Iterative Methods in Linear Algebra
(part 2)

Stan Tomov

Innovative Computing Laboratory
Computer Science Department
The University of Tennessee

Wednesday April 7, 2010
Topics

Projection in Scientific Computing

Sparse matrices, parallel implementations

PDEs, Numerical solution, Tools, etc.

Iterative Methods
Outline

■ Part I
   Krylov iterative solvers

■ Part II
   Convergence and preconditioning

■ Part III
   Iterative eigen-solvers
Part I

Krylov iterative solvers
Krylov iterative solvers

Building blocks for Krylov iterative solvers covered so far

- Projection/minimization in a subspace
  - Petrov-Galerkin conditions
  - Least squares minimization, etc.

- Orthogonalization
  - CGS and MGS
  - Cholesky or Householder based QR
We also covered abstract formulations for iterative solvers and eigen-solvers.

What are the goals of this lecture?

- Give specific examples of Krylov solvers
- Show how examples relate to the abstract formulation
- Show how examples relate to the building blocks covered so far, specifically to
  - Projection, and
  - Orthogonalization
- But we are not going into the details!
How are these techniques related to Krylov iterative solvers?

**Projection and iterative solvers**

- The problem: Solve $Ax = b$ in $\mathbb{R}^n$
- Iterative solution: at iteration $i$ extract an approximate $x_i$ from just a subspace $V = \text{span}\{v_1, ..., v_m\}$ of $\mathbb{R}^n$
- How? As on slide 22, impose constraints:
  $b - Ax \perp \text{subspace } W = \text{span}\{w_1, ..., w_m\}$ of $\mathbb{R}^n$, i.e.
  $V = \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\}$
- Conditions (*): known also as Petrov-Galerkin conditions. Projection is **orthogonal**: $V$ and $W$ are the same (Galerkin conditions) or **oblique**: $V$ and $W$ are different.

**Matrix representation**

- Let $V = \{v_1, ..., v_m\}, W = \{w_1, ..., w_m\}$
- Find $y \in \mathbb{R}^n$ s.t. $x = x_0 + V y$ solves $Ax = b$, i.e.
  $A V y = b - A x_0 = r_0$
  subject to the orthogonality constraints:
  $W^T A V y = W^T r_0$
- The choice for $V$ and $W$ is crucial and determines various methods (more in Lectures 4 and 5)

Remember projection slides 26 & 27, Lecture 7 (left)

- **Projection** in a subspace is the basis for an iterative method
- Here projection is in $V$
- In Krylov methods $V$ is the Krylov subspace:
  $K_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\}$
  where $r_0 \equiv b - Ax_0$ and $x_0$ is an initial guess.

- Often $V$ or $W$ are orthonormalized
  - The projection is 'easier' to find when we work with an orthonormal basis
    (e.g. problem 4 from homework 5: projection in general vs orthonormal basis)
  - The orthonormalization can be CGS, MGS, Cholesky or Householder based, etc.
To summarize, Krylov iterative methods in general

- expend the Krylov subspace by a matrix-vector product, and
- do a projection in it.

Various methods result by specific choices of the expansion and projection.
A specific example with the Conjugate Gradient Method (CG)
The method is for SPD matrices.

Both $V$ and $W$ are the Krylov subspaces, i.e. at iteration $i$

$$V \equiv W \equiv K_i(A, r_0) \equiv \text{span}\{r_0, Ar_0, \ldots, A^{i-1}r_0\}$$

The projection $x_i \in K_i(A, r_0)$ satisfies the Petrov-Galerkin conditions

$$(Ax_i, \phi) = (b, \phi), \text{ for } \forall \phi \in K_i(A, r_0)$$
At every iteration there is a way (to be shown later) to construct a new search direction $p_i$ such that

$$\text{span}\{p_0, p_1, \ldots, p_i\} \equiv K_{i+1}(A, r_0) \quad \text{and} \quad (Ap_i, p_j) = 0 \quad \text{for} \quad i \neq j.$$ 

**Note:** $A$ is SPD $\Rightarrow (Ap_i, p_j) \equiv (p_i, p_j)_A$ can be used as an inner product, i.e. $p_0, \ldots, p_i$ is an $(\cdot, \cdot)_A$ orthogonal basis for $K_{i+1}(A, r_0)$

$\Rightarrow$ we can easily find $x_{i+1} \approx x$ as

$$x_{i+1} = x_0 + \alpha_0 p_0 + \cdots + \alpha_i p_i \quad \text{s.t.}$$

$$(Ax_{i+1}, p_j) = (b, p_j) \quad \text{for} \quad j = 0, \ldots, i$$

Namely, because of the $(\cdot, \cdot)_A$ orthogonality of $p_0, \ldots, p_i$ at iteration $i + 1$ we have to find only $\alpha_i$

$$(Ax_{i+1}, p_j) = (A(x_i + \alpha_i p_i), p_i) = (b, p_i), \quad \Rightarrow \quad \alpha_i = \frac{(r_i, p_i)}{(Ap_i, p_i)}$$

