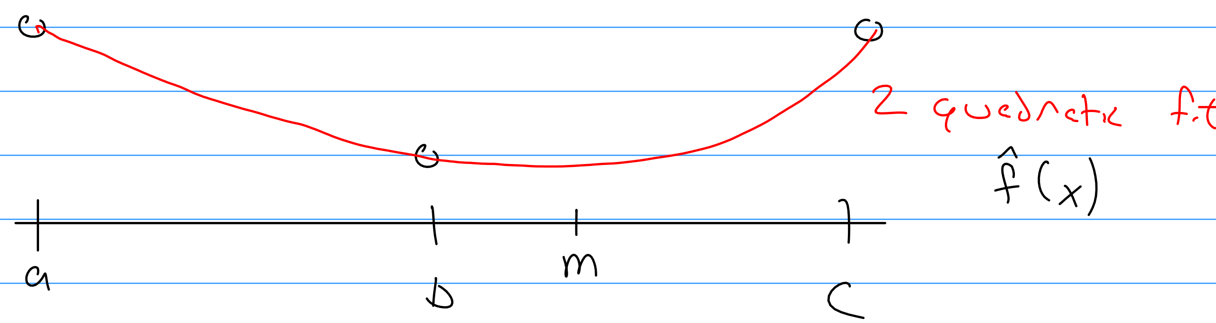
Minimization

Find minimum of objective function $f(x)$

1) Brent's Method - A bracketing method

Let $a < b < c$ such that $f(a) > f(b) < f(c)$

Minimum must exist in $x \in (a, c)$



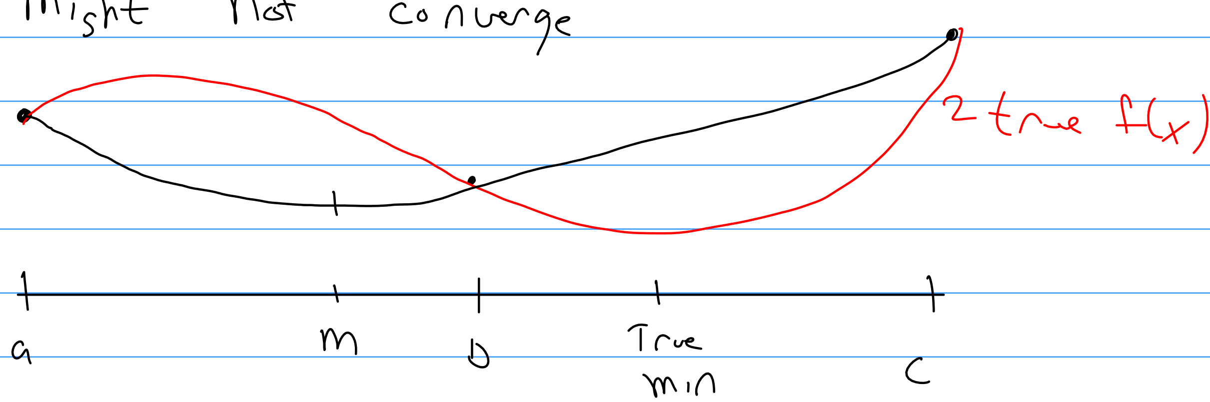
Solve $\hat{f}'(m) = 0$ $\quad \hat{f}(x) = \alpha x^2 + \beta x + \gamma$

If $a < m < b$ set (a, m, b) & repeat
$b < m < c$ set (b, m, c) & repeat

Iterate until converged.

1^{st}-order Convergence

Might not converge



2) Newton's Method

Extension of Newton - Rhapson

let x_n be close to x^* ← true minimum

Taylor Series of $f'(x)$ near x_n

$$f'(x^*) = f'(x_n) + f''(x_n)(x^* - x_n) + \tfrac{1}{2} f'''(x_n)(x^* - x_n)^2 + H.O.T.$$

Approx to first-order & solve:

$$f'(x_{n+1}) = f'(x_n) + f''(x_n)(x_{n+1} - x_n)$$

$$\Rightarrow x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$ ← Newton - Rhapson for $f'(x) = 0$

