Zero-Order Optimization Algorithms

Yinyu Ye
MS&E and ICME, Stanford University
https://web.stanford.edu/class/msande314/handout.shtml

http://www.stanford.edu/~yyye

(Chapters 7 and 8)

Introduction

Optimization algorithms tend to be iterative procedures. Starting from a given point $\mathbf{x}^0$, they generate a sequence $\{\mathbf{x}^k\}$ of iterates (or trial solutions) that converge to a "solution" – or at least they are designed to be so.

Recall that scalars $\{x^k\}$ converges to 0 if and only if for all real numbers $\varepsilon > 0$ there exists a positive integer K such that

$$|x^k| < \varepsilon \quad \text{for all } k \geq K.$$

Then $\{\mathbf{x}^k\}$ converges to solution $\mathbf{x}^*$ if and only if $\{\|\mathbf{x}^k - \mathbf{x}^*\|\}$ converges to 0.

We study algorithms that produce iterates according to

- well determined rules–Deterministic Algorithm
- random selection process–Randomized Algorithm.

The rules to be followed and the procedures that can be applied depend to a large extent on the characteristics of the problem to be solved.

The Meaning of "Solution"

What is meant by a solution may differ from one algorithm to another.

In some cases, one seeks a local minimum; in some cases, one seeks a global minimum; in others, one seeks a first-order and/or second-order stationary or KKT point of some sort as in the method of steepest descent discussed below.

In fact, there are several possibilities for defining what a solution is. Once the definition is chosen, there must be a way of testing whether or not an iterate (trial solution) belongs to the set of solutions. For example, the residuals of the KKT conditions converge to zero.

Generic Algorithms for Minimization and Global Convergence Theorem

A Generic Algorithm: A point to set mapping in a subspace of R^n.

Theorem 1 *(Page 222, L&Y) Let A be an "algorithmic mapping" defined over set X, and let sequence $\{\mathbf{x}^k\}$, starting from a given point $\mathbf{x}^0$, be generated from*

$$\mathbf{x}^{k+1} \in A(\mathbf{x}^k).$$

Let a solution set $S \subset X$ be given, and suppose

i) all points $\{\mathbf{x}^k\}$ are in a compact set;

ii) there is a continuous (merit) function $z(\mathbf{x})$ such that if $\mathbf{x} \notin S$, then $z(\mathbf{y}) < z(\mathbf{x})$ for all $\mathbf{y} \in A(\mathbf{x})$; otherwise, $z(\mathbf{y}) \leq z(\mathbf{x})$ for all $\mathbf{y} \in A(\mathbf{x})$;

iii) the mapping A is closed at points outside S ($\mathbf{x}^k \to \bar{\mathbf{x}} \in X$ and $A(\mathbf{x}^k) = \mathbf{y}^k \to \bar{\mathbf{y}}$ imply $\bar{\mathbf{y}} \in A(\bar{\mathbf{x}})$).

Then, the limit of any convergent subsequences of $\{\mathbf{x}^k\}$ is a solution in S.

Descent Direction Methods

In this case, merit function $z(\mathbf{x}) = f(\mathbf{x})$, that is, just the objective itself.

(A1) Test for convergence If the termination conditions are satisfied at $\mathbf{x}^k$, then it is taken (accepted) as a "solution." In practice, this may mean satisfying the desired conditions to within some tolerance. If so, stop. Otherwise, go to step (A2).

(A2) Compute a search direction, say $\mathbf{d}^k \neq \mathbf{0}$. This might be a direction in which the function value is known to decrease within the feasible region.

(A3) Compute a step length, say α^k such that

$$f(\mathbf{x}^k + \alpha^k \mathbf{d}^k) < f(\mathbf{x}^k).$$

This may necessitate a one-dimensional (or line) search.

(A4) Define the new iterate by setting

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

and return to step (A1).

