Lecture

• Monday – Iterative Methods: MINRES and GMRES

• Wednesday – Practicum

• Lecture 20 and 21: Based Upon Material From Professor Uri Ascher, Computer Science, Univ. of British Columbia
Non-Stationary Methods

Try to take advantage of accumulating knowledge as the iteration proceeds.
Consider $M = M_k$, and more generally

$$x_{k+1} = x_k + \alpha_k p_k.$$ 

The scalar $\alpha_k > 0$ is step size.

Restrict consideration to cases where $A$ is symmetric positive definite.
Then $Ax = b$ is equivalent to

$$\min \phi(x) = \frac{1}{2} x^T Ax - b^T x$$
Gradient Descent / Steepest Descent

**Gradient descent**: take $p_k = r_k$.

How to choose step size?

Simple-minded, greedy approach: *exact line search.*

$$
\min_{\alpha} \phi(x_k + \alpha r_k) = \frac{1}{2} (x_k + \alpha r_k)^T A (x_k + \alpha r_k) - b^T (x_k + \alpha r_k).
$$

Critical point: differentiate wrt $\alpha$ and set to 0. Obtain

$$
\alpha_k = \frac{r_k^T r_k}{r_k^T A r_k} = \frac{\langle r_k, r_k \rangle}{\langle r_k, A r_k \rangle}.
$$
Steepest Descent Algorithm

Given an initial guess \( x_0 \) and a tolerance \( tol \), set at first \( r_0 = b - A x_0 \), \( \delta_0 = \langle r_0, r_0 \rangle \), \( b_\delta = \langle b, b \rangle \) and \( k = 0 \). Then:
While \( \delta_k > tol^2 b_\delta \),

\[
\begin{align*}
    s_k &= A r_k \\
    \alpha_k &= \frac{\delta_k}{\langle r_k, s_k \rangle} \\
    x_{k+1} &= x_k + \alpha_k r_k \\
    r_{k+1} &= r_k - \alpha_k s_k \\
    \delta_{k+1} &= \langle r_{k+1}, r_{k+1} \rangle \\
    k &= k + 1.
\end{align*}
\]
Conjugate Gradient (CG) Algorithm

Given an initial guess $x_0$ and a tolerance $tol$, set at first $r_0 = b - Ax_0$, $\delta_0 = \langle r_0, r_0 \rangle$, $b_\delta = \langle b, b \rangle$, $k = 0$ and $p_0 = r_0$. Then:

While $\delta_k > tol^2 b_\delta$,

\[
\begin{align*}
    s_k &= Ap_k \\
    \alpha_k &= \frac{\delta_k}{\langle p_k, s_k \rangle} \\
    x_{k+1} &= x_k + \alpha_k p_k \\
    r_{k+1} &= r_k - \alpha_k s_k \\
    \delta_{k+1} &= \langle r_{k+1}, r_{k+1} \rangle \\
    p_{k+1} &= r_{k+1} + \frac{\delta_{k+1}}{\delta_k} p_k \\
    k &= k + 1.
\end{align*}
\]
Krylov Subspace

Easy to see that

\[ r_k = p_k(A)r_0, \]

where \( p_k \) is a polynomial of degree \( k \) satisfying \( p_k(0) = 1 \). Also

\[ e_k = p_k(A)e_0, \quad x_k - x_0 = q_{k-1}(A)r_0. \]

Define Krylov Subspace of nonsingular \( C \) with respect to \( y \) by

\[ \mathcal{K}_k(C; y) = \text{span}\{y, Cy, C^2y, \ldots, C^{k-1}y\}. \]

Thus,

\[ r_k \in \mathcal{K}_{k+1}(A; r_0), \quad \text{and} \]
\[ x_k - x_0 \in \mathcal{K}_k(A; r_0). \]
For $B$ symmetric positive definite matrix,

$$\|x\|_B = \sqrt{x^T B x} \equiv \sqrt{\langle x, B x \rangle}.$$ 

Then

$$\|r_k\|_{A^{-1}} = \|e_k\|_A.$$ 

Can interpret CG as minimizing in $k$th iteration the energy norm $\|e_k\|_A$ over space $x_0 + \mathcal{K}_k(A; r_0)$. 
Hence exact convergence in exact arithmetic after $n$ iterations.
CG Properties

- Subspace minimization (and uninteresting exact termination)
- Search directions are $A$-conjugate:
  $$\langle p_l, Ap_j \rangle = 0, \ l \neq j$$

- Residual directions are orthogonal:
  $$\langle r_l, r_j \rangle = 0, \ l \neq j$$

- \[
  \|e_k\|_A \leq 2 \left( \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|e_0\|_A,
  \]
  hence for large $\kappa$, number of iterations $k$ for reducing initial error by factor $c$
  $$k \leq .5 \sqrt{\kappa(A)} \ln(2/c) + 1.$$
Steepest Descent versus CG