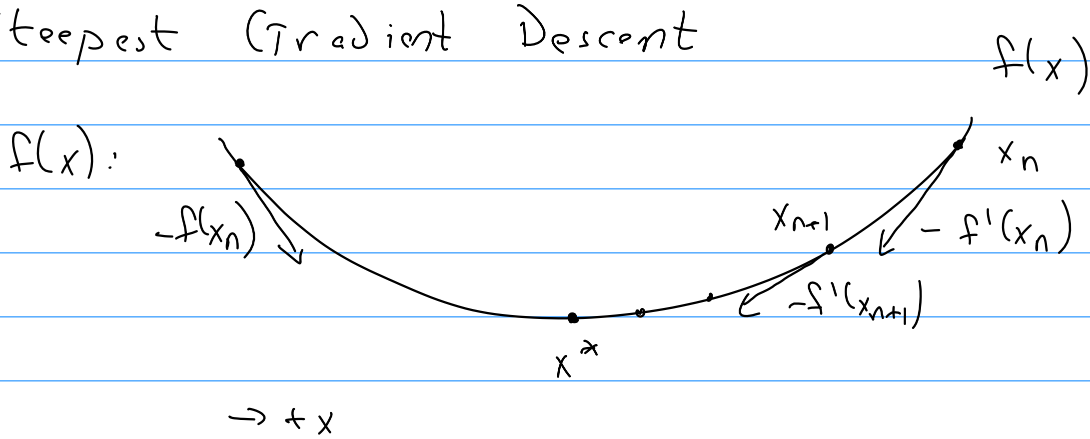
Might Diverge

Issue if $f''(x_n) \sim 0$

$f''(x_n)$ might be expensive

2^{nd} - order Convergence

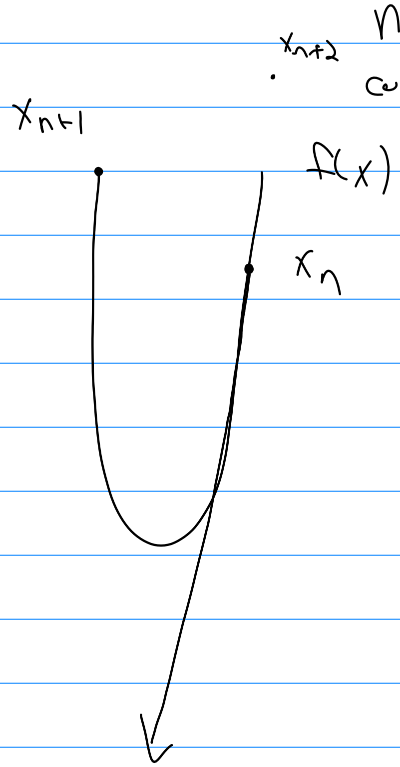
3) Steepest Gradient Descent



minimum in direction of $-f'(x_n)$

$$x_{n+1} = x_n - \alpha_n f'(x_n)$$

Need Damping due to no control over $f'(x)$



Solution: Require that $f(x_n - \alpha_n f'(x_n)) < f(x_n)$

Multiple Variable Minimization

Either minimize $\underbrace{f(x_1, x_2, \ldots, x_n)}$

or

minimize $f_1(x_1, \ldots, x_n)$ with some cost
$f_2(x_1, \ldots, x_n)$ function $g(f_1, f_2, \ldots f_m)$
$\vdots$
$f_m(x_1, \ldots, x_n)$

Methods: 1) Newton's Method
2) Quasi-Newton Methods
3) Steepest Gradient Descent

---

1) Newton's in multiple dim

If $\underline{x}^*$ is the minimum of $f(\underline{x})$ then

$$\nabla f(\underline{x}^*) = \begin{bmatrix} \partial f/\partial x_1 \big|_{\underline{x}^*} \\ \partial f/\partial x_2 \big|_{\underline{x}^*} \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

let $\underline{x}_n$ be near $\underline{x}^*$

$$\nabla f(\underline{x}^*) = \nabla f(\underline{x}_n) + \underline{H}(\underline{x}_n)(\underline{x}^* - \underline{x}_n) + \text{H.O.T.}$$

Hessian of $f(\underline{x})$

$$\underline{H}(\underline{x}) = \begin{bmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1 \partial x_2 & \partial^2 f/\partial x_1 \partial x_3 & \cdots & \partial^2 f/\partial x_1 \partial x_n \\ \partial^2 f/\partial x_1 \partial x_2 & \partial^2 f/\partial x_2^2 & \partial^2 f/\partial x_2 \partial x_3 & \cdots & \partial^2 f/\partial x_2 \partial x_n \\ & & \ddots & & \\ & & & \ddots & \\ \partial^2 f/\partial x_n \partial x_1 & & & & \partial^2 f/\partial x_n^2 \end{bmatrix}$$

Square & Symmetric matrix.

