

One-Dimensional Minimization

Lectures for PHD course on
Numerical optimization

Enrico Bertolazzi

DIMS – Università di Trento

November 21 – December 14, 2011

- 1 Golden Section minimization
 - Convergence Rate
- 2 Fibonacci Search Method
 - Convergence Rate
- 3 Polynomial Interpolation

The problem

Definition (Global minimum)

Given a function $\phi : [a, b] \mapsto \mathbb{R}$, a point $x^* \in [a, b]$ is a **global minimum** if

$$\phi(x^*) \leq \phi(x), \quad \forall x \in [a, b].$$

Definition (Local minimum)

Given a function $\phi : [a, b] \mapsto \mathbb{R}$, a point $x^* \in [a, b]$ is a **local minimum** if there exist a $\delta > 0$ such that

$$\phi(x^*) \leq \phi(x), \quad \forall x \in [a, b] \cap (x^* - \delta, x^* + \delta).$$

Finding a global minimum is generally not an easy task even in the 1D case. The algorithms presented in the following approximate **local minima**.

Interval of Searching

- In many practical problem, $\phi(x)$ is defined in the interval $(-\infty, \infty)$; if $\phi(x)$ is continuous and coercive (i.e. $\lim_{x \rightarrow \pm\infty} f(x) = +\infty$), then there exists a global minimum.
- A simple algorithm can determine an interval $[a, b]$ which contains a local minimum. The method searches 3 consecutive points a, η, b such that $\phi(a) > \phi(\eta)$ and $\phi(b) > \phi(\eta)$ in this way the interval $[a, b]$ certainly contains a local minima.
- In practice the method start from a point a and a step-length $h > 0$; if $\phi(a) > \phi(a + h)$ then the step-length $k > h$ is increased until we have $\phi(a + k) > \phi(a + h)$.
- if $\phi(a) < \phi(a + h)$, then the step-length $k > h$ is increased until we have $\phi(a + h - k) > \phi(a)$.
- This method is called **forward-backward** method.

Algorithm (forward-backward method)

- ① Let us be given α and $h > 0$ and a multiplicative factor $t > 1$ (usually 2).
- ② If $\phi(\alpha) > \phi(\alpha + h)$ goto *forward step* otherwise goto *backward step*
- ③ *forward step*: $a \leftarrow \alpha$; $\eta \leftarrow \alpha + h$;
 - ① $h \leftarrow ht$; $b \leftarrow a + h$;
 - ② if $\phi(b) \geq \phi(\eta)$ then return $[a, b]$;
 - ③ $a \leftarrow \eta$; $\eta \leftarrow b$;
 - ④ goto step 1;
- ④ *backward step*: $\eta \leftarrow \alpha$; $b \leftarrow \alpha + h$;
 - ① $h \leftarrow ht$; $a \leftarrow b - h$;
 - ② if $\phi(a) \geq \phi(\eta)$ then return $[a, b]$;
 - ③ $b \leftarrow \eta$; $\eta \leftarrow a$;
 - ④ goto step 1;

Unimodal function

Definition (Unimodal function)

A function $\phi(x)$ is **unimodal** in $[a, b]$ if there exists an $x^* \in (a, b)$ such that $\phi(x)$ is **strictly** decreasing on $[a, x^*)$ and strictly increasing on $(x^*, b]$.

Another equivalent definition is the following one

Definition (Unimodal function)

A function $\phi(x)$ is **unimodal** in $[a, b]$ if there exists an $x^* \in (a, b)$ such that for all $a < \alpha < \beta < b$ we have:

- if $\beta < x^*$ then $\phi(\alpha) > \phi(\beta)$;
- if $\alpha > x^*$ then $\phi(\alpha) < \phi(\beta)$;

Unimodal function

Golden search and Fibonacci search are based on the following theorem

Theorem (Unimodal function)

Let $\phi(x)$ *unimodal* in $[a, b]$ and let be $a < \alpha < \beta < b$. Then

- 1 if $\phi(\alpha) \leq \phi(\beta)$ then $\phi(x)$ is unimodal in $[a, \beta]$
- 2 if $\phi(\alpha) \geq \phi(\beta)$ then $\phi(x)$ is unimodal in $[\alpha, b]$

Proof.

- 1 From definition $\phi(x)$ is strictly decreasing over $[a, x^*)$, since $\phi(\alpha) \leq \phi(\beta)$ then $x^* \in (a, \beta)$.
- 2 From definition $\phi(x)$ is strictly increasing over $(x^*, b]$, since $\phi(\alpha) \geq \phi(\beta)$ then $x^* \in (\alpha, b)$.

