Numerical Optimization

A Workshop
At
Department of Mathematics
Chiang Mai University
August 4-15, 2009

Instructor: **Vira Chankong**

Electrical Engineering and Computer Science
Case Western Reserve University

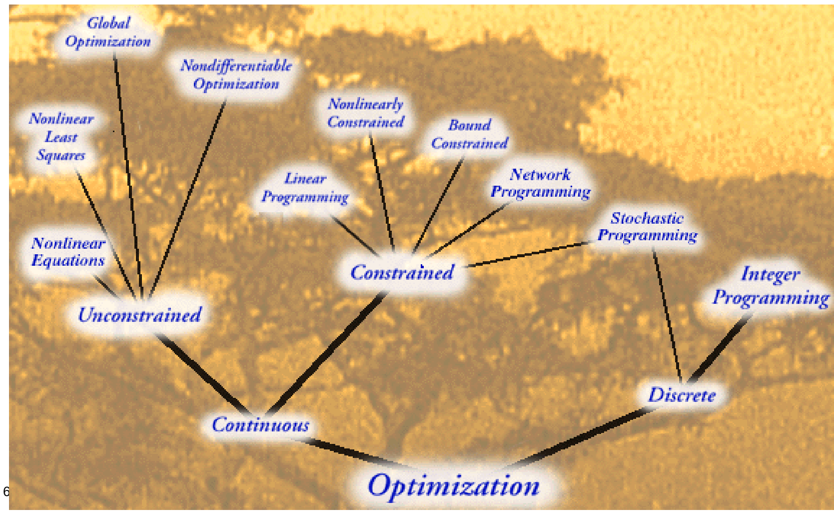
Phone: 216 368 4054, Fax: 216 368 3123
E-mail: **vira@case.edu**

Session:

Methods For Unconstrained Optimization Problems

Vira Chankong
Case Western Reserve University
Electrical Engineering and Computer Science

NEOS Guide Optimization Tree



Continuous Optimization Problems

Typical LP/NLP:

P: $\min f(x)$

s.t. $h_j(x) = 0,\ j = 1, \ldots, m_1$

$g_j(x) \le 0,\ j = 1, \ldots, m_2$

$l_i \le x_i \le u_i,\ i = 1, \ldots, n,\ (\mathbf{x} \in R^n)$

Where some or all of f, h_j, and/or g_j are nonlinear.

LP: If all f, h_j, and g_j are all linear/affine, then P is LP

NLP: If at least one of f, h_j, or g_j is nonlinear, P is NLP

Classification:

Unconstrained NLP:----$m_1 = 0$; $m_2 = 0$, $l_i = -\infty$, and $u_i = +\infty$

Equality constrained LP/NLP:----$m_1 > 1$; $m_2 = 0$

Inequality constrained LP/NLP:----$m_1 = 0$; $m_2 > 1$

Mixed inequality constrained LP/NLP:----$m_1 > 1$; $m_2 > 1$

Bounded LP/NLP:----$m_1 = 0$; $m_2 = 0$, $l_i > -\infty$, and $u_i < +\infty$

Optimization Methods

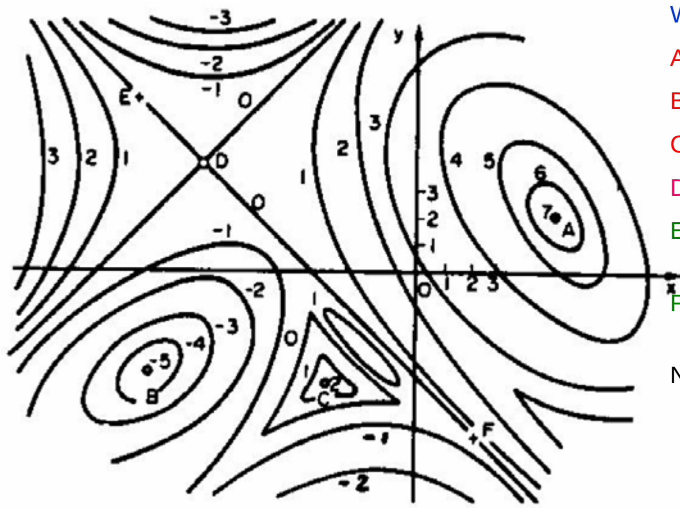
Ways to solve optimization problems

- 2D problems may be solved graphically or by common sense
- Simple and some well structured problems may be solved analytically
- Most will be solved numerically

Solving NLP graphically (2D)

- Sketch the feasible set on the x_1-x_2 plane
- Draw contours (isovalue curves) of $f(x)$
- Find the contour with the smallest value of $f(x)$ that "intersects or touches" the feasible set. The intersecting point(s) is the optimal solution $\mathbf{x}^* = (x_1^*, x_2^*)^T$ and the value of the optimal contour is $f^* = f(\mathbf{x}^*)$.

Example Contours of an objective function



With no constraint:

A= local max

B= local min

C = local max

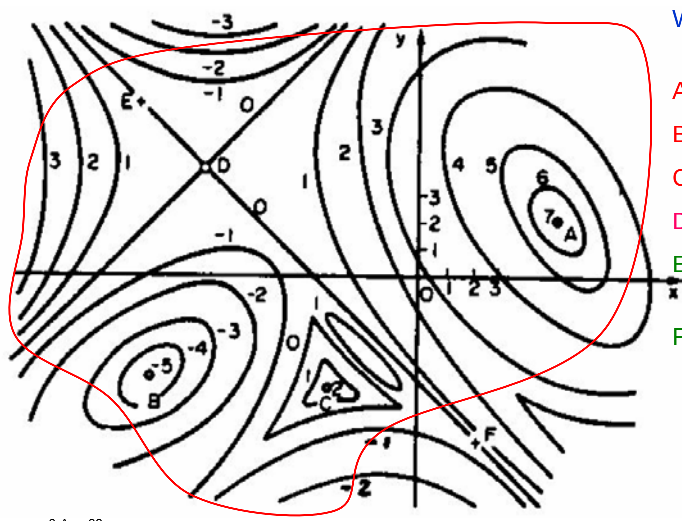
D = point of inflection

E = not a stationary point

F = not a stationary point

No global max or global min

Example Contours of an objective function



With constraint (inside red curve):

A= local max (global)

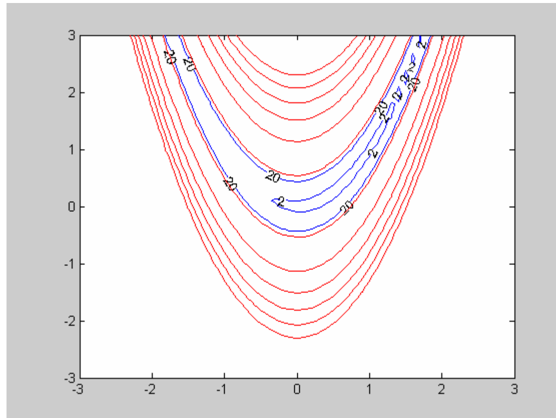
B= local min (global)

C = local max

D = point of inflection

E = not a stationary point

F = infeasible point



MATLAB code to draw contours of funcgrid

```
x1=[-3:0.1:3];
x2=[-3:0.1:3];
[X1 X2]=meshgrid(x1,x2);
f=funcgrid(X1,X2);
[cf,handf]=contour(x1,x2,f,[0,2,20],'b-');
clabel(cf,handf);
hold on;
[cf1,handf1]=contour(x1,x2,f,[30:100:600],'r-');
axis([-3,3,-3,3])
```

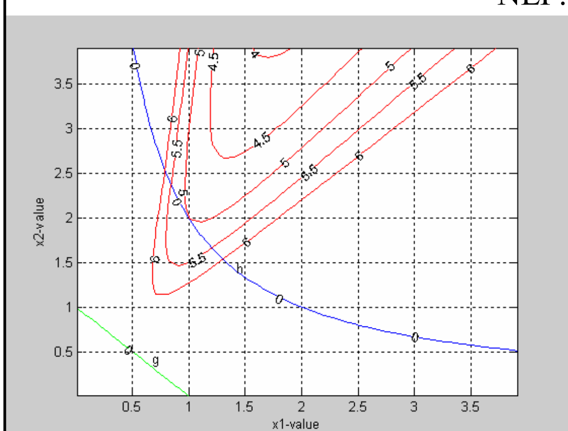
funcgrid=$(x_2 - x_1^2)^2 + (1 - x_1)^2$