- Gradient descent uses residual $r_k$ as search direction; CG uses
  $$p_k = r_k + \frac{\|r_k\|^2}{\|r_{k-1}\|^2} p_{k-1}.$$  
- SD is simpler, greedy, one-step, easier to show convergence. CG is
two-step (requires initialization), can be more fragile.
- SD requires $O(\kappa(A))$ iterations whereas CG requires $O(\sqrt{\kappa(A)})$
  iterations!
- For both methods
  $$r_k \in \mathcal{K}_{k+1}(A; r_0), \quad x_k - x_0 \in \mathcal{K}_k(A; r_0).$$
- For CG also
  $$x_k = \arg\min_{y \in S} \phi(y) = \arg\min_{z \in S} \|x - z\|_A$$
  where $S = x_0 + \mathcal{K}_k(A; r_0)$. Thus, convergence (in exact arithmetic)
in at most $n$ iterations.
Preconditioning

Often, even $O(\sqrt{\kappa(A)})$ iterations are too many!
(In Poisson example, $\sqrt{\kappa(A)} \sim N$.)
Consider solving instead

$$(P^{-1/2}AP^{-1/2})(P^{1/2}x) = P^{-1/2}b$$

with preconditioner $P$ such that matrix is better conditioned.
Fortunately, can simply consider

$$P^{-1}Ax = P^{-1}b.$$ 

The search is on for a preconditioner that is both close enough to $A$
to get a good condition number and far enough to be easily invertible.
Preconditioned CG

Given an initial guess $\mathbf{x}_0$ and a tolerance $tol$, set at first $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{h}_0 = P^{-1}\mathbf{r}_0$, $\delta_0 = \langle \mathbf{r}_0, \mathbf{h}_0 \rangle$, $b_\delta = \langle \mathbf{b}, P^{-1}\mathbf{b} \rangle$, $k = 0$ and $\mathbf{p}_0 = \mathbf{h}_0$. Then:

While $\delta_k > tol^2 b_\delta$,

\[
\begin{align*}
\mathbf{s}_k &= A\mathbf{p}_k \\
\alpha_k &= \frac{\delta_k}{\langle \mathbf{p}_k, \mathbf{s}_k \rangle} \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k \mathbf{p}_k \\
\mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_k \mathbf{s}_k \\
\mathbf{h}_{k+1} &= P^{-1}\mathbf{r}_{k+1} \\
\delta_{k+1} &= \langle \mathbf{r}_{k+1}, \mathbf{h}_{k+1} \rangle \\
\mathbf{p}_{k+1} &= \mathbf{h}_{k+1} + \frac{\delta_{k+1}}{\delta_k} \mathbf{p}_k \\
k &= k + 1.
\end{align*}
\]
Choices for Preconditioners

- Symmetric SOR (SSOR)
- Incomplete Cholesky (IC) \( P = FF^T \); more generally incomplete LU (ILU)
  - IC('0') – avoid fill-in altogether
  - IC(tol) – carry out elimination step if result above drop tolerance tol
- Often unknown in practice!
General Non-Singular Solution

- Assume $A$ nonsingular, but not necessarily symmetric positive definite.
- Extend CG by searching in similar Krylov subspaces:
  \[ x_k - x_0 \in \mathcal{K}_k(A; r_0) \]
  \[ \mathcal{K}_k(A; r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{k-1}r_0\}. \]

- **Building blocks**
  1. Construct an orthogonal basis for the Krylov subspace.
  2. Define an optimality property.
  3. Use an effective preconditioner.
Orthogonal Basis: Arnoldi

- The obvious basis (powers $A^i r_0$) is poorly conditioned.
- So orthonormalize these vectors: Arnoldi algorithm.

$$q_1 = r_0 / \|r_0\|$$

for $j = 1$ to $k$

$$z = Aq_j$$

for $i = 1$ to $j$

$$h_{i,j} = \langle q_i, z \rangle$$

$$z = z - h_{i,j} q_i$$

end

$$h_{j+1,j} = \|z\|$$

if $h_{j+1,j} = 0$, quit

$$q_{j+1} = z / h_{j+1,j}$$

end
Algorithm 9.1 Arnoldi iteration (classical Gram-Schmidt variant)

1: \[ v_1 = r_0 / \|r_0\|_2 \]
2: for \( j = 1, 2, \ldots, m \) do
3: for \( i = 1, 2, \ldots, j \) do
4: \quad Compute \( h_{ij} = (A v_j, v_i) \)
5: end for
6: Compute \( w_j = A v_j - \sum_{i=1}^{j} h_{ij} v_i \)
7: \( h_{j+1,j} = \|w_j\|_2 \)
8: if \( h_{j+1,j} = 0 \) then Stop
9: \( v_{j+1} = w_j / h_{j+1,j} \)
10: end for
Algorithm 9.2 Arnoldi iteration (Modified Gram-Schmidt variant)

1: \( v_1 = r_0/\|r_0\|_2 \)
2: for \( j = 1, 2, \ldots, m \) do
3: \hspace{1em} Initialize \( w_j = A v_j \)
4: \hspace{1em} for \( i = 1, 2, \ldots, j \) do
5: \hspace{2em} Compute \( h_{ij} = (w_j, v_i) \)
6: \hspace{2em} Update \( w_j = w_j - h_{ij} v_i \)
7: \hspace{1em} end for
8: \( h_{j+1,j} = \|w_j\|_2 \)
9: if \( h_{j+1,j} = 0 \) then Stop
10: \( v_{j+1} = w_j / h_{j+1,j} \)
11: end for
Orthogonal Basis: Lanczos