In both cases the function is unimodal in the respective intervals. □



Outline

- 1 Golden Section minimization
 - Convergence Rate
- 2 Fibonacci Search Method
 - Convergence Rate
- 3 Polynomial Interpolation

Golden Section minimization

Let $\phi(x)$ an unimodal function on $[a, b]$, the **golden section** scheme produce a series of intervals $[a_k, b_k]$ where

- $[a_0, b_0] = [a, b]$;
- $[a_{k+1}, b_{k+1}] \subset [a_k, b_k]$;
- $\lim_{k \rightarrow \infty} b_k = \lim_{k \rightarrow \infty} a_k = x^*$;

Algorithm (Generic Search Algorithm)

- 1 Let $a_0 = a, b_0 = b$
- 2 for $k = 0, 1, 2, \dots$
 choose $a_k < \lambda_k < \mu_k < b_k$;
 - 1 if $\phi(\lambda_k) \leq \phi(\mu_k)$ then $a_{k+1} = a_k$ and $b_{k+1} = \mu_k$;
 - 2 if $\phi(\lambda_k) > \phi(\mu_k)$ then $a_{k+1} = \lambda_k$ and $b_{k+1} = b_k$;

Golden Section minimization

- When an algorithm for choosing the **observations** λ_k and μ_k is defined, the **generic search algorithm** is determined.
- Apparently the previous algorithm needs the evaluation of $\phi(\lambda_k)$ and $\phi(\mu_k)$ at each iteration.
- In the **golden section** algorithm, a **fixed** reduction of the interval τ is used, i.e:

$$b_{k+1} - a_{k+1} = \tau(b_k - a_k)$$

- Due to symmetry the observations are determined as follows

$$\lambda_k = b_k - \tau(b_k - a_k)$$

$$\mu_k = a_k + \tau(b_k - a_k)$$

- By a carefully choice of τ , golden search algorithm permits to evaluate only **one** observation per step.

Golden Section minimization

Consider case 1 in the generic search: then,

$$\lambda_k = b_k - \tau(b_k - a_k), \quad \mu_k = a_k + \tau(b_k - a_k)$$

and

$$a_{k+1} = a_k, \quad b_{k+1} = \mu_k = a_k + \tau(b_k - a_k)$$

Now, evaluate

$$\lambda_{k+1} = b_{k+1} - \tau(b_{k+1} - a_{k+1}) = a_k + (\tau - \tau^2)(b_k - a_k)$$

$$\mu_{k+1} = a_{k+1} + \tau(b_{k+1} - a_{k+1}) = a_k + \tau^2(b_k - a_k)$$

The only value that can be reused is λ_k so that we try $\lambda_{k+1} = \lambda_k$ and $\mu_{k+1} = \lambda_k$.



Golden Section minimization

- If $\lambda_{k+1} = \lambda_k$, then

$$b_k - \tau(b_k - a_k) = a_k + (\tau - \tau^2)(b_k - a_k)$$

and $1 - \tau = \tau - \tau^2 \quad \Rightarrow \quad \tau = 1$. In this case there is **no** reduction so that λ_{k+1} must be computed.

- If $\mu_{k+1} = \lambda_k$, then

$$b_k - \tau(b_k - a_k) = a_k + \tau^2(b_k - a_k)$$

and

$$1 - \tau = \tau^2 \quad \Rightarrow \quad \tau^{\pm} = \frac{-1 \pm \sqrt{5}}{2}$$

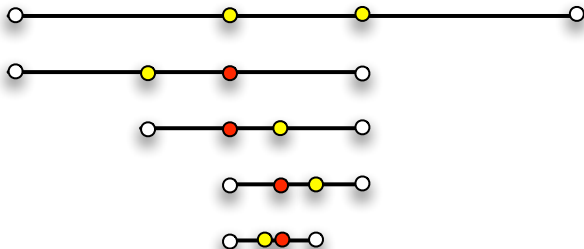
By choosing the positive root, we have

$\tau = (\sqrt{5} - 1)/2 \approx 0.618$. In this case, μ_{k+1} does not need to be computed.

Golden Section minimization

Graphical structure of the **Golden Section** algorithm.

