Unidimensional Search Methods

Dr. José Ernesto Rayas Sánchez

1

Outline

- Unidimensional optimization problems
- Well-behaved and badly-behaved functions
- Multimodal and unimodal functions
- Methods for optimizing unimodal functions
- Golden Section method
- Fibonacci method
- Quadratic interpolation method
- Available commands in Matlab

Dr. J. E. Rayas Sánchez 2

Unidimensional Optimization Problems

- Many multidimensional optimization strategies require one-dimensional techniques to search along some feasible direction at each iteration
- Given u: $\Re^n \rightarrow \Re$ and $\boldsymbol{x} \in \Re^n$, when solving

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} u(\boldsymbol{x})$$

we can select at the i-th iterate $\boldsymbol{x}_i$ a search direction $\boldsymbol{d}_i$, and the next iterate $\boldsymbol{x}_{i+1}$ can be found by solving

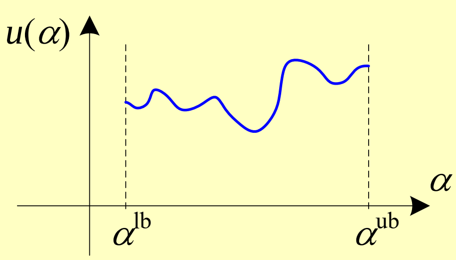
$$\alpha^* = \arg\min_{\alpha} u(\boldsymbol{x}_i + \alpha \boldsymbol{d}_i) = \arg\min_{\alpha} u(\alpha)$$

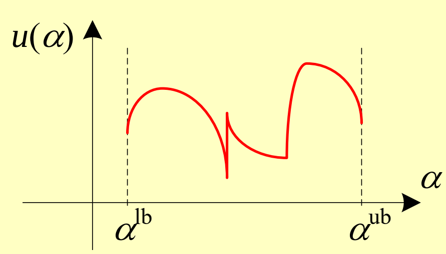
then $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha^* \boldsymbol{d}_i$

The above problem is called "exact line search"

Well-Behaved and Badly-Behaved Functions

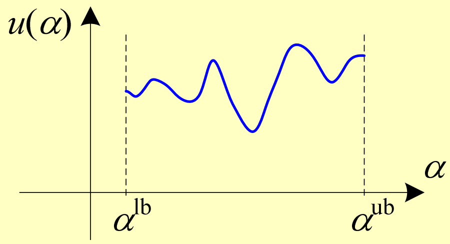
- Well-behaved functions: continuous with continuous derivatives
- Badly-behaved functions: discontinuous with discontinuous derivatives

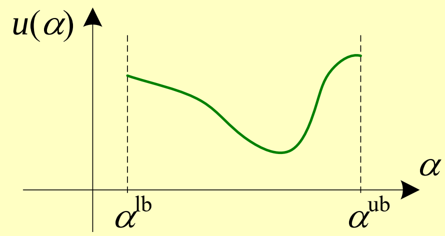




Unimodal and Multimodal Functions

- Multimodal functions: several minima at the selected interval
- Unimodal functions: only one minimum at the selected interval



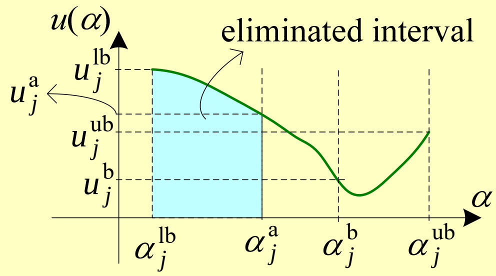


Optimization Methods for Unimodal Functions

- Interval elimination methods
  - Golden section method
  - Fibonacci search
- Interpolation methods
  - Quadratic interpolation
  - Cubic interpolation
  - Newton method
  - Secant method

Interval Elimination Methods

- Assuming a unimodal interval at the *j*-th iteration, we can always eliminate a subinterval by evaluating the function at 2 interior points