Algorithm Complexity and Speeds I

The intrinsic computational cost/time of an algorithm depends on

- number of decision variables n: cost of the inner product of two vectors, cost of solving system of linear equations
- number of constraints m: cost of the product of a matrix and a vector, cost of the product of two matrices
- number of nonzero data entries NNZ: sparse matrix/data representation
- the desired accuracy $0\epsilon < 1$: the cost could be propotional to $\frac{1}{\epsilon^2},\ \frac{1}{\epsilon},\ \log(\frac{1}{\epsilon}),\ \log[\log(\frac{1}{\epsilon})], ...$
- problem difficulty or complexity measures such as the Lipschiz constant β, the condition number of a matrix, etc

Algorithm Complexity and Speeds II

- Finite versus infinite convergence. For some classes of optimization problems there are algorithms that obtain an exact solution—or detect the unboundedness–in a finite number of iterations.

- Polynomial-time versus exponential-time. The solution time grows, in the worst-case, as a function of problem sizes (number of variables, constraints, accuracy, etc.).

- Convergence order and rate. If there is a positive number γ such that

$$\|\mathbf{x}^k - \mathbf{x}^*\| \leq \frac{O(1)}{k^\gamma}\|\mathbf{x}^0 - \mathbf{x}^*\|,$$

  then $\{\mathbf{x}^k\}$ converges arithmetically to $\mathbf{x}^*$ with power γ. If there exists a number $\gamma \in [0, 1)$ such that

$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\| \leq \gamma\|\mathbf{x}^k - \mathbf{x}^*\| \quad (\Rightarrow \|\mathbf{x}^k - \mathbf{x}^*\| \leq \gamma^k\|\mathbf{x}^0 - \mathbf{x}^*\|),$$

  then $\{\mathbf{x}^k\}$ converges geometrically or linearly to $\mathbf{x}^*$ with rate γ. If there exists a number $\gamma \in [0, 1)$

$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\| \leq \gamma\|\mathbf{x}^k - \mathbf{x}^*\|^2 \text{ after } \gamma\|\mathbf{x}^k - \mathbf{x}^*\| < 1$$

  then $\{\mathbf{x}^k\}$ converges quadratically to $\mathbf{x}^*$ (such as $\left\{(\frac{1}{2})^{2^k}\right\}$).

Algorithm Classes

Depending on information of the problem being used to create a new iterate, we have

(a) Zero-order algorithms. Popular when the gradient and Hessian information are difficult to obtain, e.g., no explicit function forms are given, functions are not differentiable, etc.

(b) First-order algorithms. Most popular now-days, suitable for large scale data optimization with low accuracy requirement, e.g., Machine Learning, Statistical Predictions...

(c) Second-order algorithms. Popular for optimization problems with high accuracy need, e.g., some scientific computing, etc.

One-Variable Optimization: Golden Section (Zero Order) Method

Assume that the one variable function $f(x)$ is Unimodel in interval $[a\ b]$, that is, for any point $x \in [a_r\ b_l]$ such that $a \leq a_r < b_l \leq b$, we have that $f(x) \leq \max\{f(a_r),\ f(b_l)\}$. How do we find x^* within an error tolerance ϵ?

0) Initialization: let $x_l = a,\ x_r = b$, and choose a constant $0 < r < 0.5$;

1) Let two other points $\hat{x}_l = x_l + r(x_r - x_l)$ and $\hat{x}_r = x_l + (1-r)(x_r - x_l)$, and evaluate their function values.

2) Update the triple points $x_r = \hat{x}_r, \hat{x}_r = \hat{x}_l, x_l = x_l$ if $f(\hat{x}_l) < f(\hat{x}_r)$; otherwise update the triple points $x_l = \hat{x}_l, \hat{x}_l = \hat{x}_r, x_r = x_r$; and return to Step 1.

In either cases, the length of the new interval after one golden section step is $(1-r)$. If we set $(1-2r)/(1-r) = r$, then only one point is new in each step and needs to be evaluated. This give $r = 0.382$ and the linear convergence rate is 0.618.