$\Rightarrow$ To order-1

$$\nabla f(\underline{x}_{n+1}) = \underbrace{\nabla f(\underline{x}_n)}_{\text{known}} + \underbrace{\underline{H}(\underline{x}_n)}_{\text{known}}(\underline{x}_{n+1} - \underline{x}_n) = \underline{0}$$

$$\Rightarrow \underline{x}_{n+1} = \underline{x}_n - \underline{H}_n^{-1} \nabla f_n$$

Typically Done as:

let $\underline{\delta}_{n+1} = \underline{x}_{n+1} - \underline{x}_n$

Solve $\underline{H}_n \underline{\delta}_{n+1} = -\nabla f_n \Rightarrow \underline{\delta}_{n+1} = -\underline{H}_n^{-1} \nabla f_n$

Update $\underline{x}_{n+1} = \underline{x}_n + \alpha_n \underline{\delta}_{n+1}$

Need to solve a linear system each iteration. Could be expensive.

but, 2^{nd}-order convergence

2) Quasi-Newton Methods

Use a newton method but approximate $\underline{H}_n$.

let $\underline{B}_n \approx \underline{H}_n$ w/ $\underline{B}_n^{-1}$ cheap(er) to compute than $\underline{H}_n^{-1}$

Then $\underline{x}_{n+1} = \underline{x}_n - \alpha_n \underline{B}_n^{-1} \nabla f_n$

Trick is how to get $\underline{B}_0$ & how to get $\underline{B}_{n+1}$ from $\underline{B}_n$.

General Algorithm:

let $\underline{x}_0$ & $\underline{B}_0$ be given

$n = 0$

until converged

$\underline{\delta}_n = -\underline{B}_n^{-1} \nabla f_n$

$\underline{x}_{n+1} = \underline{x}_n + \alpha_n \underline{\delta}_n$

$\underline{y}_n = \nabla f(\underline{x}_{n+1}) - \nabla f(\underline{x}_n) = \nabla [f(\underline{x}_{n+1}) - f(\underline{x}_n)]$

$\underline{B}_{n+1} =$ function of $\underline{B}_n, \underline{\delta}_n, \underline{y}_n$ $\leftarrow$

Sample Choice for $\underline{B}_0$:

let $\underline{\delta}_0 = -\nabla f(\underline{x}_0)$ $\qquad \underline{y}_0 = \nabla f(\underline{x}_0 + \underline{\delta}_0) - \nabla f(\underline{x}_0)$

$$\underline{B}_0 = \frac{\underline{y}_0^T \underline{y}_0}{\underline{y}_0^T \underline{\delta}_0} \underline{I} \implies \underline{B}_0^{-1} = \frac{\underline{y}_0^T \underline{\delta}_0}{\underline{y}_0^T \underline{y}_0} \underline{I}$$

To get B_{n+1}

- Davidon - Fletcher - Powell (DFP) Method

$$B_{n+1} = \left(I - \frac{y_n \delta_n^T}{y_n^T \delta_n} \right) B_n \left(I - \frac{\delta_n y_n^T}{y_n^T \delta_n} \right) + \frac{y_n y_n^T}{y_n^T y_n} \leftarrow$$

$$B_{n+1}^{-1} = B_n^{-1} + \frac{\delta_n \delta_n^T}{\delta_n^T y_n} - B_n^{-1} \left(\frac{y_n y_n^T}{y_n^T \underbrace{B_n^{-1} y_n}} \right) B_n^{-1}$$

You already know B_n^{-1}

- Broyden - Fletcher - Goldfarb - Shanno (BFGS) Method

Based on rank-1 updates

$$B_{n+1} = B_n + u u^T + v v^T \qquad u, v \text{ some vectors}$$

$$B_{n+1} = B_n + \frac{y_n y_n^T}{y_n^T \delta_n} - \frac{B_n \delta_n \delta_n^T B_n^T}{\delta_n^T B_n \delta_n}$$

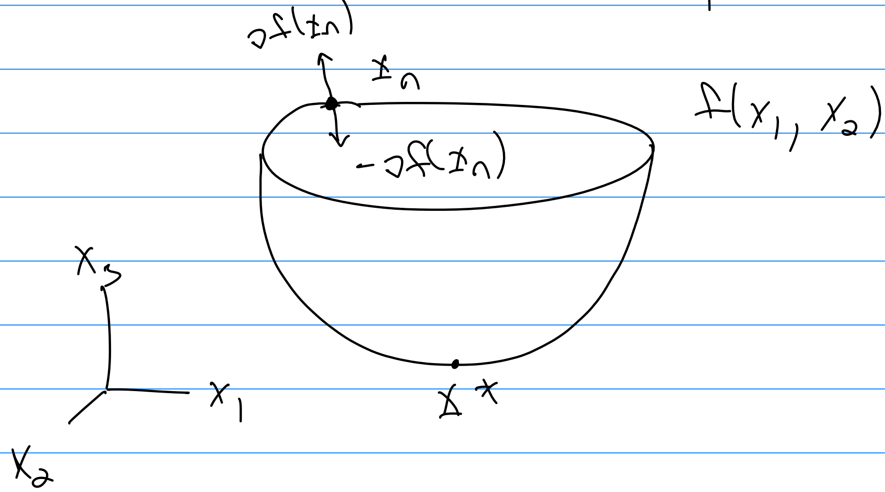
$$B_{n+1}^{-1} = \left(I - \frac{\delta_n y_n^T}{y_n^T \delta_n} \right) B_n^{-1} \left(I - \frac{y_n \delta_n^T}{y_n^T \delta_n} \right) + \frac{\delta_n \delta_n^T}{y_n^T \delta_n}$$