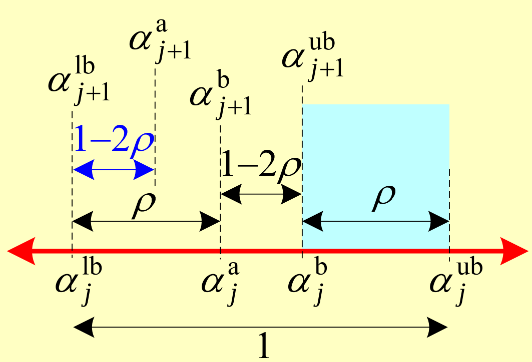
- Reducing the interval

If $u_j^a > u_j^b$ the minimum lies in $[\alpha_j^a, \alpha_j^{ub}]$ $\longrightarrow$ $\alpha_{j+1}^{lb} = \alpha_j^a,\ \alpha_{j+1}^{ub} = \alpha_j^{ub}$

If $u_j^a < u_j^b$ the minimum lies in $[\alpha_j^{lb}, \alpha_j^{b}]$ $\longrightarrow$ $\alpha_{j+1}^{lb} = \alpha_j^{lb},\ \alpha_{j+1}^{ub} = \alpha_j^{b}$

Golden Section Method

- The interior points are symmetrically selected
- The previous interior points are re-used at the next iteration
- The same relative reduction is used at each iteration



$$\rho[\rho + (1-2\rho)] = 1-2\rho$$

$$\rho^2 - 3\rho + 1 = 0$$

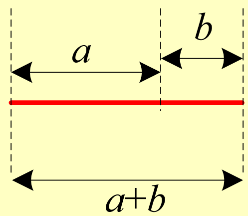
$$\rho = (3 \pm \sqrt{5})/2$$

Since $0 < \rho < 0.5$,

$$\rho = (3 - \sqrt{5})/2 = 0.38196...$$

Golden Section Method – Greek Geometers

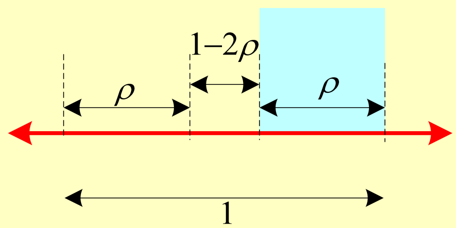
- The "golden ratio" or "golden proportion"



$$\frac{a}{b} = \frac{a+b}{a} \equiv \varphi$$

$$\varphi = \frac{1+\sqrt{5}}{2} \approx 1.61803...$$

$$\varphi = \frac{1}{\rho} - 1$$



(*wikipedia.org*)

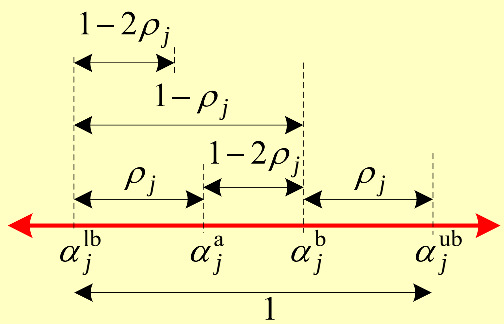
Golden Section Algorithm

$\alpha^* = \text{GoldenSection}(u, \alpha^{\text{lb}}, \alpha^{\text{ub}})$
u: $\Re \rightarrow \Re$; $\alpha^{\text{lb}}, \alpha^{\text{ub}}, \alpha^* \in \Re$