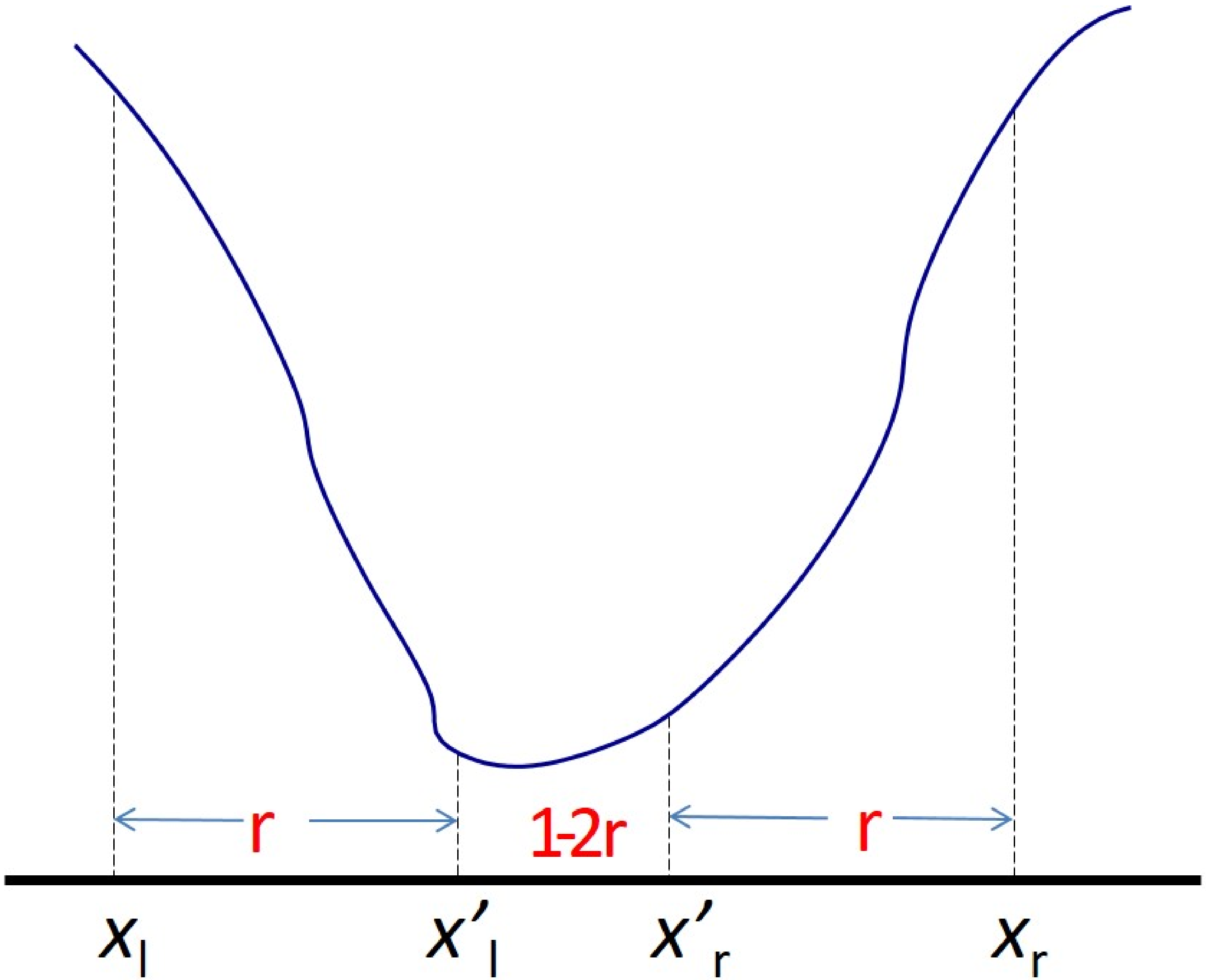

Figure 1: Illustration of Golden Section

One-Variable Optimization: Bisection (First Order) Method

For a one variable problem, an KKT point is the root of $g(x) := f'(x) = 0$.

Assume we know an interval $[a\ b]$ such that $a < b$, and $g(a)g(b) < 0$. Then we know there exists an x^*, $a < x^* < b$, such that $g(x^*) = 0$; that is, interval $[a\ b]$ contains a root of g. How do we find x within an error tolerance ϵ, that is, $|x - x^*| \leq \epsilon$?

0) Initialization: let $x_l = a,\ x_r = b$.

1) Let $x_m = (x_l + x_r)/2$, and evaluate $g(x_m)$.

2) If $g(x_m) = 0$ or $x_r - x_l < \epsilon$ stop and output $x^* = x_m$. Otherwise, if $g(x_l)g(x_m) > 0$ set $x_l = x_m$; else set $x_r = x_m$; and return to Step 1.

