Scientific Computing: Numerical Optimization

Aleksandar Donev
Courant Institute, NYU[1]
donev@courant.nyu.edu

[1]Course MATH-GA.2043 or CSCI-GA.2112, Fall 2020

October 15th, 2020

Outline

1. Mathematical Background

2. Smooth Unconstrained Optimization

3. Equality Constrained Optimization

(In) — harder more interesting

4. Conclusions

Outline

Formulation

- Optimization problems are among the most important in engineering and finance, e.g., **minimizing** production cost, **maximizing** profits, etc.

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$$

  where $\mathbf{x}$ are some variable parameters and $f : \mathbb{R}^n \to \mathbb{R}$ is a scalar **objective function**.
- Observe that one only need to consider minimization as

$$\max_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = -\min_{\mathbf{x}\in\mathbb{R}^n} [-f(\mathbf{x})]$$

- A **local minimum $\mathbf{x}^\star$** is optimal in some neighborhood,

$$f(\mathbf{x}^\star) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \quad \text{s.t.} \quad \|\mathbf{x}-\mathbf{x}^\star\| \leq R > 0.$$

  (think of finding the bottom of a valley)
- Finding the **global minimum** is generally not possible for *arbitrary* functions (think of finding Mt. Everest without a satelite).

Connection to nonlinear systems

- Assume that the objective function is **differentiable** (i.e., first-order Taylor series converges or gradient exists).
- Then a **necessary condition** for a local minimizer is that $\mathbf{x}^\star$ be a **critical point**

$$\mathbf{g}\left(\mathbf{x}^\star\right) = \nabla_{\mathbf{x}} f\left(\mathbf{x}^\star\right) = \left\{\frac{\partial f}{\partial x_i}\left(\mathbf{x}^\star\right)\right\}_i = \mathbf{0}$$

$Ax = b$

$A^T A x = A^T b$

which is a system of non-linear equations!

- In fact similar methods, such as Newton or quasi-Newton, apply to both problems.
- Vice versa, observe that solving $\mathbf{f}\left(\mathbf{x}\right) = \mathbf{0}$ is equivalent to an optimization problem

$$\min_{\mathbf{x}} \left[\mathbf{f}\left(\mathbf{x}\right)^T \mathbf{f}\left(\mathbf{x}\right)\right]$$

why

although this is only recommended under special circumstances.

Sufficient Conditions

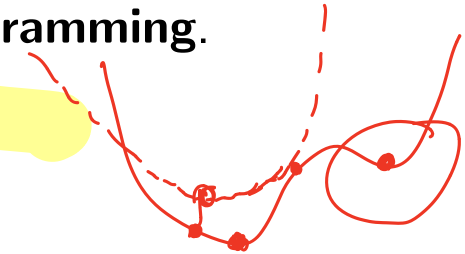
- Assume now that the objective function is **twice-differentiable** (i.e., Hessian exists).
- A critical point $\mathbf{x}^{\star}$ is a local minimum if the **Hessian is positive definite**

$$\mathbf{H}\left(\mathbf{x}^{\star}\right)=\boldsymbol{\nabla}_{\mathbf{x}}^{2} f\left(\mathbf{x}^{\star}\right) \succ \mathbf{0}$$

  which means that the minimum really looks like a valley or a **convex** bowl.
- At any local minimum the Hessian is positive **semi-definite**, $\boldsymbol{\nabla}_{\mathbf{x}}^{2} f\left(\mathbf{x}^{\star}\right) \succeq \mathbf{0}$.
- Methods that require Hessian information converge fast but are expensive.

Mathematical Programming

- The general term used is **mathematical programming**.
- Simplest case is **unconstrained optimization**



$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

where $\mathbf{x}$ are some variable parameters and $f : \mathbb{R}^n \to \mathbb{R}$ is a scalar **objective function**.

  - Find a **local minimum** $\mathbf{x}^\star$:

$$f(\mathbf{x}^\star) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \quad \text{s.t.} \quad \|\mathbf{x} - \mathbf{x}^\star\| \leq R > 0.$$

  (think of finding the bottom of a valley).

  - Find the best local minimum, i.e., the **global minimumx***: This is virtually impossible in general and there are many specialized techniques such as **genetic programming**, **simmulated annealing**, **branch-and-bound** (e.g., using interval arithmetic), etc.

- Special case: A **strictly convex objective function** has a unique local minimum which is thus also the global minimum.

Constrained Programming

- The most general form of **constrained optimization**

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

where $\mathcal{X} \subset \mathbb{R}^n$ is a **set of feasible solutions**.