begin
  $j = 0$; $\alpha_j^{\text{lb}} = \alpha^{\text{lb}}$; $\alpha_j^{\text{ub}} = \alpha^{\text{ub}}$; $\rho = (3-\sqrt{5})/2$
  $\alpha_j^{\text{a}} = \alpha_j^{\text{lb}} + \rho(\alpha_j^{\text{ub}} - \alpha_j^{\text{lb}})$; $\alpha_j^{\text{b}} = \alpha_j^{\text{lb}} + (1-\rho)(\alpha_j^{\text{ub}} - \alpha_j^{\text{lb}})$
  $u_j^{\text{a}} = u(\alpha_j^{\text{a}})$; $u_j^{\text{b}} = u(\alpha_j^{\text{b}})$
  repeat until *StoppingCriteria*
    if $u_j^{\text{a}} > u_j^{\text{b}}$
      $\alpha_{j+1}^{\text{lb}} = \alpha_j^{\text{a}}$; $\alpha_{j+1}^{\text{ub}} = \alpha_j^{\text{ub}}$
      $\alpha_{j+1}^{\text{a}} = \alpha_j^{\text{b}}$; $\alpha_{j+1}^{\text{b}} = \alpha_{j+1}^{\text{lb}} + (1-\rho)(\alpha_{j+1}^{\text{ub}} - \alpha_{j+1}^{\text{lb}})$
      $u_{j+1}^{\text{a}} = u_j^{\text{b}}$; $u_{j+1}^{\text{b}} = u(\alpha_{j+1}^{\text{b}})$
    else
      $\alpha_{j+1}^{\text{lb}} = \alpha_j^{\text{lb}}$; $\alpha_{j+1}^{\text{ub}} = \alpha_j^{\text{b}}$
      $\alpha_{j+1}^{\text{b}} = \alpha_j^{\text{a}}$; $\alpha_{j+1}^{\text{a}} = \alpha_{j+1}^{\text{lb}} + \rho(\alpha_{j+1}^{\text{ub}} - \alpha_{j+1}^{\text{lb}})$
      $u_{j+1}^{\text{a}} = u(\alpha_{j+1}^{\text{a}})$; $u_{j+1}^{\text{b}} = u_j^{\text{a}}$
    end
    $j = j + 1$
  end
  $\alpha^* = (\alpha_j^{\text{lb}} + \alpha_j^{\text{ub}})/2$
end

Fibonacci Method

- The interior points are symmetrically selected
- The previous interior points are re-used at the next iteration
- A different relative reduction is used at each iteration



$$\rho_{j+1}(1-\rho_j) = 1-2\rho_j$$

$$\rho_{j+1} = 1-\frac{\rho_j}{1-\rho_j}$$

Fibonacci Method (cont)

- A sequence of numbers that satisfy

$$\rho_{j+1} = 1-\frac{\rho_j}{1-\rho_j}$$

is the following

$$\rho_1 = 1-\frac{F_N}{F_{N+1}} \qquad \rho_2 = 1-\frac{F_{N-1}}{F_N} \qquad \cdots \qquad \rho_j = 1-\frac{F_{N-j+1}}{F_{N-j+2}}$$

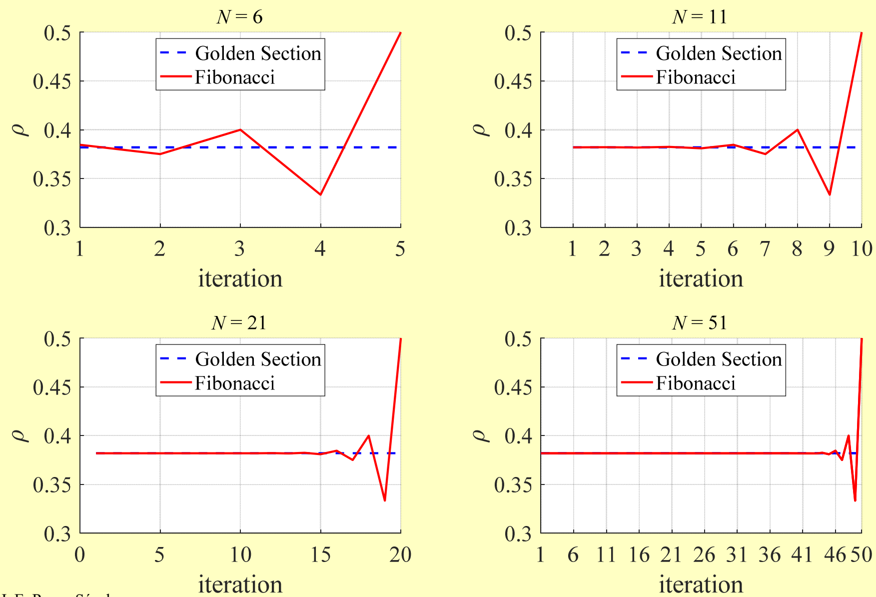
where F_k is the k-th Fibonacci number.