- If $A$ is symmetric, the upper Hessenberg $H$ is tridiagonal.
- Obtain three-term recurrence: Lanczos algorithm.

\[
\begin{align*}
    z &= Aq_j \\
    \gamma_j &= \langle q_j, z \rangle \\
    z &= z - \beta_{j-1}q_{j-1} - \gamma_jq_j \\
    \beta_j &= \|z\| \\
    q_{j+1} &= z/\beta_j.
\end{align*}
\]

Start with $q_1 = r_0/\|r_0\|$, $\beta_0 = 0$. 
Algorithm 9.3 Lanczos iteration

1: $\beta_1 = 0$, $v_0 = 0$
2: $v_1 = r_0/\|r_0\|_2$
3: for $j = 1, 2, \ldots, m$ do
4: Initialize $w_j = Av_j - \beta_j v_{j-1}$
5: Compute $\alpha_j = (w_j, v_j)$
6: Update $w_j = w_j - \alpha_j v_j$
7: $\beta_{j+1} = \|w_j\|_2$
8: if $\beta_{j+1} = 0$ then Stop
9: $v_{j+1} = w_j / \beta_{j+1}$
10: end for
Direct Lanczos

Algorithm 9.4 D-Lanczos

1: Compute $r_0 = b - Ax_0$; $\zeta_1 = \beta = \|r_0\|_2$; $v_1 = r_0/\beta$
2: $\lambda_1 = \beta_1 = 0$, $d_0 = 0$
3: for $m = 1, 2, \ldots$, do
4:     Compute $w = A v_m - \beta_m v_{m-1}$ and $\alpha_m = (w, v_m)$
5:     if $m > 1$ then compute $\lambda_m = \frac{\beta_m}{\eta_{m-1}}$ and $\zeta_m = -\lambda_m \zeta_{m-1}$
6:     $\eta_m = \alpha_m - \lambda_m \beta_m$
7:     $d_m = (v_m - \beta_m d_{m-1})/\eta_m$
8:     $x_m = x_{m-1} + \zeta_m d_m$
9:     if $x_m$ has converged, then Stop
10:    $w = w - \alpha_m v_m$; $\beta_{m+1} = \|w\|_2$; $v_{m+1} = w/\beta_{m+1}$
11: end for
GMRES and MINRES

Minimize residual norm.

- Main components of GMRES iteration:
  1. perform a step of the Arnoldi process;
  2. update the QR factorization of the updated upper Hessenberg matrix;
  3. solve the resulting least squares problem.
- GMRES\((m)\): limited memory GMRES – restart after \(m\) iterations
- MINRES: for the symmetric case: no memory problem.
Algorithm 10.1 GMRES (basic form)

1: Compute $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
2: Define the $(m+1) \times m$ matrix $H_m$ and initialize elements $h_{ij}$ to zero
3: for $j = 1, 2, \ldots, m$ do
4:    Compute $w_j = Av_j$
5:    for $i = 1, \ldots, j$ do
6:        $h_{ij} = (w_j, v_i)$
7:        $w_j = w_j - h_{ij} v_i$
6:    end for
9:    $h_{j+1,j} = \|w_j\|_2$. if $h_{j+1,j} = 0$ set $m = j$ and goto 12
10:   $v_{j+1} = w_j / h_{j+1,j}$
11: end for
12: Compute $u_m$ as the minimizer of $\|\beta c_1 - H_m u\|_2$ and set $x_m = x_0 + V_m u_m$
Exercise 10.9 The Krylov method analogous to GMRES for the special case of a symmetric, possibly indefinite matrix $A$ is called the Minimal Residual Method (MINRES). Similarly like the D-Lanczos method (which is of FOM type), MINRES involves tridiagonal matrices $H_m = T_m$, which leads to compact update formulas similar to D-Lanczos. Explore this version referring to Section 6.6 in [16] (‘Symmetric Lanczos Algorithm’).
Preconditioning

- Incomplete LU (ILU)
- Incomplete LU with drop tolerance (ILUT)
- Example: convection-diffusion equation
  \[- \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \sigma \frac{\partial u}{\partial x} + \tau \frac{\partial u}{\partial y} = g(x, y).\]
Multigrid Methods

- Consider iteration for model problem: Poisson equation on the unit square.
- Highest frequencies of residual or error correspond to largest eigenvalues, most oscillatory eigenvectors. These are “not seen” on a coarser grid.
- Simple relaxations such as damped Jacobi ($\omega = .8$) or Gauss-Seidel are slow methods but fast smoothers (reduce high frequency components of residual/error).
- Smoothed residual can be transferred to a coarser grid, where procedure can be repeated, more economically.
For Next Time

• Reread Chapter 3, read Chapter 8 and get additional material
• Start working on Practicums
• Start working on Homework 5