

# Lecture 15:CMSC 878R/AMSC698R

# Outline

- Direct Solution of Linear Systems
  - Inverse, LU decomposition, Cholesky, SVD, etc.
- Iterative methods for linear systems
  - Why?
- Matrix splittings and fixed-point schemes
  - SOR, Jacobi, Gauss Seidel, etc.
- Krylov Methods
  - Conjugate Gradient, GMRES
- Convergence and Preconditioning
- Eigenvalue Problems
- Iterative methods for eigenvalue problems

# Preliminary Comments

- FMM accelerates matrix-vector multiplication
- However, most problems require solution of linear systems
- Goal: See how linear systems can be solved using FMM
- Take home message
  - Iterative methods require matrix vector products
  - Can be written as FMM
  - Some methods can be guaranteed to converge in  $N$  steps
  - With good guess and clever algorithms may converge much faster
- Philosophy:
  - In general it is better to be a “consumer” of linear algebra
  - Use canned software which asks you to provide a product routine
  - However, with FMM there may not be “canned” or “tuned” software available and we may have to develop our own

# Direct Solution of linear systems

$$\mathbf{Ax}=\mathbf{b} \rightarrow \mathbf{x}=\mathbf{A}^{-1}\mathbf{b}$$

- Almost never a good idea to construct inverse to solve
- Computing inverse is more expensive than solving system
  - Computing inverse leads to loss of precision
- Exceptions
  - Analytical expression for inverse can be obtained, e.g.,

Sherman Morrison Woodbury

$$(\mathbf{A} + \mathbf{u} \otimes \mathbf{v})^{-1} = \mathbf{A}^{-1} - \frac{(\mathbf{A}^{-1}\mathbf{u}) \otimes (\mathbf{v} \cdot \mathbf{A}^{-1})}{1 + \lambda}, \quad \lambda \equiv \mathbf{v} \cdot \mathbf{A}^{-1}\mathbf{u}.$$

Woodbury Formula

$$(\mathbf{A} + \mathbf{UV}^T)^{-1} = \mathbf{A}^{-1} - [\mathbf{A}^{-1}\mathbf{U}(1 + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^T\mathbf{A}^{-1}].$$

# Direct solutions (2)

- Consumer level – Use LAPACK or MATLAB,  
– (vectorized, cache-optimized, etc.) -- ATLAS
- For non symmetric matrices use LU decomposition
- LU decomposition  $\simeq N^3/3$  operations
- Backsubstitution  $O(N^2)$
- For symmetric matrices use Cholesky decomposition  $\simeq N^3/6$  operations
- All these methods require storage of the matrix and for dense matrices have memory complexity  $O(N^2)$

# References

- C.T. Kelley, “Iterative methods for Linear and Nonlinear Equations, SIAM, 1995)
- J. Shewchuck, “An Introduction to the Conjugate Gradient Method Without the Agonizing Pain”  
(downloadable from <http://www-2.cs.cmu.edu/~jrs/jrspapers.html>)
- “Templates for the solution of linear systems: Building Blocks and Iterative Methods,” Barrett et al, SIAM  
(downloadable at [http://www.netlib.org/linalg/html\\_templates/Templates.html](http://www.netlib.org/linalg/html_templates/Templates.html) )
- Yousef Saad has two good books online  
*Numerical Methods for Large Eigenvalue Problems*  
*Iterative methods for sparse linear systems.*  
(downloadable at <http://www-users.cs.umn.edu/~saad/books.html>)

# Iterative Methods: Notation

$$Ax = b$$

- $A$  is a nonsingular  $N \times N$  matrix,  $x, b \in R^N$
- $Ax$  can be calculated using FMM
- $x^*$  is the solution to be found.  $x^* = A^{-1}b \in R^N$

## • Definitions

Norm of  $A$

$$\|A\| = \max_{\|x\|=1} \|Ax\|$$

Condition number of  $A$

$$\kappa(A) = \|A\| \|A^{-1}\|$$

Residual

$$r = b - Ax$$

Error

$$e = x - x^*$$

# Fixed point iteration

- In fixed point iteration we write  $x=Mx$
- If  $M$  is a contraction ( $\|M\| < 1$ ) then following converges
  - Start with guess  $x_0$
  - Generate successive estimates  $x_k=M x_{k-1}$
- How to write our equation as a fixed point scheme?

$$Ax=Ix + (A-I)x =b$$

- So,  $I x = (I-A)x +b$
- (Richardson iteration)  $x_{k+1} = (I-A)x_k +b$
- For convergence we require  $\|I-A\| < 1$
- Iteration Matrix  $M$  here is  $\|I-A\|$

# Preconditioning

- What if  $\|I-A\| = \text{or} > 1$ ?
- Let  $B$  be another matrix.
- Then  $BAx = Ix + (BA-I)x = Bb$
- So, 
$$Ix = (I-BA)x + Bb$$
- (Richardson iteration) 
$$x_{k+1} = (I-BA)x_k + Bb$$
- For convergence we require  $\|I-BA\| < 1$
- If  $B$  is simple (e.g., diagonal) this is easy to compute
- Iteration matrix  $M = \|I-BA\|$