Example: Solving NLP graphically

NLP: $\min_{\mathbf{x}\in R^2} f(\mathbf{x}) = \frac{6x_1}{x_2} + \frac{x_2}{x_1^2}$

$s.t.\quad h(\mathbf{x}) = x_1x_2 - 2 = 0$

$g(\mathbf{x}) = -x_1 - x_2 + 1 \le 0$



Optimal Solution is:

$x_1^* = 1;\ x_2^* = 2;$

$f^* = 5$

NLPex

Example: Solving NLP by Inspection

NLP: $\min_{\mathbf{x}\in R^3} f(\mathbf{x}) = -3x_1 + 4x_2 - x_3 + 2x_1x_2 - x_2x_3 + 2x_1x_2x_3$

$s.t.\quad 0 \le x_1 \le 1;\ 0 \le x_2 \le 1;\ 0 \le x_3 \le 1$

Since: $f(\mathbf{x}) = -3x_1 + 4x_2 - x_3 + 2x_1x_2 - x_2x_3 + 2x_1x_2x_3$

$= -3x_1 - x_3 + \left(4 + 2x_1 - x_3 + 2x_1x_3\right)x_2$

and since: $4 + 2x_1 - x_3 + 2x_1x_3 > 0$ for $0 \le x_i \le 1,\ i=1,2,3 \Rightarrow x_2{*} = 1$

Now we must choose x_1 and x_3 to maximize

$f(x_1, x_3, x_2 = 1) = 4 - 3x_1 - x_3 + 2x_1 - x_3 + 2x_1x_3 = 4 - x_1 - 2x_3 + 2x_1x_3$

Again we note that: $f(x_1, x_3, x_2 = 1) = 4 - x_1 - 2x_3(1 - x_1)$

and since: $1 - x_1 \ge 0$ for all values of $0 \le x_1 \le 1,\quad x_3{*} = 0$

Now we must choose x_1 to maximize $f(x_1, x_2 = 1, x_3 = 0) = 4 - x_1$

Clearly, $x_1{*} = 0$

Hence, our optimal solution is
$x_1{*} = 0, x_2{*} = 1, x_3{*} = 0, f{*} = 4$

Solving NLP Analytically: Optimality Conditions

Unconstrained optimization problems:

min $f(\mathbf{x}),\ \mathbf{x} \in R^n$

Equality constrained optimization problems:

min $f(\mathbf{x}),\ h_j(\mathbf{x}) = 0,\ j=1,..,m_1\ \ \mathbf{x} \in R^n$

Mixed equality constrained optimization problems:

min $f(\mathbf{x})$,

$h_j(\mathbf{x}) = 0,\ j=1,..,m_1$

$g_j(\mathbf{x}) = 0,\ j=1,..,m_2$

$\mathbf{x} \in R^n$

Characterizing Optimal Points: Unconstrained Problems

If at $\mathbf{x}^* \in R^n$,

i) $\nabla f(\mathbf{x}^*) = 0$

ii) $\mathbf{h}^T \nabla^2 f(\mathbf{x}^*)\mathbf{h} > 0$ for $\mathbf{h} \neq 0$ in R^n (i.e.$\nabla^2 f(\mathbf{x}^*)$ is pd)

Then $\mathbf{x}^*$ is a strict local minimizer of $f(\mathbf{x})$.

More $\mathbf{x}^*$ is a unique global minimizer of $f(\mathbf{x})$ if $f(\mathbf{x})$ is convex.

Note: Results are based on analysis of Taylor's series expansion:

$$f(\mathbf{x}^*+\mathbf{h}) = \underbrace{f(\mathbf{x}^*)}_{constant} + \underbrace{\nabla f(\mathbf{x}^*)\mathbf{h}}_{1st-order\,(linear)\,term} + \underbrace{\frac{1}{2}\mathbf{h}^T \nabla^2 f(\mathbf{x}^*)\mathbf{h}}_{2nd-order\,(quadratic)\,term} + \underbrace{o\left(\|\mathbf{h}\|^2\right)}_{higher-order\,terms}$$

Tests for Sign Definiteness of Matrix

The "sign definiteness" properties of a symmetric matrix are given by the following definitions:

The symmetric matrix **A** is

• positive definite (*pd*)	if and only if	$\mathbf{h}^T\mathbf{A}\mathbf{h} > 0$ for <u>all</u> $\mathbf{h} \in R^n$, $\mathbf{h} \neq \mathbf{0}$
• positive semidefinite (*psd*)	if and only if	$\mathbf{h}^T\mathbf{A}\mathbf{h} \geq 0$ for <u>all</u> $\mathbf{h} \in R^n$
• negative definite (*nd*)	if and only if	$\mathbf{h}^T\mathbf{A}\mathbf{h} < 0$ for <u>all</u> $\mathbf{h} \in R^n$ $\mathbf{h} \neq \mathbf{0}$
• negative semidefinite (*nsd*)	if and only if	$\mathbf{h}^T\mathbf{A}\mathbf{h} \leq 0$ for <u>all</u> $\mathbf{h} \in R^n$
• indefinite (*id*)	if and only if	$\mathbf{h}^T\mathbf{A}\mathbf{h} > 0$ for <u>some</u> $\mathbf{h} \in R^n$ and $\mathbf{h}^T\mathbf{A}\mathbf{h} < 0$ for <u>some</u> $\mathbf{h} \in R^n$

Sign Definiteness of Symmetric Matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & ,.., & a_{1n} \\ a_{21} & a_{22} & ,.., & a_{2n} \\ . & . & . & . \\ a_{1n} & a_{2n} & ,.., & a_{nn} \end{bmatrix}, \quad \mathbf{A} \text{ is } n \times n \text{ symmetric}$$

$\mathbf{h}^T\mathbf{A}\mathbf{h} = a_{11}h_1^2 + .. + a_{nn}h_n^2 + 2a_{12}h_1h_2 + 2a_{13}h_1h_3 + .. + 2a_{1n}h_1h_n + 2a_{23}h_2h_3 + .. + 2a_{n-1,n}h_{n-1}h_n$

This quadratic form is clearly a scalar quantity.

Sign Definiteness of Symmetric Matrix

Since A is symmetric, there exists an $n \times n$ orthonormal $\mathbf{P}$ such that

$$\mathbf{h}^T\mathbf{A}\mathbf{h} = \mathbf{h}^T\left(\mathbf{P}^T\mathbf{D}\mathbf{P}\right)\mathbf{h} = \sum_{j=1}^{n} \lambda_j \left(c_{1j}h_1 + \cdots + c_{nj}h_n\right)^2$$

where $\mathbf{D} = \text{diag}(\lambda_1, \lambda_2, .., \lambda_n)$, and $\lambda_1, \lambda_2, .., \lambda_n$ are real eigenvalues of $\mathbf{A}$

Thus

$\mathbf{A}$ is *psd* $\Leftrightarrow \lambda_1, \lambda_2, .., \lambda_n \geq 0$ with some $\lambda_i = 0$

$\mathbf{A}$ is *pd* $\Leftrightarrow \lambda_1, \lambda_2, .., \lambda_n > 0$

$\mathbf{A}$ is *nsd* $\Leftrightarrow \lambda_1, \lambda_2, .., \lambda_n \leq 0$ with some $\lambda_i = 0$

$\mathbf{A}$ is *nd* $\Leftrightarrow \lambda_1, \lambda_2, .., \lambda_n < 0$

$\mathbf{A}$ is *id* $\Leftrightarrow$ some $\lambda_1, \lambda_2, .., \lambda_n > 0$ and some < 0

Sign Definiteness of Symmetric Matrix Sylvester's Theorem

$$\text{Symmetric } \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & ,..., & a_{1n} \\ a_{21} & a_{22} & ,..., & a_{2n} \\ . & . & . & . \\ a_{1n} & a_{2n} & ,..., & a_{nn} \end{bmatrix}$$

Leading principal minors: $\alpha_1 = a_{11},\ \alpha_2 = \det\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, ...,\ \alpha_n = \det \mathbf{A}$

$\mathbf{A}$ is *pd* $\Leftrightarrow \alpha_1, \alpha_2, .., \alpha_n > 0$

$\mathbf{A}$ is *nd* $\Leftrightarrow \alpha_1 < 0, \alpha_2 > 0, \alpha_3 < 0, \lambda_4 > 0, ...$

