CS 205 – Class 12

Readings: Same as last
Covered in class: All

1. finding the A-orthogonal directions with Gram-Schmidt
   a. given a vector V_k, construct s_k by subtracting out the "A-overlap" of V_k with s_1 to s_{k-1} so that $s_k \cdot As_i = 0$ for i=1,k-1
   b. we define $s_k = V_k - \sum_{j=1}^{k-1} \frac{V_k \cdot As_j}{s_j \cdot As_j} s_j$
      i. note that $s_k \cdot As_i = V_k \cdot As_i - \sum_{j=1}^{k-1} \frac{V_k \cdot As_j}{s_j \cdot As_j} s_j \cdot As_i$ and then all the terms in the sum vanish except for one leaving $s_k \cdot As_i = V_k \cdot As_i - \frac{V_k \cdot As_i}{s_i \cdot As_i} s_i \cdot As_i = 0$ as desired
   c. for $i \geq k$, $s_k \cdot r_i = V_k \cdot r_i - \sum_{j=1}^{k-1} \frac{V_k \cdot As_j}{s_j \cdot As_j} s_j \cdot r_i = V_k \cdot r_i$, where the summation vanishes because the residual at step i is orthogonal to all the previous search directions
      i. when k=i this leads to $s_k \cdot r_k = V_k \cdot r_k$ and $\alpha_k = \frac{s_k \cdot r_k}{s_k \cdot As_k} = \frac{V_k \cdot r_k}{s_k \cdot As_k}$ (we'll use this below)
      ii. when k < i, $0 = V_k \cdot r_i$, i.e. the residual is orthogonal to all the previous V_k as well (we'll use this below)

2. Each new direction V is chosen in the steepest decent fashion, i.e. $V_k = -\nabla f(x_k) = r_k$.
   a. $\alpha_{k-1} = \frac{r_{k-1} \cdot r_{k-1}}{s_{k-1} \cdot As_{k-1}}$
   b. Starting with $r_k = r_{k-1} - \alpha_{k-1} As_{k-1}$, we have $r_i \cdot r_k = r_i \cdot r_{k-1} - \alpha_{k-1} r_i \cdot As_{k-1}$ or $\alpha_{k-1} r_i \cdot As_{k-1} = r_i \cdot r_{k-1} - r_i \cdot r_k$
   c. When $i = k$, $\alpha_{k-1} r_k \cdot As_{k-1} = r_k \cdot r_{k-1} - r_k \cdot r_k = -r_k \cdot r_k$ and thus $r_k \cdot As_{k-1} = \frac{-r_k \cdot r_k}{\alpha_{k-1}}$
   d. When $i > k$, $\alpha_{k-1} r_i \cdot As_{k-1} = r_i \cdot r_{k-1} - r_i \cdot r_k = 0$, i.e. $ri \cdot As_{k-1} = 0$
   e. Thus, $s_k = r_k - \sum_{j=1}^{k-1} \frac{r_k \cdot As_j}{s_j \cdot As_j} s_j = r_k + \frac{r_k \cdot r_k}{\alpha_{k-1}(s_{k-1} \cdot As_{k-1})} s_{k-1}$ since only the last term in the sum is nonzero (Note how all the dot products disappear except for one!!)
   f. Finally, plugging in the definition of α_{k-1} gives $s_k = r_k + \frac{r_k \cdot r_k}{r_{k-1} \cdot r_{k-1}} s_{k-1}$ as desired.

3. Conjugate Gradient Method - the main idea is to search with conjugate directions
   a. $s_0 = r_0 = b - Ax_0$ which is the steepest decent direction

b. $a_{k-1} = \frac{r_{k-1} \cdot r_{k-1}}{s_{k-1} \cdot As_{k-1}}$

c. $x_k = x_{k-1} + \alpha_{k-1} s_{k-1}$ and $r_k = r_{k-1} + \alpha_{k-1} As_{k-1}$ as always