- The feasible set is usually expressed in terms of **equality and inequality constraints**:

$$\left\{\begin{array}{l} \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \end{array}\right. \qquad \forall x \in \mathcal{X}$$

- The only generally solvable case: **convex programming**
Minimizing a convex function $f(\mathbf{x})$ over a convex set $\mathcal{X}$: every local minimum is global.
If $f(\mathbf{x})$ is strictly convex then there is a **unique local and global minimum**.

Special Cases

- Special case of convex programming is **linear programming**:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \left\{ \mathbf{c}^T \mathbf{x} \right\}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}.$$

solution is a vertex of polytope

- The feasible set here is a convex **polytope** (polygon, polyhedron) in $\mathbb{R}^n$, consider for now the case when it is bounded, meaning there are at least $n+1$ constraints.
- The optimal point is a **vertex** of the polyhedron, meaning a point where (generically) n constraints are **active**,

$$\mathbf{A}_{act}\mathbf{x}^{\star} = \mathbf{b}_{act}.$$

- Solving the problem therefore means finding the subset of **active constraints**:
  Combinatorial search problem, solved using the **simplex algorithm** (search along the edges of the polytope).

Outline

Necessary and Sufficient Conditions

- A **necessary condition** for a local minimizer:
  The optimum $\mathbf{x}^\star$ must be a **critical point (maximum, minimum or saddle point)**:

$$\mathbf{g}(\mathbf{x}^\star) = \boldsymbol{\nabla}_{\mathbf{x}} f(\mathbf{x}^\star) = \left\{ \frac{\partial f}{\partial x_i}(\mathbf{x}^\star) \right\}_i = \mathbf{0},$$

  and an additional **sufficient condition** for a critical point $\mathbf{x}^\star$ to be a local minimum:
  The Hessian at the optimal point must be **positive definite,**

$$\mathbf{H}(\mathbf{x}^\star) = \boldsymbol{\nabla}^2_{\mathbf{x}} f(\mathbf{x}^\star) = \left\{ \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}^\star) \right\}_{ij} \succ \mathbf{0}.$$

  which means that the minimum really looks like a valley or a **convex** bowl.

Direct-Search Methods

"black-box" optimization

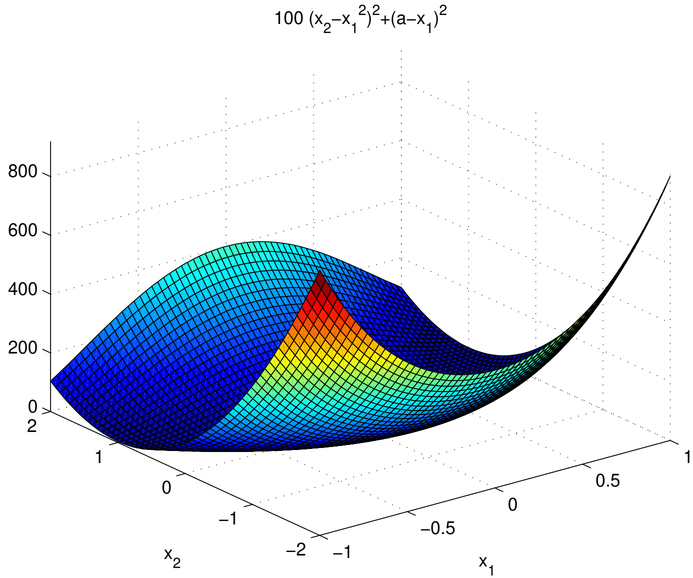
- A **direct search method** only requires $f(\mathbf{x})$ to be **continuous** but not necessarily differentiable, and requires only **function evaluations**.
- Methods that do a search similar to that in bisection can be devised in higher dimensions also, but they may fail to converge and are usually slow.
- The MATLAB function *fminsearch* uses the Nelder-Mead or **simplex-search** method, which can be thought of as rolling a simplex downhill to find the bottom of a valley. But there are many others and this is an active research area.
- **Curse of dimensionality**: As the number of variables (dimensionality) n becomes larger, direct search becomes hopeless since the number of samples needed grows as 2^n!