**Note:** $x_i$ above actually can be replaced by any $x_0 + v, v \in K_i(A, r_0)$ (Why?)
Conjugate Gradient Method

1: Compute \( r_0 = b - Ax_0 \) for some initial guess \( x_0 \)
2: for \( i = 0 \) to ... do
3: \( \rho_i = r_i^T r_i \)
4: if \( i = 0 \) then
5: \( p_0 = r_0 \)
6: else
7: \( p_i = r_i + \frac{\rho_i}{\rho_{i-1}} p_{i-1} \)
8: end if
9: \( q_i = Ap_i \)
10: \( \alpha_i = \frac{\rho_i}{p_i^T q_i} \)
11: \( x_{i+1} = x_i + \alpha_i p_i \)
12: \( r_{i+1} = r_i - \alpha_i q_i \)
13: check convergence; continue if necessary
14: end for

Note:
- One matrix vector product/iteration (at line 9)
- Two inner-products/iteration (lines 3 and 10)
- In exact arithmetic \( r_{i+1} = b - Ax_{i+1} \) (Apply \( A \) to both sides of 11 and subtract from \( b \) to get line 12)
- Update for \( x_{i+1} \) is as pointed out before, i.e. with
  \[ \alpha_i = \frac{(r_i, r_i)}{(Ap_i, p_i)} = \frac{(r_i, p_i)}{(Ap_i, p_i)} \]
  since \( (r_i, p_{i-1}) = 0 \) (exercise)
- Other relations to be proved (exercise)
  - \( p_i \)'s span the Krylov space
  - \( p_i \)'s are \( (\cdot, \cdot)_A \) orthogonal, etc.
To sum it up:

- In exact arithmetic we get the exact solution in at most $n$ steps, i.e.
  \[ x = x_0 + \alpha_0 p_0 + \cdots + \alpha_i p_i + \alpha_{i+1} p_{i+1} + \cdots + \alpha_{n-1} p_{n-1} \]

- At every iteration one more term $\alpha_j p_j$ is added to the current approximation
  \[ x_i = x_0 + \alpha_0 p_0 + \cdots + \alpha_{i-1} p_{i-1} \]
  \[ x_{i+1} = x_0 + \alpha_0 p_0 + \cdots + \alpha_{i-1} p_{i-1} + \alpha_i p_i \equiv x_i + \alpha_i p_i \]

- Note: we do not have to solve linear system at every iteration because of the A-orthogonal basis that we manage to maintain and expend at every iteration

- It can be proved that the error $e_i = x - x_i$ satisfies
  \[ \|e_i\|_A \leq 2 \left( \frac{\sqrt{k(A)} - 1}{\sqrt{k(A)} + 1} \right)^i \|e_0\|_A \]
Building orthogonal basis for a Krylov subspace

We have seen the importance in

- Defining projections
  - not just for linear solvers
- Abstract linear solvers and eigen-solver formulations
- A specific example
  - in CG where the basis for the Krylov subspaces is A-orthogonal (A is SPD)

We have seen how to build it

- CGS, MGS, Cholesky or Householder based, etc.
- These techniques can be used in a method specifically designed for Krylov subspaces (general non-Hermitian matrix), namely in the Arnoldi’s Method
Arnoldi’s Method

Arnoldi’s method:

Build an orthogonal basis for $K_m(A, r_0)$

$A$ can be general, non-Hermitian

1: $v_1 = r_0$
2: for $j = 1$ to $m$ do
3: $h_{ij} = (Av_j, v_i)$ for $i = 1, \ldots, j$
4: $w_j = Av_j - h_{1j}v_1 - \ldots - h_{jj}v_j$
5: $h_{j+1,j} = ||w_j||_2$
6: if $h_{j+1,j} = 0$ Stop
7: $v_{j+1} = \frac{w_j}{h_{j+1,j}}$
8: end for

Note:

- This orthogonalization is based on CGS (line 4)

$$w_j = Av_j - (Av_j, v_1)v_1 - \ldots - (Av_j, v_j)v_j$$

- $\Rightarrow$ up to iteration $j$ vectors $v_1, \ldots, v_j$ are orthogonal

- The space of this orthogonal basis grows by taking the next vector to be $Av_j$

- If we do not exit at step 6 we will have

$$K_m(A, r_0) = span\{v_1, v_2, \ldots, v_m\}$$

(exercise)
Arnoldi’s method in matrix notation

- Denote

\[ V_m \equiv [v_1, \ldots, v_m], \quad H_{m+1} = \{h_{ij}\}_{m+1 \times m} \]

and by \( H_m \) the matrix \( H_{m+1} \) without the last row.