Others ...

Slower than full Newton but higher than order-1

Gradient Descent

Consider a convex 2D problem: $f(x_1, x_2)$



$$\underline{x}_{n+1} = \underline{x}_n - \alpha_n \nabla f(\underline{x}_n)$$

In 1D returns $x_{n+1} = x_n - \alpha_n \frac{\partial f(x_n)}{\partial x}$

Line Search / Backtracking

Many iterations of the type $\underline{x}_{n+1} = \underline{x}_n + \alpha_n \underline{\delta}_n$

$\underline{\delta}_n = -\underline{J}_n^{-1} \underline{f}_n$ (Newton)

$\underline{\delta}_n = -\underline{A}^{-1} \underline{f}(\underline{x}_n)$ (Fixed-Point)

$\underline{\delta}_n = -\nabla f(\underline{x}_n)$ Gradient Descent

$\underline{\delta}_n = -H_n^{-1} \nabla f(\underline{x}_n)$ (Newton)

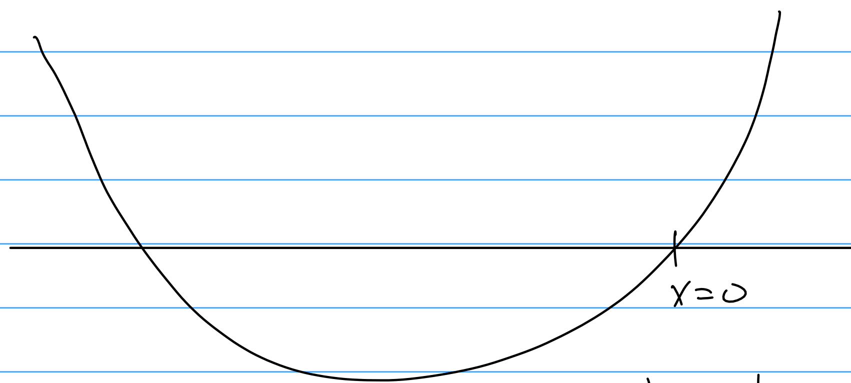
In all cases you want α_n such that

$f(\underline{x}_n + \alpha_n \underline{\delta}_n) < f(\underline{x}_n)$ or $\| \underline{g}(\underline{x}_n + \alpha_n \underline{\delta}_n) \| < \| \underline{g}(\underline{x}_n) \|$

minimization ... root finding

Focus on $f(\underline{x})$ (if root finding define $f(\underline{x}) = \| \underline{g}(\underline{x}) \|_2^2$)

It is not sufficient to simply require that $f(\underline{x}_n + \alpha_n \underline{\delta}_n) < f(\underline{x}_n) \leftarrow$



Algo states that $f(x_n) = \frac{5}{n}$

$\uparrow f^* = f(x^*) = -1$

We have $f(x_{n+1}) < f(x_n)$

but, it will never find x^*.

Converge to $x = 0$

Wolfe Conditions

These require that $\underline{x}_n + \alpha_n \underline{\delta}_n$ make sufficient progress.

Two Wolfe Conditions:

1) $f(\underline{x}_n + \alpha_n \underline{\delta}_n) \le f(\underline{x}_n) + c_1 \alpha_n (\nabla f_n)^T \underline{\delta}_n \quad c_1 \in (0, 1)$

c_1 is usually small, say $c_1 = 10^{-4}$

2) $[\nabla f(\underline{x}_n + \alpha_n \underline{\delta}_n)]^T \underline{\delta}_n \ge c_2 (\nabla f_n)^T \underline{\delta}_n$
for some $c_2 \in (c_1, 1)$
↑ make sure that α_n is not too small

$0 < c_1 < c_2 < 1$

It can be proven that if $f: \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, $\underline{\delta}_n$ is the descent direction @ $\underline{x}_n$, and if $0 < c_1 < c_2 < 1$, then there is a range of α_n that satisfies the Wolfe Conditions

Now, methods to find α_n.