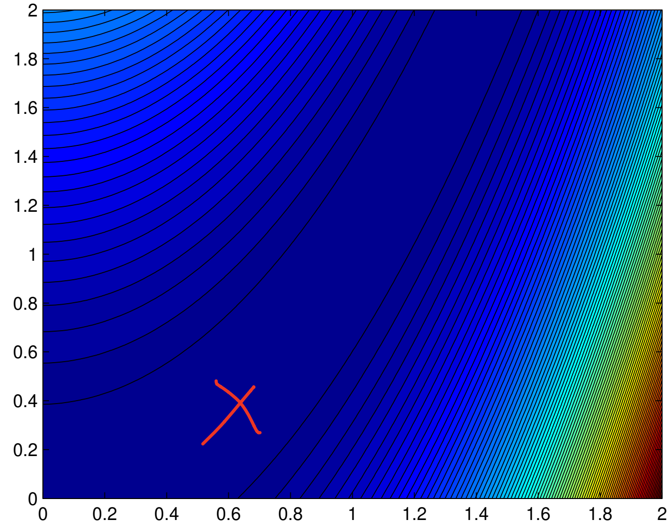
Minimum of $100(x_2 - x_1^2)^2 + (a - x_1)^2$ in MATLAB

```
% Rosenbrock or 'banana' function:
a = 1;
banana = @(x) 100*(x(2)-x(1)^2)^2+(a-x(1))^2;

% This function must accept array arguments!
banana_xy = @(x1,x2) 100*(x2-x1.^2).^2+(a-x1).^2;

[x,y] = meshgrid(linspace(0,2,100));
figure(1); ezsurf(banana_xy, [0,2,0,2])
figure(2); contourf(x,y,banana_xy(x,y),100)

% Correct answers are x=[1,1] and f(x)=0
[x,fval] = fminsearch(banana, [-1.2, 1], ...
                      optimset('TolX',1e-8))
x =     0.999999999187814    0.999999998441919
fval =        1.099088951919573e-18
```

(linspace)

Figure of Rosenbrock $f(\mathbf{x})$





Descent Methods

- Finding a local minimum is generally **easier** than the general problem of solving the non-linear equations

$$\mathbf{g}(\mathbf{x}^{\star}) = \boldsymbol{\nabla}_{\mathbf{x}} f(\mathbf{x}^{\star}) = \mathbf{0}$$

  - We can evaluate f in addition to $\boldsymbol{\nabla}_{\mathbf{x}} f$.
  - The Hessian is positive-(semi)definite near the solution (enabling simpler linear algebra such as Cholesky).

- If we have a current guess for the solution $\mathbf{x}^k$, and a **descent direction** (i.e., **downhill** direction) $\mathbf{d}^k$:

$$f\left(\mathbf{x}^k + \alpha \mathbf{d}^k\right) < f\left(\mathbf{x}^k\right) \text{ for all } 0 < \alpha \leq \alpha_{max},$$

  then we can move downhill and get closer to the minimum (valley):

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k,$$

  where $\alpha_k > 0$ is a **step length**.

Gradient Descent Methods

- For a differentiable function we can use Taylor's series:

$$f\left(\mathbf{x}^k + \alpha \mathbf{d}^k\right) \approx f\left(\mathbf{x}^k\right) + \alpha_k \left[(\boldsymbol{\nabla} f)^T \mathbf{d}^k \right]$$

$-g_k^T g_k$

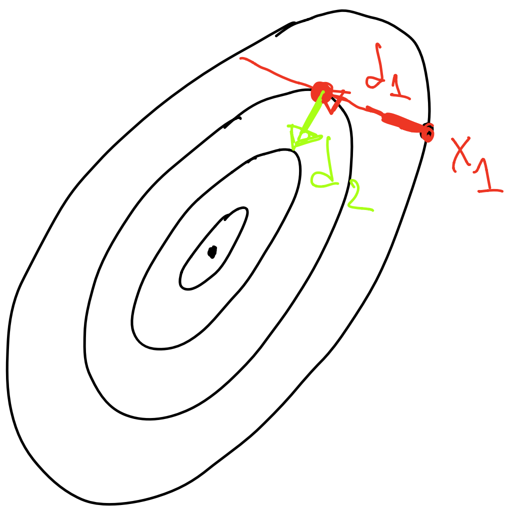
- This means that **fastest local decrease** in the objective is achieved when we move opposite of the gradient: **steepest or gradient descent**:

$$\mathbf{d}^k = -\boldsymbol{\nabla} f\left(\mathbf{x}^k\right) = -\mathbf{g}_k.$$