# Classical fixed point methods

- Write  $A=A_1+A_2$
- Then iteration becomes  $x_{k+1} = A_1^{-1}(b - A_2x_k)$ 
  - For convergence  $\|A_1^{-1}A_2\| < 1$
  - $A_1^{-1}$  should be easy to compute
  - In addition the FMM should be used to compute  $A_2x_k$
- Jacobi iteration  $A_1=D$   $A_2=L+U$ 
  - $A_1^{-1}$  is easy to compute (1/ entries along diagonal)
  - This is easy to compute with the FMM
  - At element level  $(x_{k+1})_i = a_{ii}^{-1} \left( b_i - \sum_{j \neq i} a_{ij} (x_k)_j \right)$
- Other classical iterations (Gauss-Seidel, SOR, etc. are harder to write in a way that FMM can be used).

# Krylov methods

- Different class of methods
- Do not involve an iteration matrix
- Motivation: Say in functional space we are at a point  $x_0$  and we want to reach the solution  $x^*$
- Can do it by taking steps along some directions
- Method of steepest descent
- Define function  $f(x)$   $f(x) = \frac{1}{2}x^t Ax - b^t x$
- So minimum of  $f(x)$  is attained at  $\nabla f(x) = 0$

$$\nabla f = \frac{1}{2} (A + A^t) x - b = 0$$

$$\frac{1}{2} (A + A^t) x = b \quad \text{or} \quad Ax = b$$

for symmetric  $A$

# Steepest descent

- So finding minimizer of the quadratic form gives a solution
- Start at guess  $x_0$ , take a step along  $-\nabla f(x_0)$

$$\nabla f(x_0) = Ax_0 - b = -r_0$$

$$x_1 = x_0 + \alpha r_0$$

- How big should the step  $\alpha$  be?
- We can find  $\alpha$  that minimizes  $f(x_0 + \alpha r_0)$

$$\frac{d}{d\alpha} f(x_0 + \alpha r_0) = r_0 \cdot \nabla f|_{x=x_0 + \alpha r_0}$$

- So if  $\alpha$  should be chosen so that  $r_0$  is orthogonal to  $\nabla f$

- Can show 
$$\alpha = \frac{r_0^t r_0}{r_0^t A r_0}$$

# Steepest Descent method

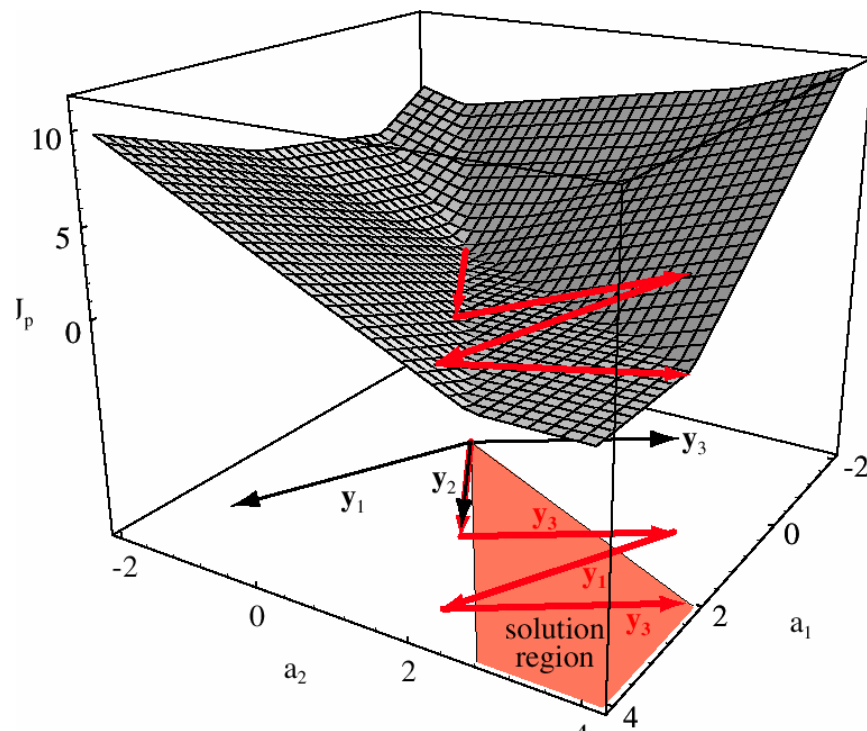
- Putting things together
- Requires two matrix vector products with  $A$  per iteration
- Can reduce one
- Can convert last equation to

$$r_{i+1} = r_i - \alpha_i A r_i$$

- only need calculate  $A r_i$
- What about convergence?
- Can show at  $k$ th iteration

$$\|e_k\|_A \leq \left( \frac{\kappa(A) - 1}{\kappa(A) + 1} \right)^k \|e_0\|_A \quad \text{where} \quad \kappa(A) = \lambda_{max} / \lambda_{min}$$

$$r^{(i)} = b - Ax^{(i)},$$
$$\alpha^{(i)} = \frac{r^{(i)T} r^{(i)}}{r^{(i)T} A r^{(i)}},$$
$$x^{(i+1)} = x^{(i)} + \alpha^{(i)} r^{(i)}.$$