- Note that \( H_m \) is upper Hessenberg (0s below the lower second sub-diagonal) and

\[
AV_m = V_m H_m + w_m e_m^T \\
V_m^T AV_m = H_m
\]

(exercise)
Variations:
- Explained using CGS
- Can be implemented with MGS, Householder, etc.

How to use it in linear solvers?
- Example with the Full Orthogonalization Method (FOM)
1: $\beta = ||r_0||_2$
2: Compute $v_1, \ldots, v_m$ with Arnoldi
3: $y_m = \beta H_m^{-1} e_1$
4: $x_m = x_0 + V_m y_m$

FOM

- Look for solution in the form

$$x_m = x_0 + y_m(1) v_1 + \cdots + y_m(m) v_m$$

$$\equiv x_0 + V_m y_m$$

- Petrov-Galerkin conditions will be

$$V_m^T A x_m = V_m^T b$$

$$\Rightarrow V_m^T A(x_0 + V_m y_m) = V_m^T b$$

$$\Rightarrow V_m^T A V_m y_m = V_m^T r_0$$

$$\Rightarrow H_m y_m = V_m^T r_0 = \beta e_1$$

which is given by steps 3 and 4 of the algorithm
What happens when $m$ increases?

- computation grows as at least $O(m^2)n$
- memory is $O(mn)$

A remedy is to restart the algorithm, leading to restarted FOM

FOM($m$)

1: $\beta = \|r_0\|_2$
2: Compute $v_1, \ldots, v_m$ with Arnoldi
3: $y_m = \beta H_m^{-1} e_1$
4: $x_m = x_0 + V_m y_m$. Stop if residual is small enough.
5: Set $x_0 := x_m$ and go to 1


**Generalized Minimum Residual Method (GMRES)**

- Similar to FOM
  - Again look for solution
    
    $$x_m = x_0 + V_m y_m$$
    
    where $V_m$ is from the Arnoldi process (i.e. $K_m(A, r_0)$).

- The test conditions $W_m$ from the abstract formulation (slide 27, Lecture 7)
  
  $$W_m^T A V_m y_m = W_m^T r_0$$
  
  are $W_m = AV_m$.

- The difference results in step 3 from FOM, namely
  
  $$y_m = \beta H_{m}^{-1} e_1$$
  
  being replaced by
  
  $$y_m = \text{argmin}_y \| \beta e_1 - H_{m+1} y \|_2$$
Similarly to FOM, GMRES can be defined with

- Various orthogonalizations in the Arnoldi process
- Restart

Note:

- Solving the least squares (LS) problem

\[ \arg\min_y \|\beta e_1 - H_{m+1}y\|_2 \]

can be done with QR factorization as discussed in Lecture 7, Slide 25
Lanczos Algorithm

Can we improve on Arnoldi if $A$ is symmetric?

- Yes! $H_m$ becomes symmetric so it will be just 3 diagonal
- the simplification of Arnoldi in this case leads to the Lanczos Algorithm
- Lanczos can be used in deriving CG

The Lanczos Algorithm

1: $v_1 = \frac{r_0}{||r_0||_2}$, $\beta_1 = 0$, $v_0 = 0$
2: for $j = 1$ to $m$ do
3: $w_j = Av_j - \beta_j v_{j-1}$
4: $\alpha_j = (w_j, v_j)$
5: $w_j = w_j - \alpha_j v_j$
6: $\beta_{j+1} = ||w_j||_2$. If $\beta_{j+1} = 0$ then Stop
7: $v_{j+1} = \frac{w_j}{\beta_{j+1}}$
8: end for

- Matrix $H_m$ here is 3-diagonal with diagonal
  \[ h_{ii} = \alpha_i \]
  and off diagonal
  \[ h_{i,i+1} = \beta_{i+1} \]
- In exact arithmetic $v_i$s’ are orthogonal but in reality orthogonalization gets lost rapidly
We saw how different basis for the Krylov spaces is characteristic for various methods, e.g.

- GMRES uses orthogonal
- CG uses $A$-orthogonal

This is true for other methods as well

- Conjugate Residual (CR; for symmetric problems) uses $A^T A$-orthogonal (i.e. $Ap_i$'s are orthogonal)
- $A^T A$-orthogonal basis can be generalized to the non-symmetric case as well, e.g. in the Generalized Conjugate Residual (GCR)
Other Krylov methods

We considered various methods that construct a basis for the Krylov subspaces

Another big class of methods is based on biortogonalization (algorithm due to Lanczos)

- For non-symmetric matrices build a pair of bi-orthogonal bases for the two subspaces

\[
K_m(A, v_1) = \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}
\]
\[
K_m(A^T, w_1) = \text{span}\{w_1, A^T w_1, \ldots, (A^T)^{m-1}w_1\}
\]