- White circles are the extrema of the successive
- Yellow circles are the newly evaluated values;
- Red circles are the already evaluated values;



Algorithm (Golden Section Algorithm)

Let $\phi(x)$ be an unimodal function in $[a, b]$,

- 1 Set $k = 0$, $\delta > 0$ and $\tau = (\sqrt{5} - 1)/2$. Evaluate $\lambda = b - \tau(b - a)$, $\mu = a + \tau(b - a)$, $\phi_a = \phi(a)$, $\phi_b = \phi(b)$, $\phi_\lambda = \phi(\lambda)$, $\phi_\mu = \phi(\mu)$.
- 2 If $\phi_\lambda > \phi_\mu$ go to step 3; else go to step 4
- 3 If $b - \lambda \leq \delta$ stop and output μ ;
otherwise, set $a \leftarrow \lambda$, $\lambda \leftarrow \mu$, $\phi_\lambda \leftarrow \phi_\mu$ and evaluate $\mu = a + \tau(b - a)$ and $\phi_\mu = \phi(\mu)$.
Go to step 5
- 4 If $\mu - a \leq \delta$ stop and output λ ;
otherwise, set $b \leftarrow \mu$, $\mu \leftarrow \lambda$, $\phi_\mu \leftarrow \phi_\lambda$ and evaluate $\lambda = b - \tau(b - a)$ and $\phi_\lambda = \phi(\lambda)$.
Go to step 5
- 5 $k \leftarrow k + 1$ goto step 2.

Golden Section convergence rate

- At each iteration the interval length containing the minimum of $\phi(x)$ is reduced by τ so that $b_k - a_k = \tau^k(b_0 - a_0)$.
- Due to the fact that $x^* \in [a_k, b_k]$ for all k then we have:

$$(b_k - x^*) \leq (b_k - a_k) \leq \tau^k(b_0 - a_0)$$

$$(x^* - a_k) \leq (b_k - a_k) \leq \tau^k(b_0 - a_0)$$

- This means that $\{a_k\}$ and $\{b_k\}$ are r -linearly convergent sequence with coefficient $\tau \approx 0.618$.



Outline

- 1 Golden Section minimization
 - Convergence Rate
- 2 Fibonacci Search Method
 - Convergence Rate
- 3 Polynomial Interpolation

Fibonacci Search Method

- In the Golden Search Method, the reduction factor τ is unchanged during the search.
- If we allow to change the reduction factor at each step we have a chance to produce a faster minimization algorithm.
- In the next slides we see that there are only two possible choice of the reduction factor:
 - The first choice is $\tau_k = (\sqrt{5} - 1)/2$ and gives the golden search method.
 - The second choice takes τ_k as the ratio of two consecutive Fibonacci numbers and gives the so-called Fibonacci search method.



Fibonacci Search Method

Consider case 1 in the generic search: the reduction step τ_k can vary with respect to the index k as

$$\lambda_k = b_k - \tau_k(b_k - a_k), \quad \mu_k = a_k + \tau_k(b_k - a_k)$$

and

$$a_{k+1} = a_k, \quad b_{k+1} = \mu_k = a_k + \tau_k(b_k - a_k)$$

Now, evaluate

$$\lambda_{k+1} = b_{k+1} - \tau_{k+1}(b_{k+1} - a_{k+1}) = a_k + (\tau_k - \tau_k\tau_{k+1})(b_k - a_k)$$

$$\mu_{k+1} = a_{k+1} + \tau_{k+1}(b_{k+1} - a_{k+1}) = a_k + \tau_k\tau_{k+1}(b_k - a_k)$$

The only value that can be reused is λ_k , so that we try $\lambda_{k+1} = \lambda_k$ and $\mu_{k+1} = \lambda_k$.



Fibonacci Search Method

- If $\lambda_{k+1} = \lambda_k$, then

$$b_k - \tau_k(b_k - a_k) = a_k + (\tau_k - \tau_k\tau_{k+1})(b_k - a_k)$$

and $1 - \tau_k = \tau_k - \tau_k\tau_{k+1}$. By searching a solution of the form $\tau_k = z_{k+1}/z_k$, we have the recurrence relation:

$$z_k - 2z_{k+1} + z_{k+2} = 0$$

which has a generic solution of the form

$$z_k = c_1 + c_2(k + 1)$$

In general, we have $\lim_{k \rightarrow \infty} \tau_k = 1$, so that reduction is **asymptotically worse** than golden section.