The length of the new interval containing a root after one bisection step is $1/2$ which gives the linear convergence rate is $1/2$, and this establishes a linear convergence rate 0.5.

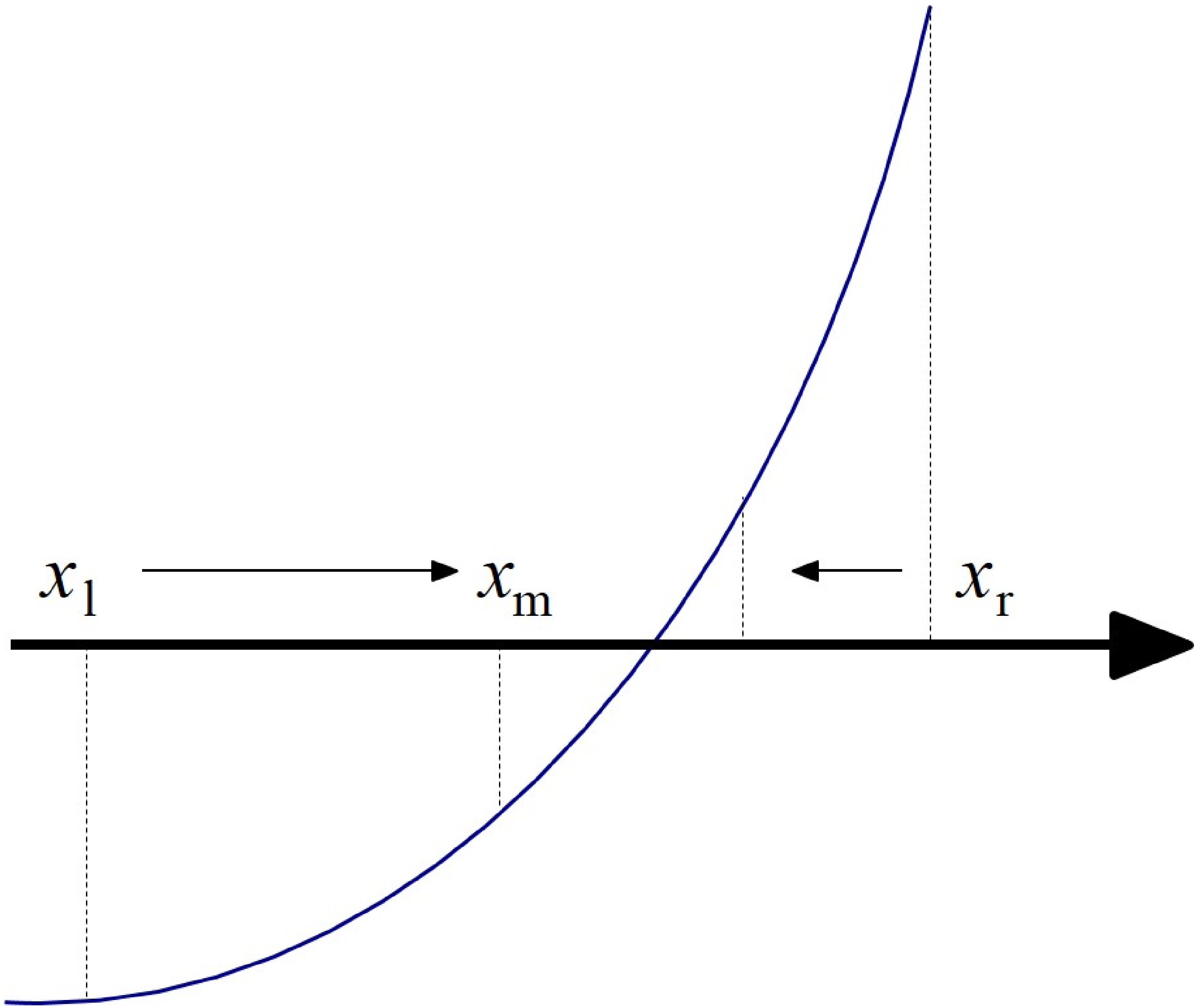

Figure 2: Illustration of Bisection

One-Variable Optimization: Newton's (Second Order) Method

For functions of a single real variable x, the KKT condition is $g(x) := f'(x) = 0$. When f is twice continuously differentiable then g is once continuously differentiable, Newton's method can be a very effective way to solve such equations and hence to locate a root of g. Given a starting point x^0, Newton's method for solving the equation $g(x) = 0$ is to generate the sequence of iterates

$$x^{k+1} = x^k - \frac{g(x^k)}{g'(x^k)}.$$

The iteration is well defined provided that $g'(x^k) \neq 0$ at each step.