# Conjugate Gradient

- Instead of minimizing each time along gradient, choose a set of basis directions to minimize along
- Choose these directions to be from the Krylov subspace
- Definition: Krylov subspace  $\mathcal{K}_k = \text{span}(r_0, Ar_0, \dots, A^{k-1}r_0)$
- Definition: Energy or  $A$ -norm of a vector  $\|x\|_A = (x^t Ax)^{1/2}$
- Idea of conjugate gradient method
  - Generate Krylov subspace directions, and take a step that minimizes the  $A$  norm of the residual along this direction
- Let search direction be at step  $k+1$  be  $d_{k+1}$
- We require  $f(x_k + \alpha_{k+1} d_{k+1})$  is minimized along direction
- Conjugacy property  $d_{k+1}^t A K^k = 0$

# CG Algorithm

- Each iteration requires computation of one matrix vector product  $Ad_i$
- Guaranteed to converge after  $N$  iterations (for exact arithmetic)
- So FMM guaranteed to solve equations in  $O(N^2)$  time
- Method does not require storage of any of the Krylov basis vectors
- Trivial to add preconditioning
- Implemented in Matlab as **pcg**

$$d_{(0)} = r_{(0)} = b - Ax_{(0)},$$

$$\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{d_{(i)}^T Ad_{(i)}} \quad ($$

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)},$$

$$r_{(i+1)} = r_{(i)} - \alpha_{(i)} Ad_{(i)},$$

$$\beta_{(i+1)} = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}},$$

$$d_{(i+1)} = r_{(i+1)} + \beta_{(i+1)} d_{(i)}.$$

# Convergence of CG

- If there are  $k$  distinct eigenvalues of  $A$ , then CG converges in at most  $k$  iterations to the exact solution (in exact arithmetic)
- Usually we go to  $\|b - Ax_k\|_2 \leq \eta \|b\|_2$

$$\frac{\|b - Ax_k\|_2}{\|b\|_2} \leq \frac{\sqrt{\kappa(A)} \|r_0\|_2}{\|b\|_2} \frac{\|x_k - x_*\|_A}{\|x_0 - x_*\|_A}$$

- Converges quickly

# Non symmetric matrices

- Can't apply CG to nonsymmetric matrices
- One simple solution – CGNR j
  - convert system to a symmetric system

$$A^tAx=A^tb$$

- However we will need two matrix vector multiplies per iteration
- Also, if  $A$  is poorly conditioned then  $A^tA$  is even more poorly conditioned (condition number is squared)
- However the method requires no storage of Krylov basis.

# GMRES

- Instead of requiring minimization along conjugate direction, minimize residual in a subspace
- Krylov subspace  $\mathcal{K}_k = \text{span}(r_0, Ar_0, \dots, A^{k-1}r_0)$
- Require  $x$  to minimize  $\|b - Ax\|_2 \quad \forall x \text{ in } x_0 + \mathcal{K}^k$
- Construct basis
 

```

w(i) = Av(i)
for k = 1, ..., i
    w(i) = w(i) - (w(i), v(k))v(k)
end
v(i+1) = w(i) / ||w(i)||
      
```
- Then require that each  $x^{(i)} = x^{(0)} + y_1 v^{(1)} + \dots + y_i v^{(i)}$ , satisfies minimum  $\|b - Ax^{(i)}\|$
- Can be done by simple minimization
- Implemented as a black-box in Matlab

# Bi Conjugate methods

- Since  $A$  is non symmetric,  $A^t$  and  $A$  have different eigenvalues and eigenvectors
- Form the Krylov basis of  $A^t$  for some vector  $g$  (usually equal to  $r_0$ )  $\bar{\mathcal{K}}_k = \text{span}(r_0, A^t r_0, \dots, (A^t)^{k-1} r_0)$
- Minimize successively along directions so that  $p_k A d_l = 0$
- Storage cost is lower ... Bi-CGSTAB

# Eigenvalue problems: Power Iteration

---

- Computes the dominant eigen value and eigen vector of a given matrix  $A$ .
- $x_k = Ax_{k-1}$  converges to the dominant eigen vector  $v_1$
- Ratio of values of given component of  $x_k$  from one iteration to next converges to the dominant eigen value  $\lambda_1$ .
- If two eigen values are equal may converge to a linear combination of the corresponding eigen vectors.
- To prevent overflow or underflow use Normalized power iteration.  $x_k = Ax_{k-1}$   $x_k = x_k / \|x_k\|_\infty$ .
- Can do power iteration with shift to increase convergence rate.