$\mathbf{A}$ is *id* $\Leftarrow \lambda_i < 0$ for some even i

Convexity of a Function

Given a function $f(\mathbf{x}),\ \mathbf{x} \in R^n$,

Gradient of $f(\mathbf{x}) = \nabla f(\mathbf{x}) = \mathbf{g}^T = \left(\frac{\partial \mathbf{x} f(\mathbf{x})}{\partial x_1}, ..., \frac{\partial \mathbf{x} f(\mathbf{x})}{\partial x_n} \right)$

$$\textit{Hessian} \text{ of } f(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial^2 x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & . & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\mathbf{x})}{\partial^2 x_2} & . & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_n} \\ . & . & . & . \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_n} & . & \frac{\partial^2 f(\mathbf{x})}{\partial^2 x_n} \end{bmatrix}$$

$f(\mathbf{x})$ is convex $\Leftrightarrow$ its Hessian is *psd* for all $\mathbf{x} \in R^n$

$f(\mathbf{x})$ is strictly convex $\Leftarrow$ its Hessian is *pd* for all $\mathbf{x} \in R^n$

$f(\mathbf{x})$ is concave $\Leftrightarrow$ its Hessian is *nsd* for all $\mathbf{x} \in R^n$

$f(\mathbf{x})$ is strictly concave $\Leftarrow$ its Hessian is *nd* for all $\mathbf{x} \in R^n$

$f(\mathbf{x})$ is not convex nor concave $\Leftrightarrow$ its Hessian is *id* for all $\mathbf{x} \in R^n$

Summary: Some well known facts for unconstrained problems

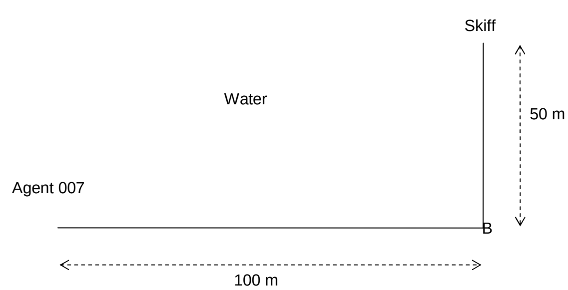
Optimality Conditions for Unconstrained optimization problems:

$\min f(\mathbf{x}),\ \mathbf{x} \in R^n$

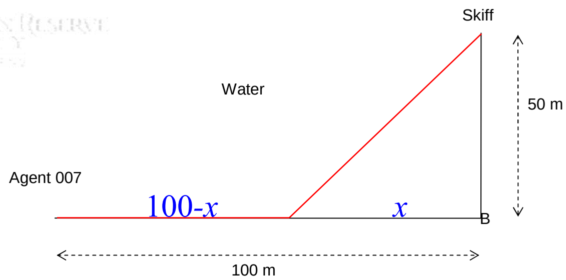
- If x* is a local minimizer of *f*, then
  $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*)$ is *positive semi-definite* (*psd*)
- If $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*)$ is *positive definite* (*pd*), then x* is a strict local minimizer of *f*
- If *f* is convex $\nabla^2 f(\mathbf{x}^*)$ is *pd for all* **x**, then any local minimizer is global

Example UNC 1

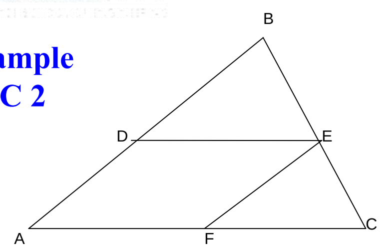
To save the world from destruction, James Bond 007 must reach a skiff 50 meters off shore from a point 100 meters away on a straight beach (point B) and then disarm a timing device. The agent can run along on the shore on the shifting sand at 5 meters per second, swim at 2 meters per second and disarm the timing device in 30 seconds. The device is set to trigger destruction in 74 seconds. Is it possible for the agent to succeed?



Example UNC 1



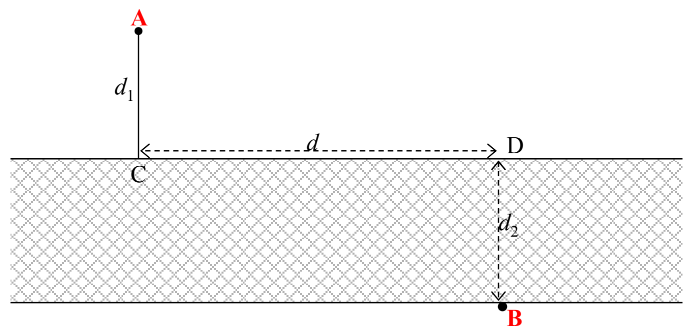
Example UNC 2



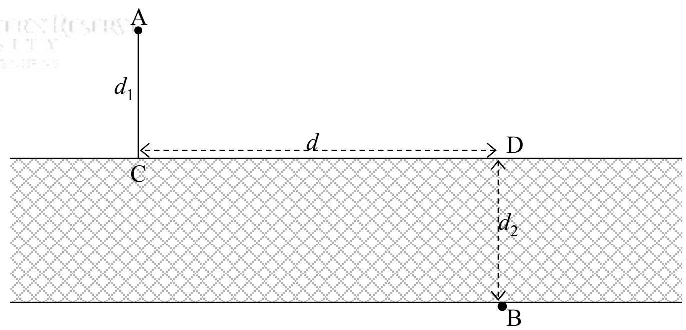
Given a triangle ABC, find a parallelogram ADEF (with D on AB, E on BC and F on AC as shown) that has the largest area possible. Formulate and solve this as an unconstrained optimization.

Example UNC 3: Snell's Law

Jaew is now at point A which is d_1 meters to the nearest point (C) on the shore of the nearby river. She wants to travel to B, a point on the opposite shore of the river. The river has a relatively constant width of d_2 meters and the distance from C to the point D directly across the river from B is d. Jaew can run on land at the speed of v_1 m/s, and can swim in a calm water at v_2 m/s. Find the best route for Jaew to travel in minimum time.

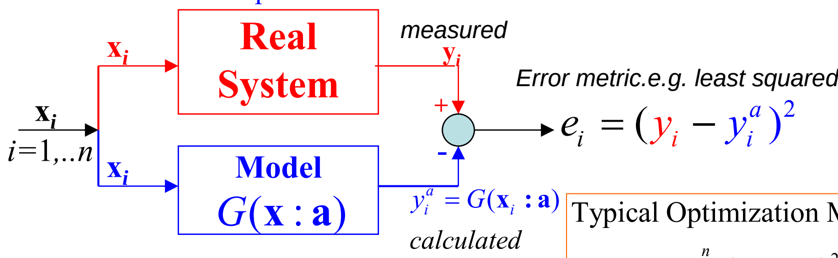


Example UNC 3



Example: Model Calibration/Parameter Estimation

In building a mathematical model of a complex system/process, the model structure (i.e. forms of mathematical relationships) is known, but the model parameters (for specific applications) are unknown and need to be estimated. Here model calibration has to be performed. This involves designing experiments, measuring outputs for the set of designed inputs, and estimating the values of parameters to best fit the calculated outputs to the measured outputs.



Typical Optimization Model

$$\min_{\mathbf{a}\in R^p} = \sum_{i=1}^{n}\left(y_i - y_i^a\right)^2$$

For example if G is affine

i.e. $y = \mathbf{a}^T\mathbf{x} + b$ Then, $y_i^a = \mathbf{a}^T\mathbf{x}_i + b$, and

the least squared error $e(\mathbf{a},b) = \sum_{i=1}^{n}\left(\mathbf{a}^T\mathbf{x}_i + b - y_i\right)^2$

So we choose the model parameters $(\mathbf{a},b)$ to minimize $e(\mathbf{a},b)$

This is what is typically called the least square estimator.

