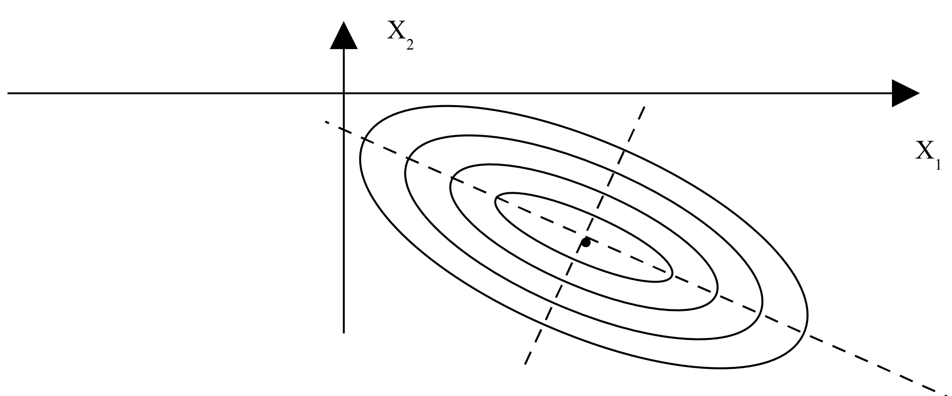
CS205 – Class 9

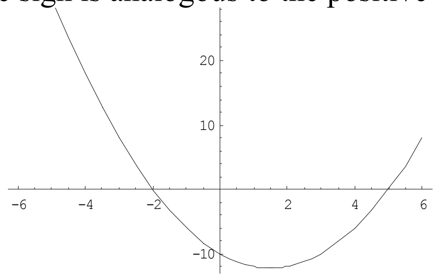
Covered in class: All
Reading: Shewchuk Paper on course web page

1. **Conjugate Gradient Method** – this covers more than just optimization, e.g. we'll use it later as an iterative solver to aid in solving pde's
2. Let's go back to linear systems of equations Ax=b.
    a. Assume that A is square, symmetric, positive definite
    b. If A is dense we might use a direct solver, but for a sparse A, iterative solvers are better as they only deal with nonzero entries
    c. Quadratic Form $f(x) = \frac{1}{2} x^T A x - b^T x + c$
    d. If A is symmetric, positive definite then f(x) is minimized by the solution x to Ax=b!
        i. $\nabla f(x) = \frac{1}{2} Ax + \frac{1}{2} A^T x - b = Ax - b$ since A is symmetric
        ii. $\nabla f(x) = 0$ is equivalent to Ax=b
            1. this makes sense considering the scalar equivalent $f(x) = \frac{1}{2} ax^2 - bx + c$ where the line of symmetry is $x = b/a$ which is the solution of ax=b and the location of the maximum or minimum
        iii. The Hessian is H=A, and since A is symmetric, positive definite so is H, and a solution to $\nabla f(x) = 0$, or Ax=b is a minimum
            1. note that symmetric negative definite A lead to maxima
            2. in the scalar case $f(x) = 1/2ax^2 - bx + c$, H=[a] and when a>0 the parabola is concave up and $x = b/a$ represents a minima
            3. [Note: Even if A is not symmetric, the Hessian $H = \frac{1}{2}\left(A + A^T\right)$ is symmetric itself, as expected since the quadratic function we considered has continuous second derivatives]
        iv. Moreover, since H=A is constant, f(x) has a bowl shape everywhere –



v. Consider this in 1D. We have $\begin{array}{l} f(x)=\frac{1}{2}xax-bx+c=\frac{1}{2}ax^2-bx+c \\ f'(x)=ax-b \end{array}$ so minimum is $x=b/a$.

Then the second derivative sign is analogous to the positive or negative definiteness of the general matrix case. Here



vi. *f(x)=1/2 * 2 * x^2-3x-10* minimum is at *b/a=3/2*.

3. Steepest Decent – for *Ax=b*
   a. We look in the direction $-\nabla f = b - Ax = r$. As we have shown, the residual direction is the steepest decent direction!
   b. Another way to think about the residual is $r = b - Ax = \; = Ax_{exact} - Ax = A(x_{exact} - x) = -Ae$ where $e = x - x_{exact}$ is the error. Thus, the residual is the error transformed by A into the space where b resides.
   c. $-\nabla f = r = -Ae$ so the search direction is predicted by r, not by e, whereas e is the correct search direction . Note that in 1d the directions of e and r are coincident, but in multi-d this problem manifests itself. The residual may or may not be a good measure of error. Consider 1D example with *r=ae*. Suppose r=10^{-8}. Then *e* could be arbitrarily large as we make *a* smaller (where *a* is the concavity).
   d. Recall that we choose α using a 1D minimization problem
      i. The solution occurs where the new $\nabla f(x)$ is orthogonal to the search line,
         1. i.e. go in the direction until you reach a spot where direction is tangent to level curves
         2. i.e. $\perp to\ \nabla f(x)$
         3. i.e. $\nabla f(x) \perp s_k$ where s_k is search direction at iteration k
         4. i.e. $\nabla f(x) \cdot s_k = 0$
         5. i.e. $\nabla f(x_{k+1}) \cdot r_k = 0$
         6. i.e. $r_{k+1} \cdot r_k = 0$.
      ii. If we knew the absolute error e_k, we could use it to write:
      $x_{k+1} = x_k + s_k \alpha = x_k - e_k \alpha = x_k - (x_k - x_{exact})\alpha$ gives $x_{k+1} = x_{exact}$ for $\alpha = 1$.
      iii. However, using $r_{k+1} \cdot r_k = 0$ implies $(b - Ax_{k+1}) \cdot r_k = 0$ or $(b - A(x_k + r_k \alpha)) \cdot r_k = 0$ or
      $(b - Ax_k) \cdot r_k - (Ar_k \alpha) \cdot r_k = 0$ or $r_k \cdot r_k - \alpha r_k \cdot Ar_k = 0$ so that $\alpha = \frac{r_k \cdot r_k}{r_k \cdot Ar_k} = \frac{r_k^T r_k}{r_k^T Ar_k}$
   e. So, the steepest decent method applied to Ax=b is $r_k = b - Ax_k$, $\alpha = \frac{r_k^T r_k}{r_k^T Ar_k}$, $x_{k+1} = x_k + r_k \alpha$
   f. Sometimes people iterate on the residual directly using $r_{k+1} = b - Ax_{k+1} = b - A(x_k + r_k \alpha) = r_k - \alpha Ar_k$ to find the r_k, while still updating $x_{k+1} = x_k + r_k \alpha$ along the way (although x no longer feeds back into the algorithm)

i. The advantage of this is that we no longer need the extra multiplication by A in $r_k = b - Ax_k$.

Both the computation of $\alpha = \frac{r_k^T r_k}{r_k^T A r_k}$ and $r_{k+1} = r_k - \alpha A r_k$ use the same Ar_k