- The Fibonacci sequence is

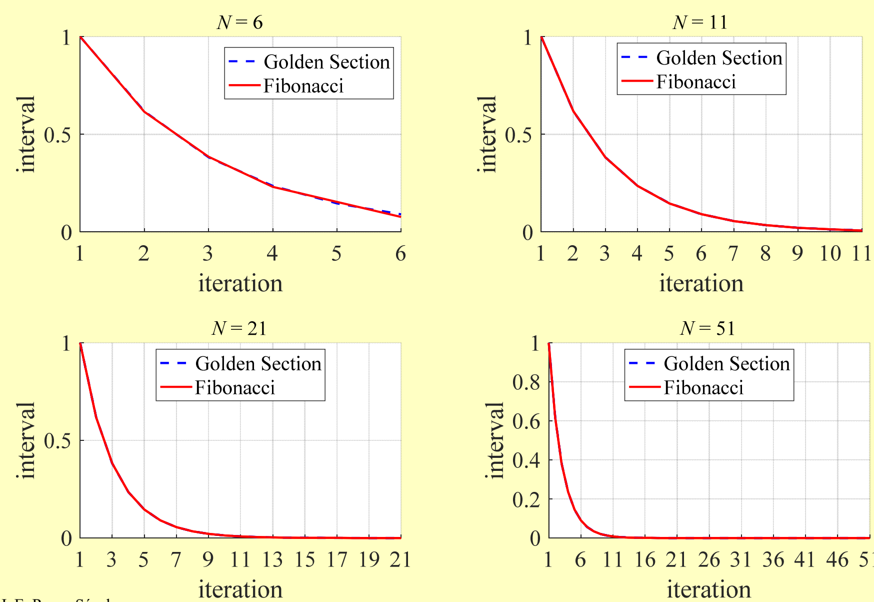
$F_{k+1} = F_k + F_{k-1}$ with $F_{-1} = 0, F_0 = 1$

$$\{1, 2, 3, 5, 8, 13, 21, \ldots\}$$

Relative Reduction: Fibonacci *vs* Golden Section



Search Interval: Fibonacci *vs* Golden Section



Golden Section Method *vs* Fibonacci Method

- The Fibonacci method has slightly higher rate of convergence than the Golden Section method
- For a large number of iterations (N large), both methods achieve almost the same uncertainty interval
- The Golden Section method is preferred because it does not require to define N in advance

Quadratic Interpolation Method

- At the j-th iteration it also assumes an unimodal interval

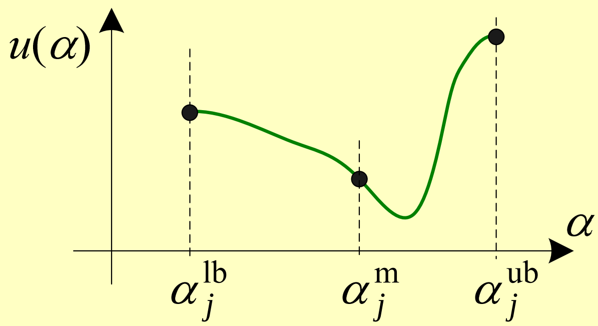
$$[\alpha_j^{\mathrm{lb}}, \alpha_j^{\mathrm{ub}}]$$