For strictly convex function, Newton's method has a linear convergence rate and, when the point is "close" to the root, the convergence becomes quadratic, which leads to the iterations bound of $\log[\log(\frac{1}{\epsilon})]$.

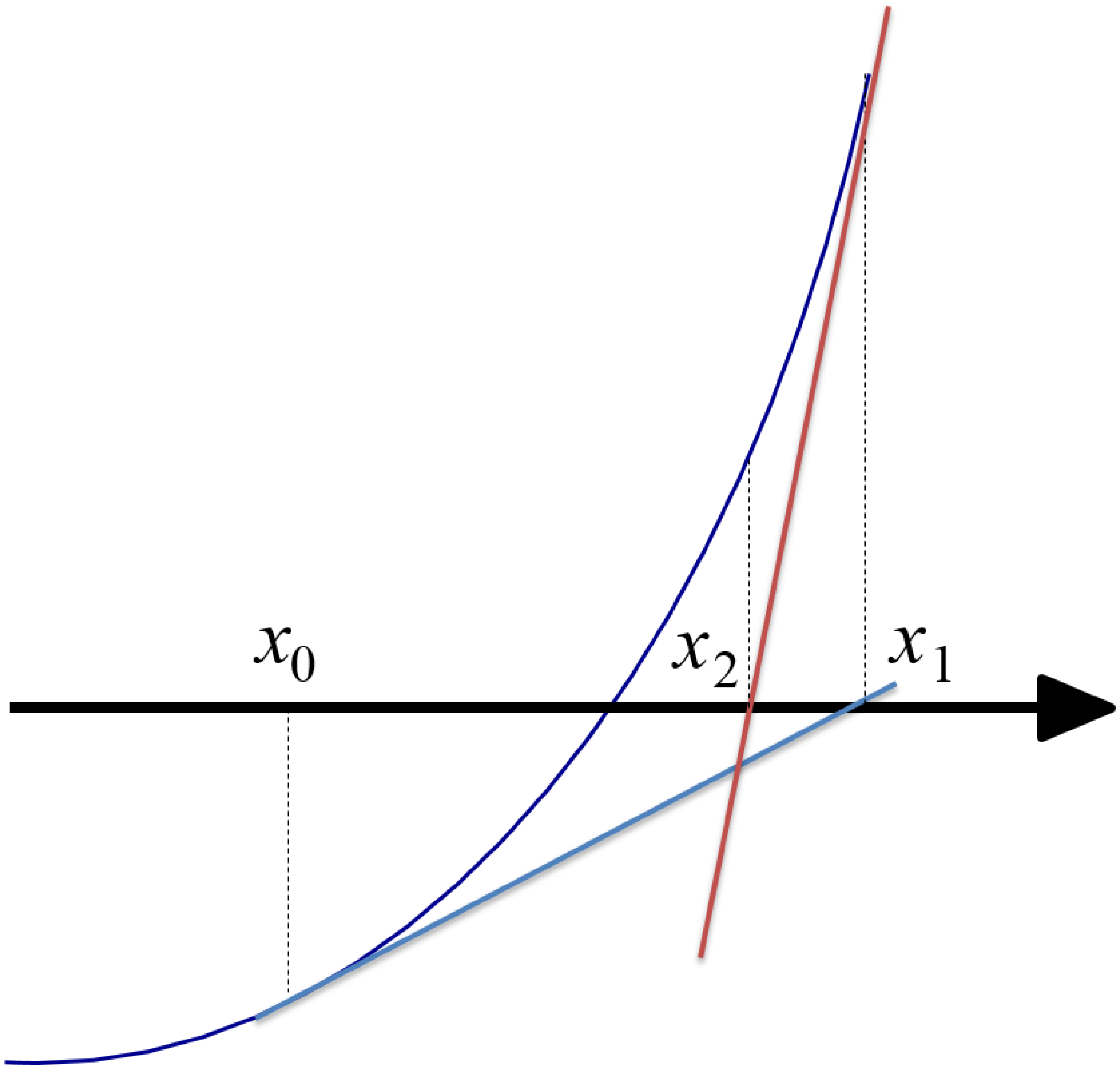

Figure 3: Illustration of Newton's Method

How Close is Close: One-variable Criterion

Theorem 2 *(Smale 86). Let $g(x)$ be an analytic function. Then, if x in the domain of g satisfies*