CASE SCHOOL OF ENGINEERING

For an affine G

Squared error: $e(\mathbf{a},b) = \sum_{i=1}^{n}\left(\mathbf{a}^T\mathbf{x}_i + b - y_i\right)^2$

First $\frac{de(\mathbf{a},b)}{db} = 2\sum_{i=1}^{n}\left(\mathbf{a}^T\mathbf{x}_i + b - y_i\right) = 0 \Rightarrow \mathbf{a}^T\left(\sum_{i=1}^{n}\mathbf{x}_i\right) + nb - \sum_{i=1}^{n} y_i = 0$

With $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$; we have $\mathbf{a}^T\bar{\mathbf{x}} + b - \bar{y} = 0$ ---(1)

Next : $\frac{de(\mathbf{a},b)}{d\mathbf{a}} = 2\sum_{i=1}^{n}\left(\mathbf{a}^T\mathbf{x}_i + b - y_i\right)\mathbf{x}_i = 0 \Rightarrow \bar{\mathbf{X}}\mathbf{a} + b\bar{\mathbf{x}} - \overline{y\mathbf{x}} = 0$ ---(2)

where $\mathbf{X}_i = \mathbf{x}_i\mathbf{x}_i^T$; $\bar{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_i$; and $\overline{y\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} y_i\mathbf{x}_i$

Thus the least squared estimator $(\hat{\mathbf{a}},\hat{b})$ is a solution of the $(n+1)\times(n+1)$ linear system:

$$\begin{pmatrix}\bar{\mathbf{x}}^T & 1\\ \bar{\mathbf{X}} & \bar{\mathbf{x}}\end{pmatrix}\begin{pmatrix}\mathbf{a}\\ b\end{pmatrix} = \begin{pmatrix}\bar{y}\\ \overline{y\mathbf{x}}\end{pmatrix}$$

CASE SCHOOL OF ENGINEERING

Numerical Optimization Methods

- Most optimization problems will have to be solved by numerical methods
- A numerical method is an iterative procedure in which a sequence of points--$\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$,.., $\mathbf{x}^{(k)}$,..is generated from an initial point $\mathbf{x}^{(1)}$, and hopefully converging to an optimal point $\mathbf{x}^*$

Numerical Optimization Methods

We look for methods which are

- Effective: Find **x*** every time, always stop at the right solution (convergence issue: Stop or not, right point or not)
- Efficient: Find **x*** quickly (speed of convergence issue: measured by # of iterations)
- Cheap: Low cost per iteration (time: # of function evaluations; storage; errors)

Numerical Optimization Methods

2 basic steps:

At a typical iteration k with iterate $\mathbf{x}^{(k)}$

- Find the direction of search $\mathbf{d}^{(k)}$ emanating from $\mathbf{x}^{(k)}$---Direction-finding problem
- Find how far to move along $\mathbf{d}^{(k)}$ --Line search problem to find step-size $\alpha^{(k)}$

Update: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$

Do until termination

Termination Criteria

- For unconstrained problems, $\mathbf{x}^{(k)}$ is a local minimizer if

  $\nabla f(\mathbf{x}^{(k)}) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^{(k)})$ is *positive definite*
- We may also want to stop if there is no significant change in $f(\mathbf{x}^{(k)})$ and/or $\mathbf{x}^{(k)}$ from one iteration to the next.
- These combined with attempts to overcome various numerical difficulty regarding scaling and units lead to the following combined termination criteria:

Termination Criteria

Some or all of the following must be met:

1) $\|\nabla f(\mathbf{x}^{(k)})\| \le \varepsilon_1(1+|f(\mathbf{x}^{(k)})|)$
2) $f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k+1)}) \le \varepsilon_2(1+|f(\mathbf{x}^{(k)})|)$
3) $\|\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)}\| \le \varepsilon_3(1+|\mathbf{x}^{(k)}|)$
4) $\nabla^2 f(\mathbf{x}^{(k)}) + \varepsilon_4 I$ is *positive semi-definite*

Where

- (4) would only be performed if $\nabla^2 f(\mathbf{x}^{(k)})$ is available
- ε should be chosen appropriately based on machine accuracy

Termination Criteria

For example: if a machine arithmetic is accurate up to 16 digits, then

$\varepsilon_2 = 10^{-16}$

$\varepsilon_1 = \varepsilon_3 = \sqrt{\varepsilon_2} = 10^{-8}$

and

choose $\varepsilon_4 = \varepsilon_2 \|\nabla^2 f(\mathbf{x}^{(k)})\|$

Direction-Finding Methods

Direction-Finding

- Given the current iterate $\mathbf{x}^{(k)}$, what should be the direction $\mathbf{d}^{(k)}$ to move from $\mathbf{x}^{(k)}$?
- Desirable properties of $\mathbf{d}^{(k)}$
  - Cheap to compute (time and storage)
  - Lead to good convergence properties
    - descent (improving) ⇔ $\nabla f(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)} < 0$
    - No sudden changes (closeness)

Direction-Finding Methods

Given $\mathbf{x}^{(k)}$, let

$\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)}) = (\partial f/\partial x_1,..,\partial f/\partial x_n)^T =$ **gradient of** f

and let $\mathbf{p}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})$ **and** $\mathbf{q}^{(k)} = \mathbf{g}^{(k)} - \mathbf{g}^{(k-1)}$

Steepest descent (SD): $\mathbf{d}^{(k)} = -\mathbf{g}^{(k)}$ *negative gradient*

- Descent: If $\mathbf{g}^{(k)} \neq \mathbf{0}$, $\nabla f(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)} = -\|\mathbf{g}^{(k)}\|^2 < 0$
- f decreases at fastest rate
- Good when start far from $\mathbf{x}^*$, but very slow when close to $\mathbf{x}^*$ since $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)}) \approx \mathbf{0}$

Newton-Raphson Method (NR)

NR: $\mathbf{d}^{(k)} = -\mathbf{H}(\mathbf{x}^{(k)})^{-1}\mathbf{g}^{(k)}$

where $\mathbf{H}(\mathbf{x}^{(k)})$ is Hessian of f at $\mathbf{x}^{(k)}$ ..symmetric

- Descent, if $\mathbf{H}(\mathbf{x}^{(k)})$ is positive definite (*pd*) & $\mathbf{g}^{(k)} \neq 0$
  $\nabla f(x^{(k)})^T d^{(k)} = -\mathbf{g}^{(k)T}\mathbf{H}(\mathbf{x}^{(k)})\,\mathbf{g}^{(k)} < 0$ since $\mathbf{H}(\mathbf{x}^{(k)})$ is pd
- Very good if start from near x*, i.e. very good when it works.
- May diverge if $\mathbf{x}^{(0)}$ is poor
- Very expensive since Hessian and its inverse and hence second derivatives have to be computed.

Conjugate Directions

CG: $\mathbf{d}^{(k)} = -\mathbf{g}^{(k)} + \beta_k\,\mathbf{d}^{(k-1)}$

where $\beta_k = \|\mathbf{g}^{(k)}\|^2/\|\mathbf{g}^{(k-1)}\|^2$ …*Fletcher-Reeve (FR)*

or $\beta_k = \mathbf{g}^{(k)T}(\mathbf{g}^{(k)} - \mathbf{g}^{(k-1)})/\|\mathbf{g}^{(k-1)}\|^2$ ..*Polak-Ribere (FR)*

- Descent if optimal line search used:
  $\nabla f(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})^T\mathbf{g}^{(k)} + \beta_k\,\nabla f(\mathbf{x}^{(k)})^T\mathbf{d}^{(k-1)} = -\|g^{(k)}\|^2 < 0$
- Superlinear convergence (performance is between SD and NR
- Low storage requirement $\Rightarrow$ good for high *n* problems
- Quadratic convergence and PR is usually better

Quasi Newton Methods

QN: $\mathbf{d}^{(k)} = -\mathbf{H}^{(k)}\mathbf{g}^{(k)}$

where $\mathbf{H}^{(1)}$ is *pd*, and subsequent $\mathbf{H}^{(k)}$ are also *pd*.