- It finds an initial interior point, α^{m}_j, such that

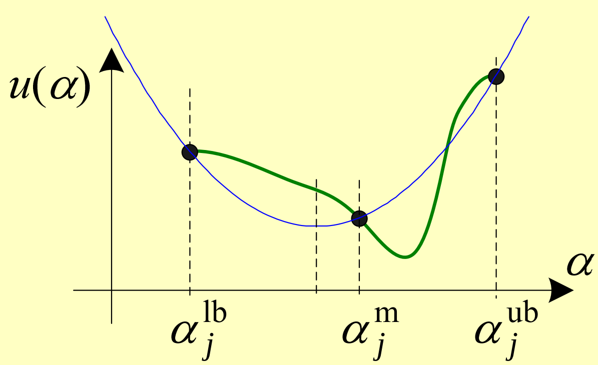
$$u(\alpha_j^{\mathrm{lb}}) > u(\alpha_j^{\mathrm{m}}) \text{ and } u(\alpha_j^{\mathrm{ub}}) > u(\alpha_j^{\mathrm{m}})$$

- It fits a quadratic polynomial to the function $u(\alpha)$ over the three previous points at each iteration
- The minimum of the quadratic polynomial, and 2 of the 3 previous points are used for successive interpolations
- Convergence is guaranteed

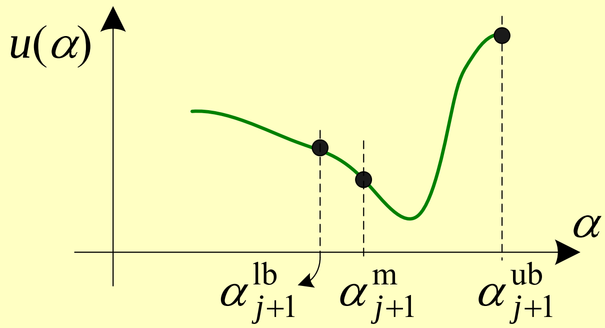
Quadratic Interpolation – Illustration



Quadratic Interpolation – Illustration (cont)

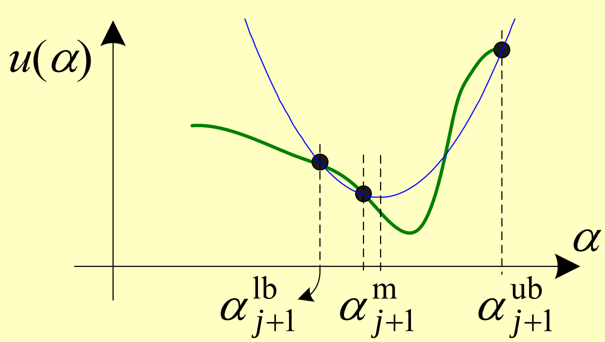


Quadratic Interpolation – Illustration (cont)



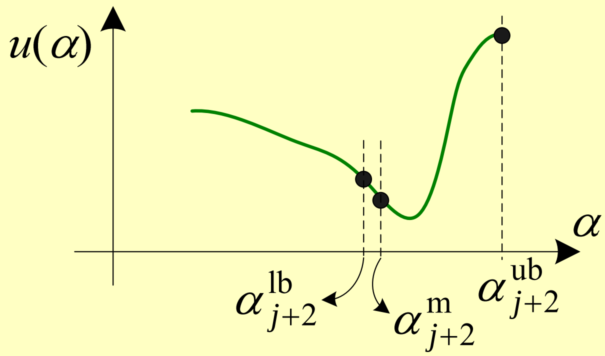
Dr. J. E. Rayas Sánchez 19

Quadratic Interpolation – Illustration (cont)



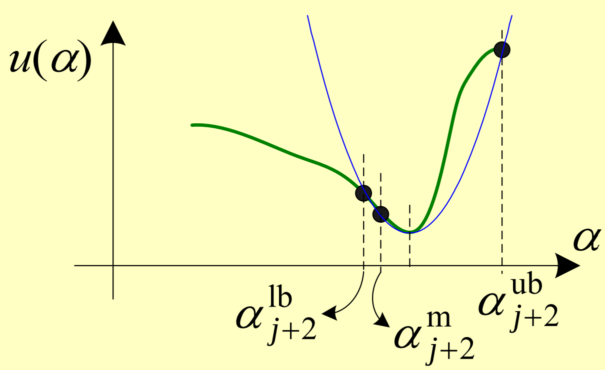
Dr. J. E. Rayas Sánchez 20

Quadratic Interpolation – Illustration (cont)