$$\sup_{k>1} \left| \frac{g^{(k)}(x)}{k!g'(x)} \right|^{1/(k-1)} \leq (1/8) \left| \frac{g'(x)}{g(x)} \right|.$$

Then, x is an approximate root of g.

In the following, for simplicity, let the root be in interval $[0 \;\; R]$.

Corollary 1 *(Y. 92). Let $g(x)$ be an analytic function in R^{++} and let g be convex and monotonically decreasing. Furthermore, for $x \in R^{++}$ and $k > 1$ let*

$$\left| \frac{g^{(k)}(x)}{k!g'(x)} \right|^{1/(k-1)} \leq \frac{\alpha}{8} \cdot \frac{1}{x}$$

for some constant $\alpha > 0$. Then, if the root $\bar{x} \in [\hat{x}, (1 + 1/\alpha)\hat{x}] \subset R^{++}$, $\hat{x}$ is an approximate root of g.

Hybrid of Bisection and Newton I

Note that the interval becomes wider and wider at geometric rate when $\hat{x}$ is increased.

Thus, we may symbolically construct a sequence of points:

$$\hat{x}_0 = \epsilon,\ \hat{x}_1 = (1 + 1/\alpha)\hat{x}_0, ..., \text{ and } \hat{x}_j = (1 + 1/\alpha)\hat{x}_{j-1}, ...$$

until $\hat{x}_j = \hat{x}_J \geq R$. Obviously the total number of points, J, of these points is bounded by $O(\log(R/\epsilon))$. Moreover, define a sequence of intervals

$$I_j = [\hat{x}_{j-1}, \hat{x}_j] = [\hat{x}_{j-1}, (1 + 1/\alpha)\hat{x}_{j-1}].$$

Then, if the root $\bar{x}$ of g is in any one of these intervals, say in I_j, then the front point $\hat{x}_{j-1}$ of the interval is an approximate root of g so that starting from it Newton's method generates an x with $|x - \bar{x}| \leq \epsilon$ in $O(\log\log(1/\epsilon))$ iterations.

Hybrid of Bisection and Newton II

Now the question is how to identify the interval that contains $\bar{x}$?

This time, we bisect the number of intervals, that is, evaluate function value at point $\hat{x}_{j_m}$ where $j_m = [J/2]$. Thus, each bisection reduces the total number of the intervals by a half. Since the total number of intervals is $O(\log(R/\epsilon))$, in at most $O(\log\log(R/\epsilon))$ bisection steps we shall locate the interval that contains $\bar{x}$.

Then the total number iterations, including both bisection and Newton methods, is $O(\log\log(R/\epsilon))$ iterations.

Here we take advantage of the global convergence property of Bisection and local quadratic convergence property of Newton, and we would see more of these features later...

Multi-Variable Optimization Zero-Order Algorithms: the "Simplex" Method

(1) Start with a Simplex with $d+1$ corner points and their objective function values.

(2) Reflection: Compute other $d+1$ corner points each of them is an additional corner point of a reflection simplex. If a point is better than its counter point, then the reflection simplex is an improved simplex, and select the most improved simplex and go to Step1; otherwise go to Step 3.

(3) Contraction: Compute the $d+1$ middle-face points and subdivide the simplex into smaller $d+1$ simplexes, keep the simplex with the lowest sum of the $d+1$ function values, and go to Step 1.

This method can be also implemented with exhausted enumeration in parallel. The method is suitable for solving problems whose derivatives are difficult to compute.

How to generate the initial $d+1$ points?