Let $\mathbf{p}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}$ and $\mathbf{q}^{(k)} = \mathbf{g}^{(k)} - \mathbf{g}^{(k-1)}$

$$\mathbf{H}^{(k)} = \mathbf{H}^{(k-1)} + \frac{\mathbf{p}^{(k)}\mathbf{p}^{(k)T}}{\mathbf{p}^{(k)T}\mathbf{q}^{(k)}} - \frac{\mathbf{H}^{(k-1)}\mathbf{q}^{(k)}\mathbf{q}^{(k)T}\mathbf{H}^{(k-1)}}{\mathbf{q}^{(k)T}\mathbf{H}^{(k-1)}\mathbf{q}^{(k)}} \qquad \text{(DFP)}$$

or

$$\mathbf{H}^{(k)} = \mathbf{H}^{(k-1)} + \left(1 + \frac{\mathbf{q}^{(k)T}\mathbf{H}^{(k-1)}\mathbf{q}^{(k)}}{\mathbf{p}^{(k)T}\mathbf{q}^{(k)}}\right)\frac{\mathbf{p}^{(k)}\mathbf{p}^{(k)T}}{\mathbf{p}^{(k)T}\mathbf{q}^{(k)}} - \frac{\mathbf{p}^{(k)}\mathbf{q}^{(k)T}\mathbf{H}^{(k-1)} + \mathbf{H}^{(k-1)}\mathbf{q}^{(k)}\mathbf{p}^{(k)T}}{\mathbf{p}^{(k)T}\mathbf{q}^{(k)}} \qquad \text{(BFGS)}$$

Quasi Newton Methods

- **Descent if $\mathbf{H}^{(k)}$ are kept pd.**
  $\nabla f(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)} = -\mathbf{g}^{(k)T}\mathbf{H}^{(k)}\mathbf{g}^{(k)} < 0$ since $\mathbf{H}^{(k)}$ is pd
- **If $\mathbf{H}^{(k-1)}$ is *pd,* so is $\mathbf{H}^{(k)}$ if $\mathbf{p}^{(k)T}\mathbf{q}^{(k)} > 0$ ..guaranteed by an optimal line search or Wolfe or Wolfe Powell tests**
- **Superlinear convergence, very good performance generally better than conjugate directions(*CD*) methods**
- **Higher storage requirement than *CD* methods**
- **Quadratic convergence**
- **Best methods for all problems except large ones**

Trust-Region Method

Here we use quadratic approximation of f(x) at $x^{(k)}$ and find both the direction of search and step size by solving a quadratic programming (QP) problem:

min $q(\mathbf{d}) = f(\mathbf{x}^{(k)}) + \nabla f(\mathbf{x}^{(k)})\mathbf{d} + 0.5(\mathbf{d}^T \nabla^2 f(\mathbf{x}^{(k)})\mathbf{d})$

s.t $-h^{(k)} \leq d_i \leq h^{(k)},\ i = 1,\ldots,n$

Then we set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$

The next step-size $h^{(k+1)}$ is then determined by examining how well the quadratic function q(.) approximates the true function f at $x^{(k+1)}$.

Trust-Region Method

Compute:

$r^{(k)} = \Delta f/\Delta q$

where

$\Delta f = f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k+1)})$

$\Delta q = f(\mathbf{x}^{(k)}) - q(\mathbf{d}^{(k)})$

If $r^{(k)} < 0.25$, **set** $h^{(k+1)} = ||\mathbf{d}^{(k)}||/4$

If $r^{(k)} > 0.75$ **and** $h^{(k)} = ||\mathbf{d}^{(k)}||$, **set** $h^{(k+1)} = 2h^{(k)}$

Otherwise, set $h^{(k+1)} = h^{(k)}$

Then, proceed to iteration $k+1$

Until a termination criterion is met.

Nongradient-based: Nelder-Mead

Basic idea:

- Create a regular simplex (n+1 vertices and its convex hull)
- Role it around toward optimum expanding, contracting or reflecting as appropriate
- Until the size of the simplex is small enough to fit a ball of size ε
- Direction finding and line search are done simultaneously

Line Search Methods

CASE SCHOOL OF ENGINEERING

Line search problems

Given $\mathbf{x}^{(k)}$ and a search direction $\mathbf{d}^{(k)}$, find how far to move along $\mathbf{d}^{(k)}$--i.e. find a step-size $\alpha^{(k)}$ such that $f(\mathbf{x}^{(k+1)})$ meets some criteria of improvement.

At a point $\mathbf{x}$ along $\mathbf{d}^{(k)}$: $\mathbf{x} = \mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}$

⇒ α is the distance from $\mathbf{x}$ to $\mathbf{x}^{(k)}$ along $\mathbf{d}^{(k)}$.

⇒ $f(\mathbf{x}) = f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) = \phi(\alpha)$ a function of one variable only.

CASE SCHOOL OF ENGINEERING

Line Search Methods

For example, given

$\boldsymbol{f}(\mathbf{x}) = 3x_1^2 + x_1x_2 + 2x_2^2 - 8x_1,$

and $\mathbf{x}^{(1)} = (-1,1)^T$ **,** $\mathbf{d}^{(1)} = (2,-1)^T$

⇒ $\mathbf{x} = \mathbf{x}^{(1)} + \alpha\mathbf{d}^{(1)} = (-1,1)^T + \alpha(2,-1)^T$

$= (-1 + 2\alpha, 1 - \alpha)^T$

⇒ $\phi(\alpha) = \boldsymbol{f}(\mathbf{x}^{(1)} + \alpha\mathbf{d}^{(1)})$

$= 3(-1 + 2\alpha)^3 + (-1 + 2\alpha)(1 - \alpha) + 2(1 - \alpha)^2 - 8(-1 + 2\alpha)$

Line Search Methods

If $f(\mathbf{x})$ is unimodal $\Rightarrow$ $\phi(\alpha)$ is unimodal

Also $\quad \phi'(\alpha) = \nabla f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})\mathbf{d}^{(k)} \quad$ (1)

$\Rightarrow \quad \phi'(0) = \nabla f(\mathbf{x}^{(k)})\mathbf{d}^{(k)} \quad$ **(2)**

$\Rightarrow \quad \phi'(\alpha^{(k)}) = \nabla f(\mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)})\mathbf{d}^{(k)}$

$= \nabla f(\mathbf{x}^{(k+1)})\mathbf{d}^{(k)} \quad$ **(3)**

Thus

$\mathbf{d}^{(k)}$ is a descent direction $\nabla f(\mathbf{x}^{(k)})\mathbf{d}^{(k)} < 0 \Leftrightarrow \phi'(0) < 0$

e.g. $\phi'(0) = 18(-1 + 2\alpha)^2 + 2(1 - \alpha) - (1 + 2\alpha) - 4(1 - \alpha) - 16$

Thus $\phi'(0) = 1 > 0 \Rightarrow \mathbf{d}^{(k)}$ is not a descent direction.

Line Search Methods

In numerical optimization, we require that $\mathbf{d}^{(k)}$ be a descent direction to ensure convergence

There are two types of line search methods:

1. **Inaccurate** line-search: an "acceptable" step-size is sought, using some kind of "acceptability" tests.
2. **Optimal** (also known as accurate or exact) line-search: Here a step-size that gives the "best" value of $f(\mathbf{x})$ along $\mathbf{d}^{(k)}$ is sought. This is equivalent to solving: $\min_{\alpha \geq 0} \phi(\alpha)$

Line Search Methods

Note:

- **Optimal line-search:** $\min_{\alpha\geq 0} \phi(\alpha)$ **is "almost" an unconstrained problem except for** $\alpha \geq 0$**, and**
- **If** $\alpha^{(k)}$ **is an optimal step-size with** $\alpha^{(k)} > 0$**, then** $\phi'(\alpha^{(k)}) = \nabla f(\mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)})\mathbf{d}^{(k)} = \nabla f(\mathbf{x}^{(k+1)})\mathbf{d}^{(k)} = 0$

$\Rightarrow$ **an optimal line-search will produce a new point** $\mathbf{x}^{(k+1)}$ **at which the gradient of** f **is orthogonal to the direction of the search d**

Inaccurate Line Search

The step-size α^* **is considered "acceptable" if it is not too large and not too small.**

Not-Too-Large Test: **For a given** $0< \rho <0.5$

$$\phi(\alpha^*) \leq \phi(0) + \rho\phi'(0)\alpha^* \qquad \textbf{(NTL)}$$

Not-Too-Small Test:

Armijo's NTS: $\phi(\eta\alpha^*) > \phi(0) + \rho\phi'(0)\eta\alpha^*$, $\eta>1$

Goldstien's NTS: $\phi(\alpha^*) > \phi(0) + \sigma\phi'(0)\alpha$, $0.5< \sigma <1$

Wolfe's NTS: $\phi'(\alpha^*) > \sigma\phi'(0)$, $0.5 < \sigma <1$

Wolfe-Powell's NTS: $|\varphi'(\alpha^*)| < -\sigma\phi'(0)$, $0.5 < \sigma <1$

Inaccurate Line Search

EXAMPLE: Given $\varphi(\alpha) = e^{-2\alpha} + \alpha$, $\eta = 2$, $\rho = 0.1$, **and** $\sigma = 0.9$, **is** $\alpha^* = 0.5$ **is an acceptable step-size.**

$\phi(0) = 1$, $\phi'(\alpha) = -2e^{-2\alpha} + 1$, and $\phi'(0) = -1 < 0$.

