, size, and number of associated vectors
  - File may contain right-hand-side vectors, vectors for starting guesses, and the solution vector

Overview of Lab Software

- We will use ISIS++, a portable, object-oriented framework for solving sparse systems of linear equations.
- ISIS++ is freely available from http://z.ca.sandia.gov/isis/
- We will also be using elements of Matlab, as well as SPARSKIT and other numerical software available through the WWW.

Overview of Lab Software

- We will test both preconditioned and nonpreconditioned GMRES and BICGSTAB on a variety of problem from various sources.
- Refer to the handout to get started.