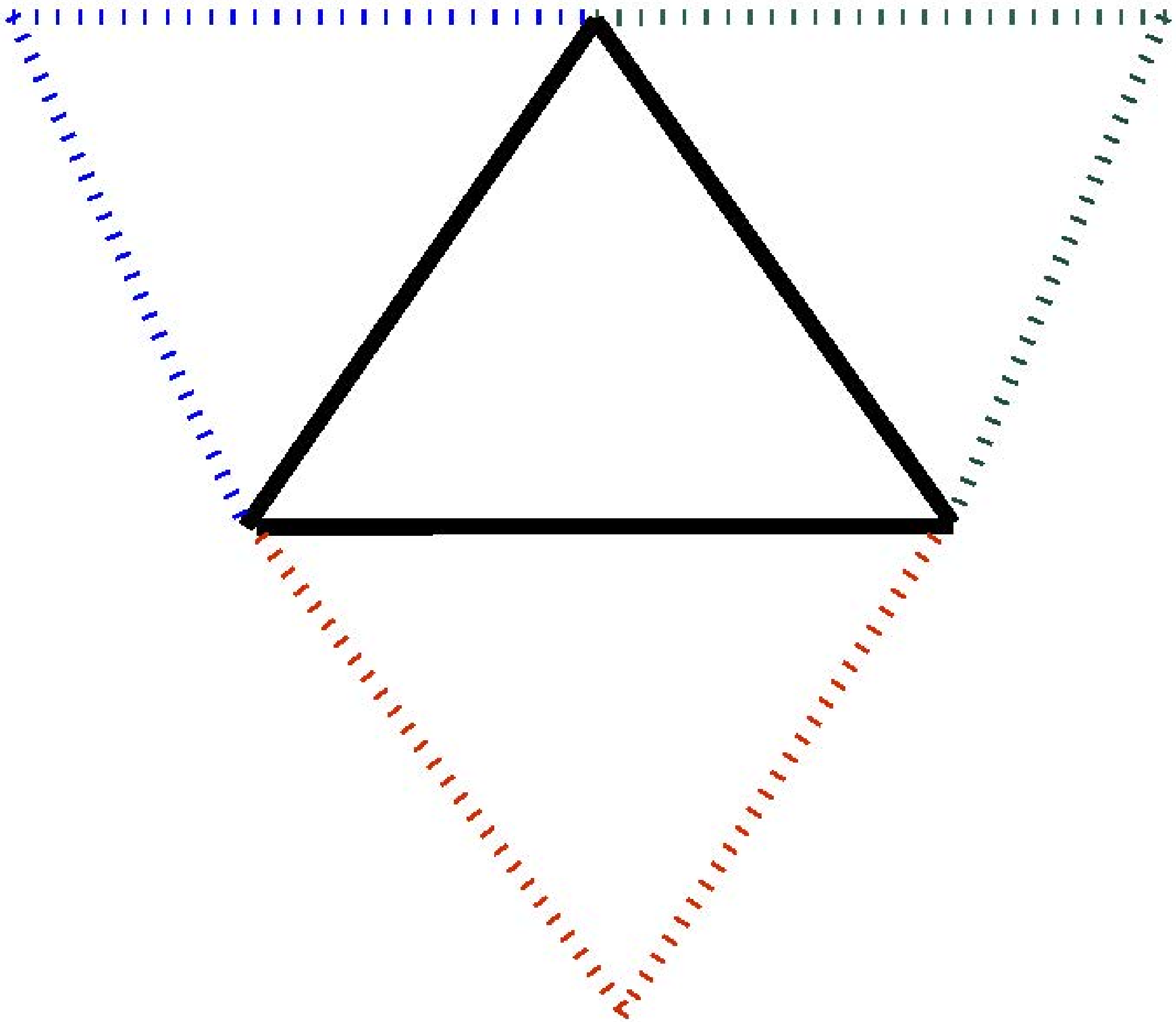

Figure 4: Reflection Simplexes

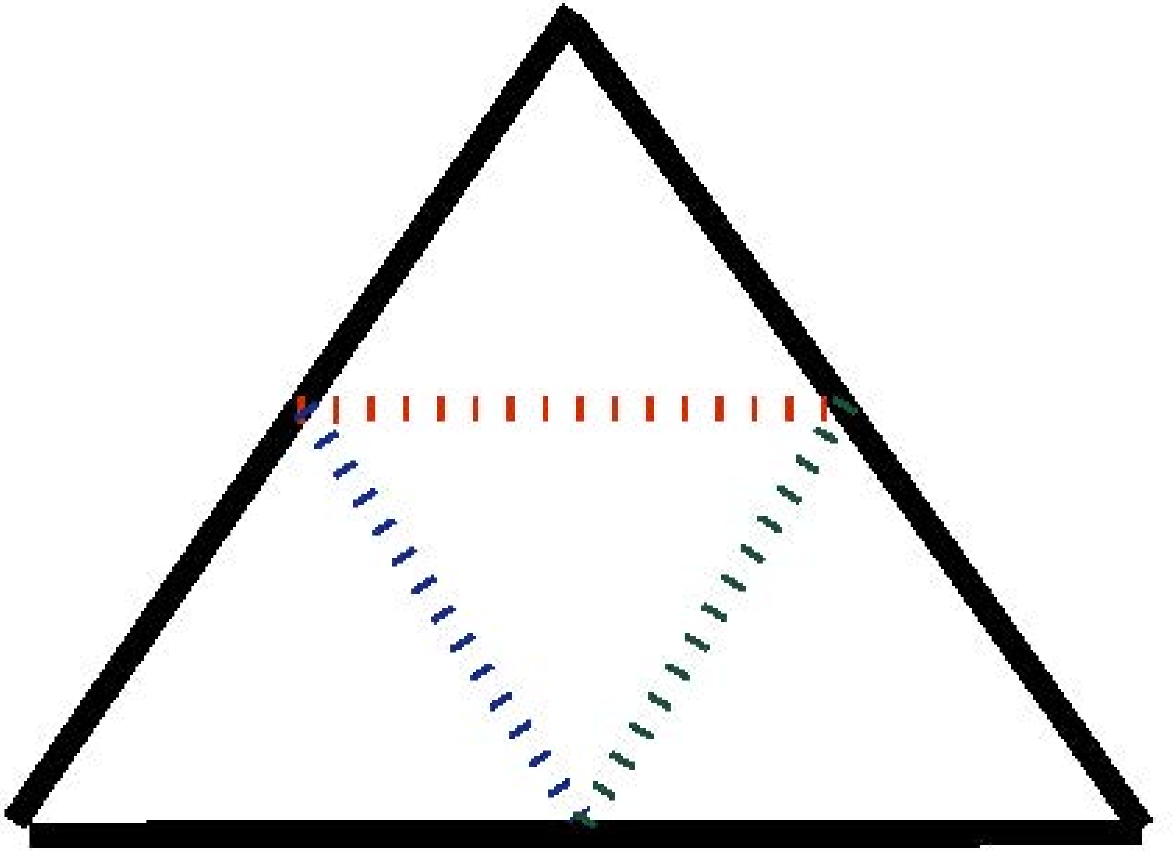

Figure 5: Contraction Simplexes

Multi-Variable Optimization Zero-Order Algorithms: the Finite-Difference Gradient

$$\nabla f(\mathbf{x}^k)_j \sim \frac{1}{\delta}\left(f(\mathbf{x}^k + \delta \mathbf{e}_j) - f(\mathbf{x}^k)\right) \ \forall j$$

for a small $\delta(> 0)$, and they can be estimated in parallel.

Randomized Finite-Dirfference Gradient: Randomly select a block of variables $B \subset of\{1, 2, ..., n\}$ and approximate the gradient vector by

$$\nabla f(\mathbf{x}^k) \sim \frac{n}{|B|}\sum_{j\in B}\frac{1}{\delta}\left(f(\mathbf{x}^k + \delta \mathbf{e}_j) - f(\mathbf{x}^k)\right).$$

Randomly generate n_k i.i.d. Gaussian vectors $\mathbf{u}_i,\ i = 1, ..., n_k$ and and approximate the gradient vector by

$$\nabla f(\mathbf{x}^k) \sim \frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{u}_i[\frac{1}{\delta}\left(f(\mathbf{x}^k + \delta \mathbf{u}_i) - f(\mathbf{x}^k)\right)].$$

Check ZeroorderNLP.m and ZeroordersubNLP.m, which is modified from the derivative-free nonlinear optimization solver "SOLNP". For more advanced one, see "SOLNP+" (https://arxiv.org/abs/2210.07160)!