NTL?: $\phi(0.5) = 0.8679$

$\phi(0) + \rho\phi'(0)\alpha^* = 1 + (0.1)(-1)(0.5) = 0.95$

$\Rightarrow \phi(\alpha) > \phi(0) + \rho\phi'(0)\alpha^* \Rightarrow \alpha^*$ **is not too large.**

NTS?: Armijo's: $\phi(\eta\alpha^*) = \phi(1) = 1.1353$

$\Rightarrow \phi(0) + \eta\phi'(0)\eta\alpha^* = 1 + (0.1)(-1)(0.5) = 0.95$

$\Rightarrow \phi(\eta\alpha^*) > \phi(0) + \rho\phi'(0)\eta\alpha^*$,

$\Rightarrow \alpha^*$ **is not too small according to Armijo's test.**

Inaccurate Line Search

Goldstein's NTS: $\phi(\alpha^*) = \phi(0.5) = 0.8679$

$\phi(0) + \sigma\phi'(0)\alpha^* = 1 + (0.9)(-1)(0.5) = 0.955$

$\Rightarrow \phi(\alpha^*) < \phi(0) + \sigma\phi'(0)\alpha^*$

$\Rightarrow \alpha^*$ **is too small according to Goldstein's test.**

Wolfe's NTS: $\phi'(\alpha^*) = \phi'(0.5) = 0.2642$

$\sigma\phi'(0) = (0.9)(-1) = -0.9 \Rightarrow \phi'(\alpha^*) > \sigma\phi'(0)$

$\Rightarrow \alpha^*$ **is not too small according to Wolfe's test.**

Wolfe-Powell's NTS: $|\phi'(\alpha^*)| = \phi'(0.5) = 0.2642$,

and $-\sigma\phi'(0) = -(0.9)(-1) = 0.9 \Rightarrow |\phi'(\alpha^*)| < -\sigma\phi'(0)$

$\Rightarrow \alpha^*$ **is not too small based on Wolfe-Powell's test.**

Accurate Line Search

Popular Methods:

- Interval Reduction Methods
  - Golden Section Search
  - Fibonacci
  - Quadratic Interpolation
  - Brent's
  - Others: Bisection, Equal Interval
- Approximation (Extrapolation) Methods
  - Newton's
  - Method of False Position or Sectioning Search
  - Bisection

Accurate Line Search

	Interval Reduction	Approximation Methods
Derivatives not available	• Golden Section GSS • Fibonacci • Quadratic Interpolation • Brent's method • Others: Bisection, Equal Interval , etc…	
Derivatives available	• Bisection	• Newton's • Method of False Position • Cubic Interpolation

Interval Reduction

- Find an initial bracket with one end at α_1
- Successively reduce the bracket until its length is within a desired tolerance limit

$\Rightarrow$ Any point in the final bracket $\approx \alpha^*$

- These methods consist of two phases:
  - the bracketing phase
  - the interval reduction phase.
- A bracket = an interval that is known to contain the true minimizer α^* with certainty

Brackets

- A bracket = an interval that is known to contain the true minimizer α^* with certainty
- $[a,b]$ = Bracket
  - if $\exists\, c \in (a,b) \ni \ \phi(a) > \phi(c)$ and $\phi(c) < \phi(b)$ or
  - If $\phi'(a) < 0$ and $\phi'(b) > 0$
- The first test is good if derivatives of $\phi(\alpha)$ are difficult or expensive to find
- The second is good if derivatives of $\phi(\alpha)$ are easy or cheap to find

Bracketing Procedure

Begin with α_1 (Given $x^{(k)}$ and direction $d^{(k)}$):

- Select a suitable step length Δ and verify that $d^{(k)}$ is descending, i.e. $\phi(\alpha_1+\Delta) < \phi(\alpha_1)$
- If $\phi(\alpha_1+\Delta) > \phi(\alpha_1)$ reset the step-size $\Delta \leftarrow -\Delta$
- If $\phi(\alpha_1+\Delta) = \phi(\alpha_1)$ the interval $[\alpha_1, \alpha_1+\Delta]$ represents an initial bracket and no further work is needed
- Then proceed as follows:

Bracketing Procedure

(if $\phi'(\alpha)$ is not available)

- **For** $k = 1,2,3,4,.....$

1: DO

$$\alpha_{k+1} = \alpha_k + 2^{k-1}\Delta$$

UNTIL

$$\phi(\alpha_k) < \phi(\alpha_{k+1})$$

2: Compute $\alpha_{try} = \alpha_k + 2^{k-2}\Delta$

If $\phi(\alpha_k) < \phi(\alpha_{try})$, **BRACKET** $= [\alpha_{k-1}, \alpha_k, \alpha_{try}]$

Else BRACKET $= [\alpha_k, \alpha_{try}, \alpha_{k+1}]$

Bracketing Procedure

min $\phi(\alpha) = \alpha^2 + 10/(\alpha+1)$ **given that** $\alpha_1 = 0$ **and** $\Delta = 0.2$**.**

k	$\alpha_{k+1} = \alpha_k + 2^{k-1}\Delta$	$\phi(\alpha_k)$	Comment
1	0	10	$\alpha_1 = 0$
2	0.2	8.3733	$\phi(\alpha_2) < \phi(\alpha_1) \Rightarrow$ retain $+\Delta$
3	0.6	6.61	
4	1.4	6.1267	
5	3.0	11.5	$\phi(\alpha_2) < \phi(\alpha_1) \Rightarrow$ step back
try	2.2	7.9650	

$\alpha_{\text{try}} = \alpha_4 + 2^2\Delta$

Since $\phi(\alpha_4) < \phi(\alpha_{try}) \Rightarrow$ our initial bracket is [0.6, 1.4, 2.2]

Bracketing Procedure

(if $\phi'(\alpha)$ is available)

- **If** $\phi'(\alpha_1) > 0$**, set** $\Delta \leftarrow -\Delta$ and proceed. Otherwise, proceed
- **For** k **= 1,2,3,4,.....**

1: DO

$\alpha_{k+1} = \alpha_k + 2^{k-1}\Delta$

UNTIL

$\phi'(\alpha_{k+1}) > 0$

2: Compute $\alpha_{try} = \alpha_k + 2^{k-2}\Delta$

If $\phi'(\alpha_{try}) > 0$ **, BRACKET =** $[\alpha_{k-1}, \alpha_k, \alpha_{try}]$

Else BRACKET = $[\alpha_k, \alpha_{try}, \alpha_{k+1}]$

Interval Reduction

Given an initial bracket, successively reduce the bracket to a desired length and location.

When $\phi'(\alpha)$ is not available:

- **Compare the values of $\phi(\alpha)$ at two distinct interior points in the bracket**
- **use the following rule to reduce the interval:**
  - **Let $x \in [a, b]$, and $y \in [a, b]$ with $x < y$.**
  - **If $\phi(x) > \phi(y)$, then $\alpha^* \in [x, b]$**
  - **Else $\alpha^* \in [a, x]$**

Golden Section search (GSS) and Fibonacci method

- **Use one function per iteration (except the first)**
- **Guarantee predictable rate of interval reduction by placing x_k and y_k so that the length of the next bracket $[a_k, y_k]$ or $[x_k, b_k]$ is the same**
- **Successively application yields the following relationship of lengths of three successive brackets**
  - $L_k = L_{k+1} + L_{k+2}$ **(1)**
  - **Note if $[a_{k+1}, b_{k+1}] = [x_k, b_k]$, $y_k \to x_{k+1}$ and y_{k+1} is the only new point generated at iteration $k+1$. Similarly if $[a_{k+1}, b_{k+1}] = [a_k, y_k]$, $x_k \to y_{k+1}$ and x_{k+1} is the only new point generated.**
- **Both GSS and the Fibonacci method make use of (1), but this is where the similarity ends.**

Golden Section search (GSS) and Fibonacci method

- **GSS**
  - a constant rate of interval reduction is desired and
  - the calculations are made using as many iterations as needed to reduce the bracket to a desired accuracy.
- **Fibonacci method**
  - the number of iterations is pre-determined and
  - an effort is made to make the best possible reduction within the predetermined number of iterations.

Golden Section search (GSS)

Let τ be the constant rate of interval reduction sought in GSS, i.e. $L_{k+1}/L_k = \tau$, for all k = 1,2,3,.....

Dividing (1) through by L_k ⇒

$$1 = L_{k+1}/L_k + (L_{k+2}/L_{k+1})*(L_{k+1}/L_k) = \tau + \tau^2$$

⇒ $\tau^2 + \tau - 1 = 0$

⇒**positive root of (2) is $\tau = (\sqrt{5} - 1)/2 = 0.618034$ ⇒ Golden Number (⇒ name Golden Section Search).**

⇒ $L_{k+1} = \tau L_k = 0.618 L_k = \tau^k L_k = 0.618^k L_1$

where $L_1 = b_1 - a_1$, the length of the initial bracket.