1) $$\alpha_n = \frac{(\overbrace{\underline{x}_n - \underline{x}_{n-1}}^{\underline{\delta}_{n-1}})^T (\nabla f(\underline{x}_n) - \nabla f(\underline{x}_{n-1}))}{\| \nabla f(\underline{x}_n) - \nabla f(\underline{x}_{n-1}) \|_2^2}$$ to find $\underline{x}_{n+1}$

Issues: $\nabla f(\underline{x}_n)$ might be expensive
might not be "optimal"

2) Define $\phi(\alpha) = f(\underline{x}_n + \alpha \underline{\delta}_n)$

$$\phi'(\alpha) = \underline{\delta}_n^T \nabla f(\underline{x}_n + \alpha \underline{\delta}_n)$$

First Wolfe Condition: $\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'(0)$

Choose α_0 (usually $\alpha_0 = 1$)

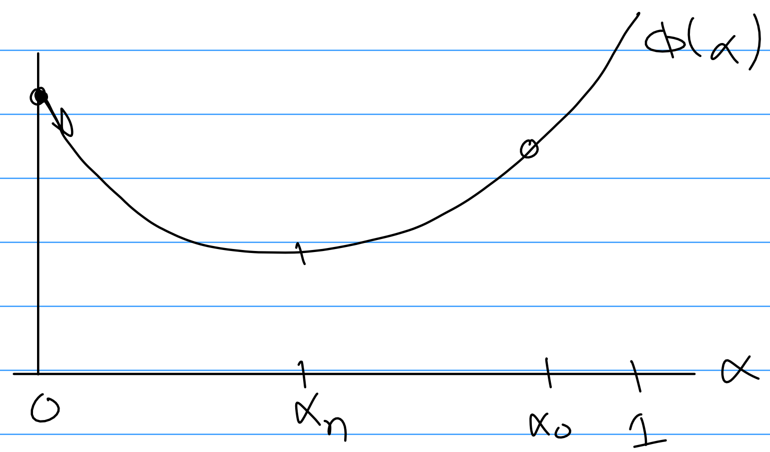
if $\phi(\alpha_0) \leq \phi(0) + c_1 \alpha \phi'(0)$

use α_0

If not true α is $\alpha \in (0, \alpha_0)$

Form interpolant using $\phi(0)$, $\phi'(0)$ & $\phi(\alpha)$

Aside:



$$\hat{\phi}(\alpha) = \left(\frac{\phi(\alpha_0) - \phi(0) - \alpha_0 \phi'(0)}{\alpha_0^2} \right) \alpha^2 + \phi'(0)\alpha + \phi(0)$$

minimize $\hat{\phi}(\alpha)$

$$\alpha_1 = - \frac{\phi'(0)\, \alpha_0^2}{2\left[\phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0\right]}$$

If $\phi(\alpha_1) \le \phi(0) + c_1 \alpha_1 \underline{\phi'(0)}$ use α_1

If it does not form a $\underline{\text{cubic}}$ interpolant

$$\phi(0),\ \phi'(0),\ \phi(\alpha_0),\ \phi(\alpha_1)$$

$$\tilde{\phi}(\alpha) = a\alpha^3 + b\alpha^2 + \phi'(0)\alpha + \phi(0)$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\alpha_0^2 \alpha_1^2 (\alpha_1 - \alpha_0)} \begin{bmatrix} \alpha_0^2 & -\alpha_1^2 \\ -\alpha_0^3 & \alpha_1^3 \end{bmatrix} \begin{bmatrix} \phi(\alpha_1) - \phi(0) - \alpha_1 \phi'(0) \\ \phi(\alpha_0) - \phi(0) - \alpha_0 \phi'(0) \end{bmatrix}$$

minimum of $\tilde{\phi}(\alpha)$ is $\alpha_2 = \dfrac{-b + \sqrt{b^2 - 3a\phi'(0)}}{3a}$

If $\phi(\alpha_2) \le \phi(0) + c_1 \alpha_2 \phi'(0)$ use α_2

Otherwise repeat w/ $\phi(0), \phi'(0), \phi(\alpha_{k-1}), \phi(\alpha_{k-2})$
to get α_k

Sequence of $\alpha_0 > \alpha_1 > \alpha_2 > \dots > \alpha_{k-1} > \alpha_{k-2}$

If $\alpha_{k-1} - \alpha_k < \varepsilon$ or if $\alpha_k \ll \alpha_{k-1}$
use $\alpha_k = \frac{1}{2}\alpha_{k-1}$ & continue