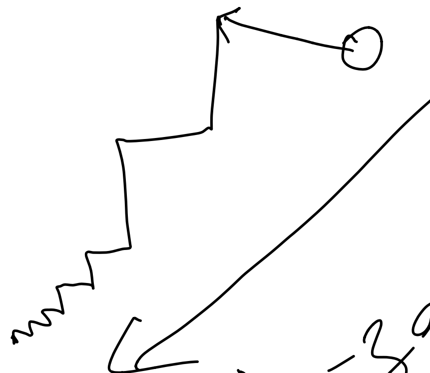
- One option is to choose the step length using a **line search** one-dimensional minimization:

$$\alpha_k = \arg\min_{\alpha} f\left(\mathbf{x}^k + \alpha \mathbf{d}^k\right),$$

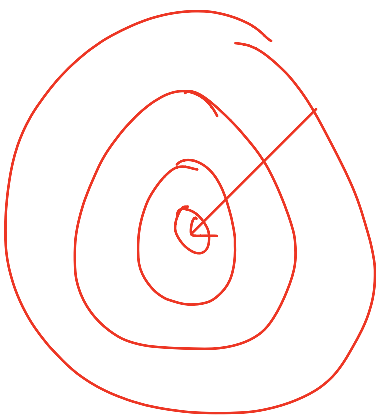
which needs to be solved **only approximately**, see **Wolfe conditions** on **inexact line search** in Wikipedia for details.



$d_1 \perp d_2$



zig-zag



Steepest Descent

- Assume an exact line search was used, i.e., $\alpha_k = \arg\min_\alpha \phi(\alpha)$ where

$$\phi(\alpha) = f\left(\mathbf{x}^k + \alpha \mathbf{d}^k\right).$$

$\nabla f^{k+1} \perp \nabla f^k$

$$\phi'(\alpha) = 0 = \left[\nabla f\left(\overbrace{\mathbf{x}^k + \alpha \mathbf{d}^k}^{x^{k+1}}\right)\right]^T \mathbf{d}^k.$$

- This means that steepest descent takes **a zig-zag path** down to the minimum.
- Second-order analysis shows that steepest descent has **linear convergence** with convergence coefficient

If $r \ll 1$

$$C \sim \frac{1-r}{1+r} < 1, \text{ where } r = \frac{\lambda_{min}(\mathbf{H})}{\lambda_{max}(\mathbf{H})} = \frac{1}{\kappa_2(\mathbf{H})},$$

$C \approx 1$

inversely proportional to the **condition number** of the Hessian.

- Steepest descent can be very slow for ill-conditioned Hessians: One improvement is to use **conjugate-gradient method instead**.

Newton's Method

quadratic convergence

- Making a second-order or quadratic model of the function:

$$f(\mathbf{x}^k + \Delta\mathbf{x}) = f(\mathbf{x}^k) + \left[\mathbf{g}\left(\mathbf{x}^k\right)\right]^T (\Delta\mathbf{x}) + \frac{1}{2}(\Delta\mathbf{x})^T \left[\mathbf{H}\left(\mathbf{x}^k\right)\right](\Delta\mathbf{x})$$

we obtain **Newton's method**:

$$H^k \Delta x^k = -g^k$$

$$\mathbf{g}(\mathbf{x} + \Delta\mathbf{x}) = \nabla f(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{0} = \mathbf{g} + \mathbf{H}(\Delta\mathbf{x}) \quad \Rightarrow$$

$$\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g} \quad \Rightarrow \quad \mathbf{x}^{k+1} = \mathbf{x}^k - \left[\mathbf{H}\left(\mathbf{x}^k\right)\right]^{-1}\left[\mathbf{g}\left(\mathbf{x}^k\right)\right].$$

- Note that this is identical to using the Newton-Raphson method for solving the nonlinear system $\nabla_{\mathbf{x}} f\left(\mathbf{x}^\star\right) = \mathbf{0}$.
- At the minimum $\mathbf{H}\left(\mathbf{x}^\star\right) \succ \mathbf{0}$ so one can use **Cholesky factorization** to compute $\left[\mathbf{H}\left(\mathbf{x}^k\right)\right]^{-1}\left[\mathbf{g}\left(\mathbf{x}^k\right)\right]$ sufficiently close to the minimum.

Problems with Newton's Method

- Newton's method is **exact for a quadratic function** (this is another way to define order of convergence!) and converges in one step when $\mathbf{H} \equiv \mathbf{H}\left(\mathbf{x}^k\right) = \text{const}$.
- For non-linear objective functions, however, Newton's method requires solving a linear system every step: **expensive**.
- It may not converge at all if the initial guess is not very good, or may converge to a saddle-point or maximum: **unreliable**. not robust
- All of these are addressed by using variants of **quasi-Newton** and **trust-region methods**:

$B^k d^k = -g^k$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k = \mathbf{x}^k - \alpha_k \left(\mathbf{B}^k\right)^{-1} \mathbf{g}\left(\mathbf{x}^k\right),$$

where the **step length** $0 < \alpha_k < 1$ and $\mathbf{B}^k$ is an **approximation** to the true Hessian.