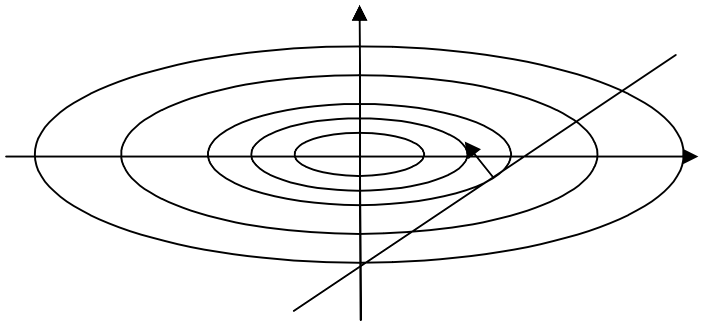
d. $s_k = r_k + \frac{r_k \cdot r_k}{r_{k-1} \cdot r_{k-1}} s_{k-1}$

4. Preconditioning
   a. If we had an approximate inverse, we can transform Ax=b into $\hat{A}^{-1}Ax = \hat{A}^{-1}b$ or $\hat{I}x = \hat{b}$ where $\hat{I}$ is approximately the identity matrix
   b. If all the eigenvalues of $\hat{I}$ are approximately equal to 1, then we have "circles" instead of "ellipses" and CG converges much faster because of the duplicate or near duplicate eigenvalues
   c. That is, preconditioning works great!
   d. Diagonal or Jacobi preconditioning scales the quadratic form along the coordinate axis to make it better conditioned (whereas it would be optimal to scale along the *eignevector* axis)
   e. Incomplete Choleski preconditioning does a Choleski factorization with the caveat that only the nonzero entries are modified, i.e. all the zeros remain zeroes

5. Constrained Optimization (not covered in class)
   a. Minimize $f(\vec{x})$ subject to constraints $\vec{g}(\vec{x}) = 0$
      i. Here $\vec{x} \in R^n$ and $\vec{g}(\vec{x}) = 0$ is as system of $m \le n$ equations
      ii. One can show that a solution $\vec{x}$ must satisfy $-\nabla f(\vec{x}) = J_g^T(\vec{x})\vec{\lambda}$
         1. $J_g(\vec{x})$ is the Jacobian matrix of g
         2. $\vec{\lambda}$ is an m-vector of *Lagrange multipliers*
         3. This condition says that we cannot reduce the objective function without violating the constraints
      iii. Define $L(\vec{x}, \vec{\lambda}) = f(\vec{x}) + \vec{\lambda}^T g(\vec{x})$
         1. The critical points are found by setting $\nabla L(\vec{x}, \vec{\lambda}) = \begin{bmatrix} \nabla f(\vec{x}) + J_g^T(\vec{x})\vec{\lambda} \\ g(\vec{x}) \end{bmatrix} = \vec{0}$
         2. Suppose for simplicity that g is a linear function. Then the Hessian is $H(\vec{x}, \vec{\lambda}) = \begin{bmatrix} H_f(\vec{x}) & J_g^T(\vec{x}) \\ J_g(\vec{x}) & 0 \end{bmatrix}$ where the x partial derivatives of $J_g^T(\vec{x})\vec{\lambda}$ vanish because g is linear.
            a. Note that H is not positive definite
            b. It turns out that positive definiteness is only needed on the tangent space to the constraint surface, i.e. on the null space of J_g.
      iv. Consider $f(x) = .5x_1^2 + 2.5x_2^2$ with $g(x) = x_1 - x_2 - 1 = 0$
         1. $L(\vec{x}, \vec{\lambda}) = .5x_1^2 + 2.5x_2^2 + \lambda\left(x_1 - x_2 - 1\right)$

2. $\nabla L(\vec{x}, \vec{\lambda}) = \begin{bmatrix} x_1 + \lambda \\ 5x_2 - \lambda \\ x_1 - x_2 - 1 \end{bmatrix} = \vec{0}$

3. so we solve $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 5 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ to obtain $\begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} .833 \\ -.167 \\ -.833 \end{bmatrix}$



The gradient of the function is perpendicular to the constraint surface at the constrained minimum.

6. Linear Programming (not covered in class)
   a. Minimize $\vec{c} \cdot \vec{x}$ subject to constraints $A\vec{x} = \vec{b}$ and $\vec{x} \geq \vec{0}$
   b. The feasible region is a convex polyhedron in n-dimensional space
   c. The minimum must occur at one of the vertices of the polyhedron
   d. *Simplex method* - systematically examine a sequence of vertices to find the one yielding the minimum