Dr. J. E. Rayas Sánchez 21

Quadratic Interpolation – Illustration (cont)



Dr. J. E. Rayas Sánchez 22

Quadratic Interpolation Formula

- At the *j*-th iteration, let

$$a = \alpha_j^{\text{lb}} \qquad b = \alpha_j^{\text{m}} \qquad c = \alpha_j^{\text{ub}}$$

$$u_a = u(a) \qquad u_b = u(b) \qquad u_c = u(c)$$

- The minimizer of the quadratic, *d*, is calculated using

$$d = \frac{1}{2}\frac{(b^2 - c^2)u_a + (c^2 - a^2)u_b + (a^2 - b^2)u_c}{(b-c)u_a + (c-a)u_b + (a-b)u_c}$$

Dr. J. E. Rayas Sánchez *(Bandler, 1997)* 23

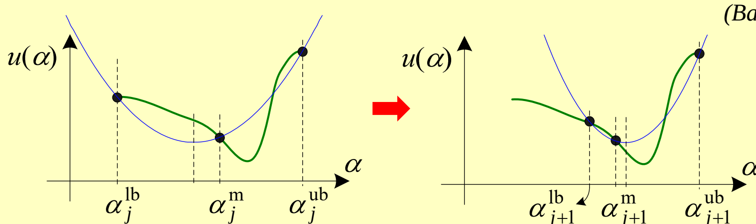
Quadratic Interpolation Method – Next Points

- The next points $\alpha_{j+1}^{\text{lb}} \quad \alpha_{j+1}^{\text{m}} \quad \alpha_{j+1}^{\text{ub}}$

are obtained using

$$\text{If} \begin{cases} b > d \text{ and } \begin{cases} u_b > u_d & \text{then } \alpha_{j+1}^{\text{lb}} = a, \alpha_{j+1}^{\text{m}} = d, \alpha_{j+1}^{\text{ub}} = b \\ u_b < u_d & \text{then } \alpha_{j+1}^{\text{lb}} = d, \alpha_{j+1}^{\text{m}} = b, \alpha_{j+1}^{\text{ub}} = c \end{cases} \\ b < d \text{ and } \begin{cases} u_b > u_d & \text{then } \alpha_{j+1}^{\text{lb}} = b, \alpha_{j+1}^{\text{m}} = d, \alpha_{j+1}^{\text{ub}} = c \\ u_b < u_d & \text{then } \alpha_{j+1}^{\text{lb}} = a, \alpha_{j+1}^{\text{m}} = b, \alpha_{j+1}^{\text{ub}} = d \end{cases} \end{cases}$$

(Bandler, 1997)



Dr. J. E. Rayas Sánchez 24

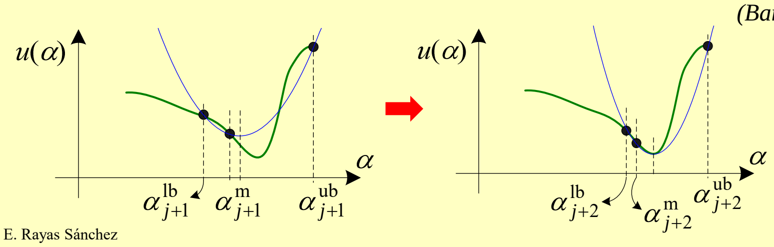
Quadratic Interpolation Method – Next Points

