Steepest Descent method

Friday, October 4, 2024 1:14 PM

Class 17: October 4, 2024

Recall: Last class, we went over the implementation details for the Broyden method, tested it against the same examples we ran for the Newton method. Finally, we went over the derivation of the Broyden update formula.

From this, the most important thing to remember about Broyden can be summarized as follows:

- Broyden is a QuasiNewton method that **retains superlinear convergence**, while addressing the **two limitations of Newton** in terms of potential high cost for the Newton step. That is: it only evaluates J_F(x0) one time (or none) and the linear system solve is done cleverly so it is much faster.
- It works by replacing J_F(x_k) (the Jacobian matrix of derivatives) by a matrix Bk that is close.
- Bk is obtained by rank 1 updates. We have an update formula Bk = B0 + UV^T where U,V are n x k matrices.
- Bk^{-1} can also be obtained by rank 1 updates. We have an update formula Bk^{-1} = B0^{-1} + UV^T where U,V are n x k matrices.
- A good implementation of Broyden breaks down the solve for Bk as a "fast" solve for B0 (e.g. LU) and a matrix vector product for UV^T.

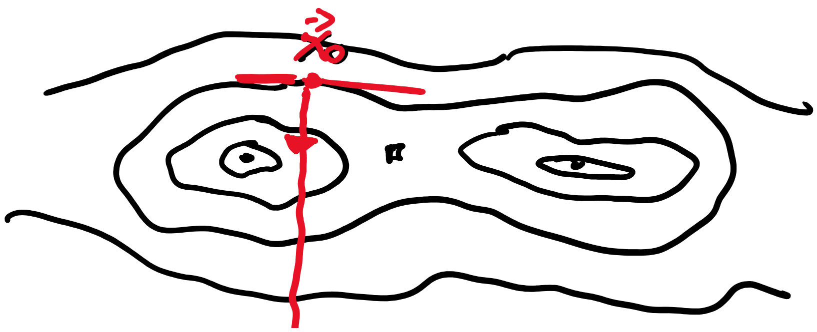
Much like it was true of secant and Newton, Broyden has the same issues Newton has when it comes to global convergence. It is only guaranteed to converge superlinearly near the root.

Today, we will introduce our final method for rootfinding: the Steepest Descent method. We do this by connecting rootfinding to optimization problems (minimization).

OPTIMIZATION PROBLEM	ROOTFINDING.
$\min q(\vec{x})$ (LOCAL MIN)	Find $\vec{r}$ s.t. $\underline{F(\vec{r}) = \vec{0}}$.
$q: \mathbb{R}^n \to \mathbb{R}$ smooth (C^2)	$F: \mathbb{R}^n \to \mathbb{R}^n$.

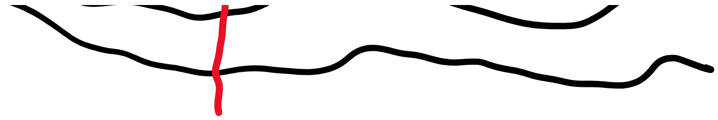
- $\underline{F(\vec{r}) = \vec{0}}$ $\qquad \underline{q(\vec{x})} = F_1(x)^2 + F_2(x)^2 + \ldots + F_n(\vec{x})^2$
- $\underline{\min q(\vec{x})} \longrightarrow$ CRITICAL PTS $\underline{\nabla q(\vec{r}) = \vec{0}}$.

Steepest Descent

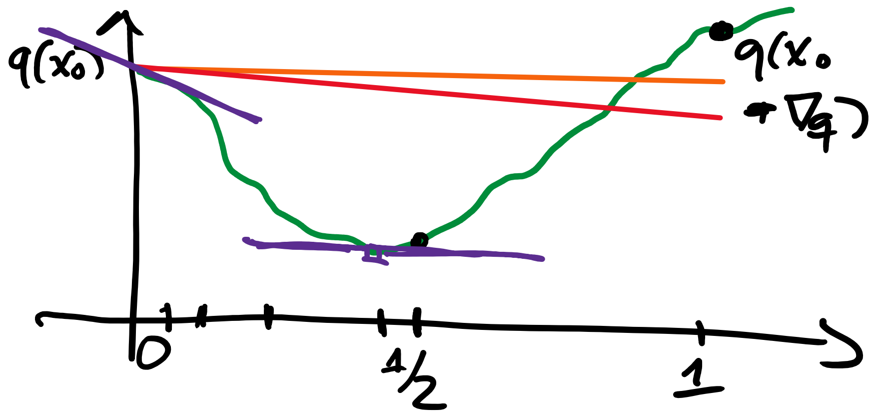


$D_{\vec{u}} q \to u^T \nabla q.$

$-\nabla q$ "DIRECTION OF Steepest Descent"



$$\phi(\alpha) = q(\vec{x}_0 - \alpha \nabla q) \qquad \alpha \in [0, 1]$$



How do we find α_k?
↓
"Line search" (Backtracking)

$$x_1 = x_0 - \alpha_0 \nabla q(x_0)$$

$$x_{k+1} = x_k - \alpha_k \nabla q(x_k) \rightarrow \underline{\text{SD step.}}$$

- Condition for sufficient descent (Armijo cond).
- Curvature (slope) conditions (Wolfe)

→ max # of backtrack.

$$F(x) = 0 \rightarrow q(x) = \left(F_1(x)^2 + F_2(x)^2 + \dots F_n(x)^2\right)\left(\tfrac{1}{2}\right)$$

$$\nabla q(\vec{x}) \; ?$$

$$\nabla q(\vec{x})\ ?$$

$$\frac{\partial q}{\partial x_j} = \Big(2F_1(x)\frac{\partial F_1}{\partial x_j} + 2F_2(x)\frac{\partial F_2}{\partial x_j} + \dots + 2F_n(x)\frac{\partial F_n}{\partial x_j} \Big)\left(\tfrac{1}{2}\right)$$

$$= \sum_{i=1}^{n} F_i(x)\frac{\partial F_i}{\partial x_j}(x).$$

Check.

$$\boxed{\nabla q = J_F(x)^T F(x)}$$