Fibonacci Search Method

- If $\mu_{k+1} = \lambda_k$, then

$$b_k - \tau_k(b_k - a_k) = a_k + \tau_k\tau_{k+1}(b_k - a_k)$$

and $1 - \tau_k = \tau_k\tau_{k+1}$. By searching a solution of the form $\tau_k = z_{k+1}/z_k$, we have the recurrence relation:

$$z_k = z_{k+1} + z_{k+2}$$

which is a **reverse** Fibonacci succession. The computation of z_k involves complex number.



Fibonacci Search Method

- A simpler way to compute z_k is to take the length of the reduction step **constant**, say n and compute the Fibonacci sequence up to n as follows

$$F_0 = F_1 = 1, \quad F_{k+1} = F_k + F_{k-1}$$

then, set $z_k = F_{n-k+1}$ so that $\tau_k = F_{n-k}/F_{n-k+1}$.

- In the Fibonacci search we evaluate reduction factor τ_k by choosing the number of reductions **before** starting the algorithm
- A way to evaluate this number is to choose a tolerance δ so that

$$b_n - a_n \leq \delta$$



Fibonacci Search Method

- ① From the definition of the reduction factor τ_k , it is easy to evaluate $b_n - a_n$:

$$\begin{aligned} b_n - a_n &= \frac{F_1}{F_2} (b_{n-1} - a_{n-1}) = \frac{F_1}{F_2} \frac{F_2}{F_3} (b_{n-2} - a_{n-2}) \\ &= \frac{F_1}{F_2} \frac{F_2}{F_3} \cdots \frac{F_n}{F_{n+1}} (b_0 - a_0) = \frac{b_0 - a_0}{F_{n+1}} \end{aligned}$$

- ② In this way the number of reductions n is deduced from:

$$F_{n+1} \geq \frac{b_0 - a_0}{\delta}$$



Algorithm (Fibonacci Search Algorithm)

Let $\phi(x)$ be an unimodal function in $[a, b]$

- 1 Set $k = 0$, $\delta > 0$ and n such that $F_{n+1} \geq (b_0 - a_0)/\delta$.
Evaluate $\tau = F_n/F_{n+1}$, $\lambda = b - \tau(b - a)$, $\mu = a + \tau(b - a)$,
 $\phi_a = \phi(a)$, $\phi_b = \phi(b)$, $\phi_\lambda = \phi(\lambda)$, $\phi_\mu = \phi(\mu)$.
- 2 If $\phi_\lambda > \phi_\mu$ go to step 3; else go to step 4
- 3 If $b - \lambda \leq \delta$ stop and output μ ;
otherwise set $a \leftarrow \lambda$, $\lambda \leftarrow \mu$, $\phi_\lambda \leftarrow \phi_\mu$ evaluate
 $\mu = a + \tau(b - a)$ and $\phi_\mu = \phi(\mu)$.
Go to step 5
- 4 If $\mu - a \leq \delta$ stop and output λ ;
otherwise set $b \leftarrow \mu$, $\mu \leftarrow \lambda$, $\phi_\mu \leftarrow \phi_\lambda$ evaluate
 $\lambda = b - \tau(b - a)$ and $\phi_\lambda = \phi(\lambda)$.
Go to step 5
- 5 set $k \leftarrow k + 1$ and $\tau \leftarrow F_{n-k}/F_{n-k+1}$ goto step 2.



Fibonacci Search convergence rate

- At each iteration, the interval length containing the minimum of $\phi(x)$ is

$$b_k - a_k = (b_0 - a_0)(F_{n-k+1}/F_{n+1})$$

- Due to the fact that $x^* \in [a_k, b_k]$ for all k , we have:

$$(b_k - x^*) \leq (b_k - a_k) \leq (F_{n-k+1}/F_{n+1})(b_0 - a_0)$$

$$(x^* - a_k) \leq (b_k - a_k) \leq (F_{n-k+1}/F_{n+1})(b_0 - a_0)$$



Fibonacci Search convergence rate

- To estimate convergence rate we need the expression of F_k

$$F_k = \frac{1}{\sqrt{5}} \left\{ \left(\frac{1 + \sqrt{5}}{2} \right)^{k+1} - \left(\frac{1 - \sqrt{5}}{2} \right)^{k+1} \right\}$$

- and for large k

$$F_k \approx \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^{k+1}$$

- in this way we can approximate

$$\frac{F_{n-k+1}}{F_{n+1}} \approx \left(\frac{1 + \sqrt{5}}{2} \right)^{-k} = \left(\frac{\sqrt{5} - 1}{2} \right)^k$$



Fibonacci Search convergence rate

- This means that $\{a_k\}$ and $\{b_k\}$ are r -linearly convergent sequences with coefficient $\tau \approx 0.618$.
- So, golden search and Fibonacci search perform similarly for large n . Golden search is easier, for this reason, normally Golden search is preferred to Fibonacci search.