- The next points α_{j+1}^{lb} α_{j+1}^{m} α_{j+1}^{ub}

  are obtained using

$$
\text{If}\begin{cases} b > d \ \text{ and } \begin{cases} u_b > u_d \ \text{ then } \ \alpha_{j+1}^{\text{lb}} = a, \alpha_{j+1}^{\text{m}} = d, \alpha_{j+1}^{\text{ub}} = b \\ u_b < u_d \ \text{ then } \ \alpha_{j+1}^{\text{lb}} = d, \alpha_{j+1}^{\text{m}} = b, \alpha_{j+1}^{\text{ub}} = c \end{cases} \\ b < d \ \text{ and } \begin{cases} u_b > u_d \ \text{ then } \ \alpha_{j+1}^{\text{lb}} = b, \alpha_{j+1}^{\text{m}} = d, \alpha_{j+1}^{\text{ub}} = c \\ u_b < u_d \ \text{ then } \ \alpha_{j+1}^{\text{lb}} = a, \alpha_{j+1}^{\text{m}} = b, \alpha_{j+1}^{\text{ub}} = d \end{cases} \end{cases}
$$

(Bandler, 1997)



Available Commands in Matlab

- The standard version of Matlab has the following command for minimizing scalar unidimensional functions:

  x = fminbnd(fun,x1,x2)

  returns a scalar x that is a local minimizer in the interval x1 ≤ x ≤ x2 of the scalar unidimensional function whose name is in string variable fun

- Matlab employs an algorithm based on the Golden Section and the quadratic interpolation methods; the method is very efficient

Exact Line Search

- Given u: $\Re^n \rightarrow \Re$ and $\boldsymbol{x} \in \Re^n$, when solving

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} u(\boldsymbol{x})$$

at the i-th iterate $\boldsymbol{x}_i$ a descent search direction $\boldsymbol{d}_i$ is used and the next iterate $\boldsymbol{x}_{i+1}$ is found by solving

$$\alpha^* = \arg\min_{\alpha>0} u(\boldsymbol{x}_i + \alpha \boldsymbol{d}_i) = \arg\min_{\alpha>0} u(\alpha)$$

then

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha^* \boldsymbol{d}_i$$

Exact Line Search (cont.)

- Defining $u(\alpha)$ with no unitary direction

$$\alpha^* = \arg\min_{\alpha>0} u(\boldsymbol{x}_i + \alpha \boldsymbol{d}_i) = \arg\min_{\alpha>0} u(\alpha)$$

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha^* \boldsymbol{d}_i$$

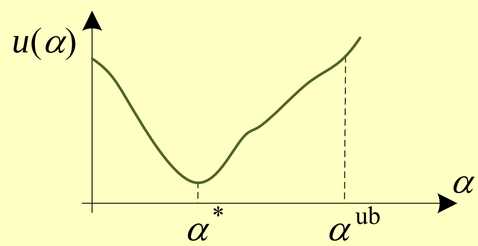
- Defining $u(\alpha)$ with a unitary direction

$$\alpha^* = \arg\min_{\alpha>0} u(\boldsymbol{x}_i + \alpha \frac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|}) = \arg\min_{\alpha>0} u(\alpha)$$

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha^* \frac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|}$$

Bounding the Exact Line Search

$$\alpha^* = \arg \min_{0<\alpha\leq\alpha^{\text{ub}}} u(\alpha)$$



Defining α^{ub} :

1) Arbitrary, e.g., $\alpha^{\text{ub}} = 10$

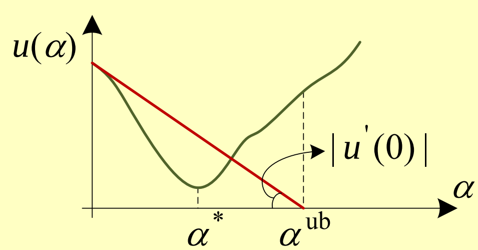
2) Proportional to the current iterate ($m > 0$),

if using $u(\alpha) = u(\boldsymbol{x}_i + \alpha \boldsymbol{d}_i)$ then $\alpha^{\text{ub}} = m \dfrac{\|\boldsymbol{x}_i\|_2 + \varepsilon}{\|\boldsymbol{d}_i\|_2}$

if using $u(\alpha) = u\left(\boldsymbol{x}_i + \alpha \dfrac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|_2}\right)$ then $\alpha^{\text{ub}} = m(\|\boldsymbol{x}_i\|_2 + \varepsilon)$