Quasi-Newton Methods

- The approximation of the Hessian in quasi-Newton methods is built using **low-rank updates** (recall Woodbury formula from Homework 2) to estimate the Hessian using finite differences with a small cost per step.
- The Hessian estimate satisfies the **secant condition**

finite difference in higher dimensions

If f quadratic exact if B=H

$$\mathbf{g}\left(\mathbf{x}^{k+1}\right)-\mathbf{g}\left(\mathbf{x}^{k}\right)=\mathbf{y}^{k}=\mathbf{B}^{k+1}\Delta\mathbf{x}^{k}.$$

- A popular **rank-2** update of the Hessian is the Broyden–Fletcher–Goldfarb–Shanno **(BFGS) algorithm**:

$$\mathbf{B}^{k+1}=\mathbf{B}^{k}+\underbrace{\frac{\mathbf{y}^{k}\left(\mathbf{y}^{k}\right)^{T}}{\left(\mathbf{y}^{k}\right)^{T}\Delta\mathbf{x}^{k}}}_{\text{rank-1}}-\underbrace{\frac{\mathbf{z}^{k}\left(\mathbf{z}^{k}\right)^{T}}{\left(\mathbf{z}^{k}\right)^{T}\Delta\mathbf{x}^{k}}}_{\text{rank-1}},$$

where $\mathbf{z}^{k}=\mathbf{B}^{k}\Delta\mathbf{x}^{k}$.

- This update is symmetric and with careful line search it **ensures that the Hessian estimate remains symmetric positive semi-definite** so Cholesky factorization (or conjugate gradient) can be used.

Outline

Penalty Approach

- The idea is the convert the constrained optimization problem:

$$\begin{array}{ll} & \min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}) = \mathbf{0} \end{array} \quad . \Big\} \text{ PDE}$$

  into an unconstrained optimization problem.

- Consider minimizing the **penalized function**

Lagrangian

$$\mathcal{L}_\alpha(\mathbf{x}) = f(\mathbf{x}) + \alpha \|\mathbf{h}(\mathbf{x})\|_2^2 = f(\mathbf{x}) + \alpha \left[\mathbf{h}(\mathbf{x})\right]^T \left[\mathbf{h}(\mathbf{x})\right],$$

  where $\alpha > 0$ is a **penalty parameter**.

- Note that one can use **penalty functions** other than sum of squares.
- If the constraint is exactly satisfied, then $\mathcal{L}_\alpha(\mathbf{x}) = f(\mathbf{x})$.
  As $\alpha \to \infty$ violations of the constraint are penalized more and more, so that the equality will be satisfied with higher accuracy.

Penalty Method

- The above suggest the **penalty method** (see homework):
  For a monotonically diverging sequence $\alpha_1 < \alpha_2 < \cdots$, solve a **sequence of unconstrained problems**

$$\mathbf{x}^k = \mathbf{x}(\alpha_k) = \arg\min_{\mathbf{x}} \left\{ \mathcal{L}_k(\mathbf{x}) = f(\mathbf{x}) + \alpha_k \left[\mathbf{h}(\mathbf{x})\right]^T \left[\mathbf{h}(\mathbf{x})\right] \right\}$$

  and the solution should converge to the optimum $\mathbf{x}^\star$,

$$\mathbf{x}^k \to \mathbf{x}^\star = \mathbf{x}(\alpha_k \to \infty).$$

- Note that one can use $\mathbf{x}^{k-1}$ as an **initial guess** for, for example, Newton's method.
- Also note that the problem becomes more and more **ill-conditioned** as α grows.
  A better approach uses Lagrange multipliers in addition to penalty (**augmented Lagrangian**).

Outline

Conclusions/Summary

- Optimization, or **mathematical programming**, is one of the most important numerical problems in practice.
- Optimization problems can be **constrained** or **unconstrained**, and the nature (linear, convex, quadratic, algebraic, etc.) of the functions involved matters.
- Finding a **global minimum** of a general function is virtually **impossible** in high dimensions, but very important in practice.
- An unconstrained local minimum can be found using **direct search**, **gradient descent**, or **Newton-like methods**.
- Equality-constrained optimization is **tractable**, but the best method **depends on the specifics**.
- Constrained optimization is tractable for the convex case, otherwise often hard, and even **NP-complete** for **integer programming**.