Golden Section search (GSS)

Typical calculations of the GSS method:

0: Given an initial bracket [a_1,b_1] and the interval of uncertainty ε

Compute $x_1 = b_1 - \tau L_1$; $\phi(x_1)$ and $y_1 = a_1 + \tau L_1$; $\phi(y_1)$

1: For $k = 1,2,3,\ldots$

DO

If $\phi(x_k) < \phi(y_k)$, **then**

set $a_{k+1} = a_k$; $b_{k+1} = y_k$; **and** $y_{k+1} = x_k$ **(hence** $\phi(y_{k+1}) = \phi(x_k)$**)**

Compute $L_{k+2} = \tau L_{k+1}$ **or** $\tau^{k+1} L_1$

Compute $x_{k+1} = b_{k+1} - L_{k+2}$

Compute $f(x_{k+1})$

Else

set $a_{k+1} = x_k$; $b_{k+1} = b_k$; **and** $x_{k+1} = y_k$

Compute $L_{k+2} = \tau L_{k+1}$ **or** $\tau^{k+1} L_1$

Compute $y_{k+1} = a_{k+1} + L_{k+2}$

Compute $f(y_{k+1})$

UNTIL

$L_k < \varepsilon$

2: Return

the final bracket $[a_k, b_k]$ **and**

$\alpha^* = (a_k + b_k)/2$ **or the best known interior point in the final bracket** $[a_k, b_k]$.

Golden Section search (GSS)

Example: Given $\phi(\alpha) = 2e^{-2\alpha} + \alpha$, **and** $\varepsilon = 0.1$, **and**

the initial bracket $[a_1, b_1]$ **= [0,1.2707]** ⇒ $L_1 = b_1 - a_1$ **= 1.2707**

k	$L_{k+1} = \tau^k L_1$	a_k	x_k	y_k	b_k	$f(x_k)$	$f(y_k)$
1	.7853	0	.4854	.7853	1.2707	1.2429	1.2011
2	.4854	.4854	.7853	.9706	1.2707	1.2011	1.2577
3	.2999	.4854	.6707	.7853	0.9706	1.1937	1.2011
4	.1854	.4854	.5999	.6707	0.7853	1.2024	1.1937
5	.1145	.5999	.6707	.7144	0.7853	1.1937	1.1936
6	.0708	.6707	.7144	.7415	0.7853	1.1936	1.1954

$< \varepsilon$ ⇒ STOP

⇒ **Final bracket** $[a_7, b_7]$ **= [0.6707,0.7415],**

⇒ **a good estimate of** x^***= 0.7144 or (0.6707+0.7415)/2 = 0.7061**.

Fibonacci Search

- **For a fixed number of iterations N, to find α^* to within the level of accuracy ε, use Fibonacci search**
- **Let $F_0=1$, $F_1=1$, and $F_k = F_{k-1} + F_{k-2}$ for $k = 2, 3,..$**
  where F_k's are known as Fibonacci numbers

$\Rightarrow$ **Require** $L_N = L^* \le \varepsilon$ **(also** $L_N = L_{N+1} = L_{N+2} = \ldots$**)**

$\Rightarrow L_{N-1} = L_N + L_{N+1} = L_N + L_N = F_0 L^* + F_1 L^* = F_2 L^*$

$\Rightarrow L_{N-2} = L_{N-1} + L_N = 2L_N + L_N = F_1 L^* + F_2 L^* = F_3 L^*$

.....

$\Rightarrow L_{N-k} = L_{N-k+1} + L_{N-k+2} = F_{k-1} L^* + F_k L^* = F_{k+1} L^*$

.....

$\Rightarrow L_2 = L_{N-(N-2)} = F_{N-1} L^*$

$\Rightarrow L_1 = L_{N-(N-1)} = F_N L^* \Rightarrow L^* = L_1/F_N$

Fibonacci Search

$\Rightarrow L_N = (F_1/F_N)L_1 \le \varepsilon$

$\Rightarrow L_{N-1} = (F_2/F_N)L_1$

$\Rightarrow L_{N-2} = (F_3/F_N)L_1$

.....

$\Rightarrow L_{N-k} = (F_{k+1}/F_N)L_1$ $\Rightarrow L_k = (F_{N-k+1}/F_N)L_1$

.....

$\Rightarrow L_2 = (F_{N-1}/F_N)L_1$

$\Rightarrow L_1 = (F_N/F_N)L_1$

where F_k's are Fibonacci numbers

$F_0=1$, $F_1=1$,

$F_2 = F_1 + F_0 = 2$, $F_3 = F_2 + F_1 = 3$, $F_4 = F_3 + F_2 = 5$, ...

Fibonacci Search

Typical calculations of the Fibonacci method:

0: **Given an initial bracket $[a_1,b_1]$ and the interval of uncertainty ε**

Find N: $L_N = L_1/F_N \le \varepsilon \Rightarrow F_N \ge L_1/\varepsilon \Rightarrow$ N can be found from Table of Fibonacci numbers

Then compute $L_2 = (F_{N-1}/F_N)L_1$; $x_1 = b_1 - L_2$; $\phi(x_1)$ and $y_1 = a_1 + L_2$; $\phi(y_1)$

1: **DO** **k = 1,2,3,…,N**

If $\phi(x_k) < \phi(y_k)$, then

set $a_{k+1} = a_k$; $b_{k+1} = y_k$; and $y_{k+1} = x_k$ (hence $\phi(y_{k+1}) = \phi(x_k)$)

Compute $L_{k+2} = (F_{N-k-1}/F_N)L_1$

Compute $x_{k+1} = b_{k+1} - L_{k+2}$

Compute $f(x_{k+1})$

Else

set $a_{k+1} = x_k$; $b_{k+1} = b_k$; and $x_{k+1} = y_k$

Compute $L_{k+2} = (F_{N-k-1}/F_N)L_1$

Compute $y_{k+1} = a_{k+1} + L_{k+2}$

Compute $f(y_{k+1})$

CONTINUE

2: **Return the final estimate** $x_N = y_N \approx \alpha^*$ (This assumes small calculation errors.)

Quadratic Interpolation

Given the current bracket $[a_k, x_k, b_k]$

and the values $\phi(a_k)$, $\phi(x_k)$, $\phi(b_k)$

we could approximate by a quadratic curve

$$q(\alpha) = c_0 + c_1\alpha + c_2\alpha^2$$

passing through the same 3 points as $\phi(\alpha)$:

$$[a_k, \phi(a_k)], [x_k, \phi(x_k)], [b_k, \phi(b_k)]$$

i.e.

$$q(a_k) = c_0 + c_1 a_k + c_2 a_k^2 = \phi(a_k)$$
$$q(x_k) = c_0 + c_1 x_k + c_2 x_k^2 = \phi(x_k)$$
$$q(b_k) = c_0 + c_1 b_k + c_2 b_k^2 = \phi(b_k)$$

Quadratic Interpolation

We can solve for c_0, c_1, c_2 and find a minimizer y_k of $q(\alpha) = c_0 + c_1\alpha + c_2\alpha^2$ (by finding the root of $q'(\alpha) = 0$) to get:

$$y_k = \frac{1}{2}\left(\frac{(b_k^2 - c_k^2)\phi(a_k) + (c_k^2 - a_k^2)\phi(b_k) + (a_k^2 - b_k^2)\phi(c_k)}{(b_k - c_k)\phi(a_k) + (c_k - a_k)\phi(b_k) + (a_k - b_k)\phi(c_k)}\right)$$

We can now check whether we can use y_k to reduce the interval and form a smaller bracket

e.g. if $a_k < y_k < x_k$ and is not too close to any either a_k or x_k and if $\phi(y_k), > \phi(x_k)$ then the new bracket is $[y_k, x_k, b_k]$ (i.e. $[a_{k+1}, x_{k+1}, b_{k+1}] \leftarrow [y_k, x_k, b_k]$)

Quadratic Interpolation

Typical calculations of the QI method:

0: Given an initial bracket $[a_1, x_1, b_1]$ and the interval of uncertainty ε

1: For k = 1,2,3,...