Bounding the Exact Line Search

$$\alpha^* = \arg \min_{0<\alpha\leq\alpha^{\text{ub}}} u(\alpha)$$



Defining α^{ub} :

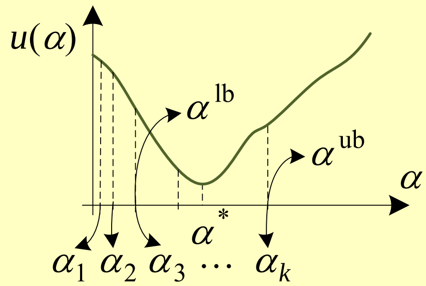
3) Using the rate of change at the current iterate,

taking $|u'(0)| = \dfrac{|u(0)|}{\alpha^{\text{ub}}}$ then $\alpha^{\text{ub}} = |u(0)| / |u'(0)|$

if using $u(\alpha) = u(\boldsymbol{x}_i + \alpha \boldsymbol{d}_i)$ then $u'(\alpha) = \dfrac{du}{d\boldsymbol{x}} \dfrac{d\boldsymbol{x}}{d\alpha} = \nabla u^{\text{T}} \boldsymbol{d}_i$

if using $u(\alpha) = u\left(\boldsymbol{x}_i + \alpha \dfrac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|_2}\right)$ then $u'(\alpha) = \nabla u^{\text{T}} \dfrac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|_2}$

Screening to Define Bounds

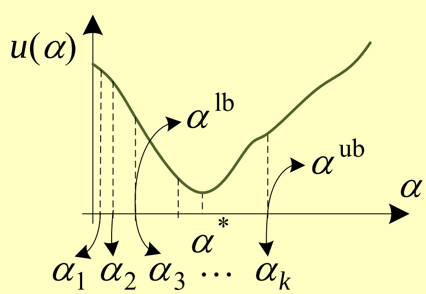


$\{\alpha^{lb}, \alpha^{ub}, k\}$ = LineSearchBounds(u, $\boldsymbol{x}_i$, $\boldsymbol{d}_i$)
u: $\Re \rightarrow \Re$; $\boldsymbol{x}_i$, $\boldsymbol{d}_i \in \Re^n$; α^{lb}, α^{ub}, $k \in \Re$

begin
  $k = 1$; $\alpha = 0$; $u = u(\alpha)$;
  $\alpha = 0.001$; $u_{new} = u(\alpha)$; $k = k + 1$;
  while $u > u_{new}$
    $u = u_{new}$
    $\alpha = 2\alpha$
    $u_{new} = u(\alpha)$
    $k = k + 1$
  end
  if $k = 2$ **then** $\alpha^{lb} = 0$
    else $\alpha^{lb} = \alpha/4$
  end
  $\alpha^{ub} = \alpha$
end

Screening at
$\alpha = 2^k(0.001)$

Screening to Define Bounds (cont.)



$\{\alpha^{lb}, \alpha^{ub}, k\}$ = LineSearchBounds(u, $\boldsymbol{x}_i$, $\boldsymbol{d}_i$)
u: $\Re \rightarrow \Re$; $\boldsymbol{x}_i$, $\boldsymbol{d}_i \in \Re^n$; α^{lb}, α^{ub}, $k \in \Re$

begin
  initialize r , α_{min}
  $k = 1$; $\alpha = 0$; $u = u(\alpha)$;
  $\alpha = \alpha_{min}$; $u_{new} = u(\alpha)$; $k = k + 1$;
  while $u > u_{new}$
    $u = u_{new}$
    $\alpha = r\alpha$
    $u_{new} = u(\alpha)$
    $k = k + 1$
  end
  if $k = 2$ **then** $\alpha^{lb} = 0$
    else $\alpha^{lb} = \alpha/r^2$
  end
  $\alpha^{ub} = \alpha$
end

Screening at
$\alpha = r^k(\alpha_{min})$

Screening to Define Bounds (cont.)

Screening at $\alpha = r^k(\alpha_{\min})$