Outline

- 1 Golden Section minimization
 - Convergence Rate
- 2 Fibonacci Search Method
 - Convergence Rate
- 3 Polynomial Interpolation

Polynomial Interpolation

- Fibonacci and golden search are r -linearly convergent methods.
- Approximating the function $\phi(x)$ with a polynomial model and minimizing the polynomial result in algorithms which are normally superior to Fibonacci and golden search.

Polynomial Interpolation

- Suppose that an initial guess x_0 is known, and the interval $[0, x_0]$ contains a minimum.
- We can form the quadratic approximation $p(x)$ to $\phi(x)$ by interpolating $\phi(0)$, $\phi(x_0)$ and $\phi'(0)$.

$$q(x) = \frac{\phi(x_0) - \phi(0) - x_0\phi'(0)}{x_0^2}x^2 + \phi'(0)x + \phi(0).$$

The new trial minimum is defined as the minimum of the polynomial approximation $q(x)$, and takes the value:

$$x_1 = -\frac{\phi'(0)x_0^2}{2[\phi(x_0) - \phi(0) - \phi'(0)x_0]}$$



Polynomial Interpolation

- If $\phi'(x_1)$ is **small enough** (we are near a stationary point) we can stop the iteration, otherwise we can construct a **cubic** polynomial that interpolates $\phi(0)$, $\phi'(0)$, $\phi(x_0)$ and $\phi(x_1)$.

$$c(x) = A_1x^3 + B_1x^2 + \phi'(0)x + \phi(0).$$

where

$$\begin{pmatrix} A_1 \\ B_1 \end{pmatrix} = \frac{1}{x_0^2 x_1^2 (x_1 - x_0)} \begin{pmatrix} x_0^2 & -x_1^2 \\ -x_0^3 & x_1^3 \end{pmatrix} \begin{pmatrix} \phi(x_1) - \phi(0) - \phi'(0)x_1 \\ \phi(x_0) - \phi(0) - \phi'(0)x_0 \end{pmatrix}$$

The new trial minimum is defined as the minimum of the polynomial approximation $c(x)$.



Polynomial Interpolation

- By differentiating $c(x)$ and taking the root nearest the 0 values we obtain:

$$\begin{aligned} x_2 &= \frac{-B_1 + \sqrt{B_1^2 - 3A_1\phi'(0)}}{A_1} \\ &= \frac{-\phi'(0)}{B_1 + \sqrt{B_1^2 - 3A_1\phi'(0)}} \end{aligned}$$

where for stability reason we use the first expression when $B_1 < 0$, the second expression when $B_1 \geq 0$.

- If the new trial minimum is not accepted, we repeat the procedure with $\phi(0)$, $\phi'(0)$, $\phi(x_1)$ and $\phi(x_2)$.



Polynomial Interpolation

- In general we can approximate the minimum by the procedure

$$\begin{aligned}
 x_{k+1} &= \frac{-B_k + \sqrt{B_k^2 - 3A_k\phi'(0)}}{A_k} \\
 &= \frac{-\phi'(0)}{B_k + \sqrt{B_k^2 - 3A_k\phi'(0)}}
 \end{aligned}$$

- where

$$\begin{aligned}
 \begin{pmatrix} A_k \\ B_k \end{pmatrix} &= \frac{1}{x_{k-1}^2 x_k^2 (x_k - x_{k-1})} \begin{pmatrix} x_{k-1}^2 & -x_k^2 \\ -x_{k-1}^3 & x_k^3 \end{pmatrix} \\
 &\quad \times \begin{pmatrix} \phi(x_k) - \phi(0) - \phi'(0)x_k \\ \phi(x_{k-1}) - \phi(0) - \phi'(0)x_{k-1} \end{pmatrix}
 \end{aligned}$$

References



J. Stoer and R. Bulirsch

Introduction to numerical analysis

Springer-Verlag, Texts in Applied Mathematics, **12**, 2002.



J. E. Dennis, Jr. and Robert B. Schnabel

Numerical Methods for Unconstrained Optimization and
Nonlinear Equations

SIAM, Classics in Applied Mathematics, **16**, 1996.