- Examples here are BCG and QMR (not to be discussed)
- These methods are more difficult to analyze
Part II

Convergence and preconditioning
Convergence can be analyzed by

- Exploit the optimality properties (of projection) when such properties exist
- A useful tool is Chebyshev polynomials
- Depend on the condition number of the matrix, e.g.
  - In CG it is

\[
\|e_i\|_A \leq 2 \left( \frac{\sqrt{k(A)} - 1}{\sqrt{k(A)} + 1} \right)^i \|e_0\|_A
\]
Preconditioning

Convergence can be slow or even stagnate

- for ill-conditioned matrices (with large condition number)

But can be improved with **preconditioning**

\[ x_{i+1} = x_i + P(b - Ax_i) \]

- Think of \( P \) as a preconditioner, an operator/matrix \( P \approx A^{-1} \)
- for \( P = A^{-1} \) it takes 1 iteration
Preconditioning

Properties desired in a preconditioner:

- Should approximate $A^{-1}$
- Should be easy to compute, apply to a vector, and store

Iterative solvers can be extended to support preconditioning (How?)
Extending Iterative solvers to support preconditioning

- The same solver can be used but on a modified problem, e.g.
- Problem $Ax = b$ is transformed into

$$PAx = Pb$$

known as **left preconditioning**

- Problem $Ax = b$ is transformed into

$$APx = b, \quad x = Pu$$

known as **right preconditioning**

- Convergence of the modified problem would depend on $k(PA)$ (e.g. with left preconditioning)
Preconditioning

Examples:

■ Incomplete LU factorization (e.g. ILU(0))
■ Jacobi (inverse of the diagonal)
■ Other stationary iterative solvers (GS, SOR, SSOR)
■ Block preconditioners and domain decomposition
  ■ Additive Schwarz (thing of Block-Jacobi)
  ■ Multiplicative Schwarz (think of Block-GS)
Examples so far:

- algebraic preconditioners, i.e. exclusively based on the the matrix

Often, for problems coming from PDEs, PDE and discretization information can be used in designing a preconditioner, e.g.

- FFTs’ can be involved to approximate differential operators on regular grids (as in Fourier space the operators are diagonal matrices)

- Grid and problem information to define multigrid preconditioners

- Indefinite problems are often composed of sub-blocks that are definite: used in defining specific preconditioners and even modify solvers for these needs, etc.
Part III

Iterative eigen-solvers
Iterative Eigen-Solvers

How are iterative eigensolvers related to Krylov subspaces?

Projection and Eigen-Solvers
- The problem: Solve $Ax = \lambda x$ in $\mathbb{R}^n$.
- As in linear solvers: at iteration $i$ extract an approximate $x_i$ from a subspace $V = \text{span}\{v_1, \ldots, v_m\}$ of $\mathbb{R}^n$.
- How? As on slides 22 and 26, impose constraints: $\lambda x - Ax \perp \text{subspace } W = \text{span}\{w_1, \ldots, w_m\}$ of $\mathbb{R}^n$, i.e.
  \[
  \langle \lambda x, w \rangle = 0 \quad \text{for } \forall w \in W = \text{span}\{w_1, \ldots, w_m\}
  \]
- This procedure is known as Rayleigh-Ritz.
- Again projection can be orthogonal or oblique.

Matrix representation
- Let $V = \{v_1, \ldots, v_m\}$, $W = \{w_1, \ldots, w_m\}$.
  - Find $y \in \mathbb{R}^m$ s.t. $x = V y$ solves $Ax = \lambda x$, i.e.
    \[
    A V y = \lambda V y
    \]
    subject to the orthogonality constraints:
    \[
    W^T A V y = \lambda W^T V y
    \]
- The choice for $V$ and $W$ is crucial and determines various methods (more in Lectures 4 and 5).

Remember projection slides 29 & 30, Lecture 7 (left)

- Again, as in linear solvers, projection in a subspace is the basis for an iterative eigen-solver.
  - $V$ and $W$ are often based on Krylov subspaces
  \[
  K_m(A, r_0) = \text{span}\{r_0, A r_0, A^2 r_0, \ldots, A^{m-1} r_0\}
  \]
  where $r_0 \equiv b - Ax_0$ and $x_0$ is an initial guess.

- Often parts of $V$ or $W$ are orthogonalized.
  - For stability.
  - The orthogonalization can be CGS, MGS, Cholesky or Householder based, etc.
  - The smaller Rayleigh-Ritz are usually solved with LAPACK routines.
Learning Goals

- A brief introduction to Krylov iterative solvers and eigen-solvers
  - Links to building blocks that we have already covered
    - Abstract formulation
    - Projection, and
    - Orthogonalization
  - Specific examples and issues
    (preconditioning, parallelization, etc.)