DO

Compute $\phi(a_k)$, $\phi(x_k)$, $\phi(b_k)$

Compute y_k, and $\phi(y_k)$

If $y_k > x_k$ and

If $\phi(y_k) \le \phi(x_k)$, then set $a_{k+1} = x_k$; $x_{k+1} = y_k$; $b_{k+1} = b_k$

Else set $a_{k+1} = a_k$; $x_{k+1} = x_k$; $b_{k+1} = y_k$

If $y_k < x_k$ and

If $\phi(y_k) \le \phi(x_k)$, then set $a_{k+1} = a_k$; $x_{k+1} = y_k$; $b_{k+1} = x_k$

Else set $a_{k+1} = y_k$; $x_{k+1} = x_k$; $b_{k+1} = b_k$

UNTIL $| a_k - b_k | < \varepsilon$

2: RETURN

the final bracket $[a_k, x_k, b_k]$ and $\alpha^* = (a_k+b_k)/2$ or x_k which is the best known interior point in the final $[a_k, x_k, b_k]$.

Quadratic Interpolation

- In the above algorithm, it is assumed that the new point y_k generated is not to close to any of the existing points a_k, x_k, b_k. In practice, however such situation could easily occur and if it does we should disregard y_k and use another way to generate a new y_k.
- The popular Brent's Method uses GSS to generate new y_k when the one generated by regular QI is too close to any one of the existing three points. Then QI is reactivated from that point on.

Modified QI (a version of Brent's)

0: Given an initial bracket $[a_1, x_1, b_1]$ and the interval of uncertainty ε. Set k =1.

1: Compute $\phi(a_k)$, $\phi(x_k)$, $\phi(b_k)$, and compute y_k using QI.
If y_k is indistinguishable from a_k, x_k, or b_k, GO TO 4. Else continue.

2: Compute $\phi(y_k)$.
- If $y_k > x_k$ and
  - If $\phi(y_k) \le \phi(x_k)$, then set $a_{k+1} = x_k$; $x_{k+1} = y_k$; $b_{k+1} = b_k$
  - Else set $a_{k+1} = a_k$; $x_{k+1} = x_k$; $b_{k+1} = y_k$
- If $y_k < x_k$ and
  - If $\phi(y_k) \le \phi(x_k)$, then set $a_{k+1} = a_k$; $x_{k+1} = y_k$; $b_{k+1} = x_k$
  - Else set $a_{k+1} = y_k$; $x_{k+1} = x_k$; $b_{k+1} = b_k$

3: If $|a_k - b_k| < \varepsilon$, STOP and RETURN the final bracket $[a_k, x_k, b_k]$ and $\alpha^* = (a_k+b_k)/2$ or x_k which is the best known interior point in the final $[a_k, x_k, b_k]$.
OTHERWISE, set k = k+1 and GO TO 1.

4: If x_k is distinguishable from the midpoint of $[a_k, b_k]$, then set $y_k = (a_k+b_k)/2$, set k = k+1 and GO TO 2.
Else, set $y_k = (a_k+x_k)/2$ if $\phi(a_k) \le \phi(b_k)$ or $(x_k+b_k)/2$ if $\phi(a_k) > \phi(b_k)$. Set k = k+1 and GO TO 2.

NOTE: Can also use GSS to find new x_k and y_k in 4, or any suitable method to find new y_k.

Interval Reduction
(when $\phi'(\alpha)$ is available)

- Given the current estimation point $\alpha^{(k)}$ and the value $\phi(\alpha^{(k)})$
- Suppose we also know value of $\phi'(\alpha^{(k)})$

$\Rightarrow$ The most efficient interval reduction method is the **BISECTION Method**

Bisection Method
(when $\phi'(\alpha)$ is available)

0: Given an initial bracket $[a_1,b_1]$ and the interval of uncertainty ε
(Since this is a bracket $\phi'(a_k) < 0$ and $\phi'(b_k) > 0$.)

1: For k = 1,2,3,...
DO
Compute midpoint $x_k = (a_k+b_k)/2$ and $\phi'(x_k)$
If $\phi'(x_k) < 0$, set $a_{k+1} = a_k$; $b_{k+1} = x_k$
If $\phi'(x_k) > 0$, set $a_{k+1} = x_k$; $b_{k+1} = b_k$
UNTIL
$|\phi'(x_k)| < \varepsilon_1$ or $|a_k - b_k| < \varepsilon_2$

2: RETURN
the final bracket $[a_k,b_k]$ and $\alpha^* = (a_k+b_k)/2$ or x_k

NOTE: Rate of reduction = 50% (best of all) compared with 32% of GSS

Approximation Methods

NEWTON'S METHOD

- Given the current estimation point $\alpha^{(k)}$ and the value $\phi(\alpha^{(k)})$
- Suppose we also know values of some derivatives e.g. $\phi'(\alpha^{(k)})$ and $\phi''(\alpha^{(k)})$
- We could approximate $\phi(\alpha)$ by a quadratic function

  $q(\alpha) = \phi(\alpha^{(k)}) + \phi'(\alpha^{(k)})(\alpha - \alpha^{(k)}) + \phi''(\alpha^{(k)})(\alpha - \alpha^{(k)})^2$

  Note that: $q(\alpha^{(k)}) = \phi(\alpha^{(k)})$

  $q'(\alpha^{(k)}) = \phi'(\alpha^{(k)})$

  $q''(\alpha^{(k)}) = \phi''(\alpha^{(k)})$

NEWTON'S METHOD

- If $\phi''(\alpha^{(k)}) > 0 \Rightarrow q''(\alpha^{(k)}) > 0 \Rightarrow q(\alpha)$ has a minimum.
- We can use the minimizer of $q(\alpha)$ as the next approximation of α^*
- The minimizer of $q(\alpha)$ satisfies $q'(\alpha) = 0$ or

  $$\phi'(\alpha^{(k)}) + \phi''(\alpha^{(k)})(\alpha - \alpha^{(k)}) = 0$$

- If $\alpha^{(k+1)}$ is a minimizer of $q(\alpha)$, then

  $\phi'(\alpha^{(k)}) + \phi''(\alpha^{(k)})(\alpha^{(k+1)} - \alpha^{(k)}) = 0$

  or $\alpha^{(k+1)} = \alpha^{(k)} - \phi'(\alpha^{(k)}) / \phi''(\alpha^{(k)})$

  which is used to produce successive approximations of α^*

NEWTON'S METHOD

Notes:

- When Newton's method works, it is the best since it converges to α^* very quickly
- It requires $\phi''(\alpha^{(k)})$ which is often expensive or impractical to get which limts the use of this method
- It requires $\phi''(\alpha^{(k)}) > 0$ to converge so it requires a very good starting point $\alpha^{(k)}$

Method of False Position

- To overcome the need to compute $\phi''(\alpha^{(k)})$ but still use 2nd order info to have good convergence, approximate $\phi''(\alpha^{(k)})$ by

$$\phi''(\alpha^{(k)}) \approx (\phi'(\alpha^{(k)}) - \phi'(\alpha^{(k-1)}))/(\alpha^{(k)} - \alpha^{(k-1)})$$

$$\Rightarrow \quad \alpha^{(k+1)} = \alpha^{(k)} - \phi'(\alpha^{(k)})(\alpha^{(k)} - \alpha^{(k-1)})/(\phi'(\alpha^{(k)}) - \phi'(\alpha^{(k-1)}))$$

Cubic Interpolation

- Given $\alpha^{(k-1)}, \phi(\alpha^{(k-1)}), \phi'(\alpha^{(k-1)})$ and $\alpha^{(k)}, \phi(\alpha^{(k)}), \phi'(\alpha^{(k)})$ are available, such that

  $\phi'(\alpha^{(k-1)}) < 0$ and $\phi'(\alpha^{(k)}) > 0$

  or $\phi'(\alpha^{(k-1)}) < 0$ and $\phi(\alpha^{(k-1)}) < \phi(\alpha^{(k)})$
- We can fit a cubic function $c(\alpha)$ though $\alpha^{(k-1)}$ and $\alpha^{(k)}$ having the same function values and derivatives at $\alpha^{(k-1)}$ and $\alpha^{(k)}$ as those of $\phi(\alpha)$
- The minimizer of $c(\alpha)$ can then be used as the next approximation of α^*

Cubic Interpolation

Here are the necessary formulas for Cubic Interpolation Method:

$$\alpha^{(k+1)} = \alpha^{(k)} - (\alpha^{(k)} - \alpha^{(k-1)})[\phi'(\alpha^{(k)}) + u_2 - u_1]/[\phi'(\alpha^{(k)}) - \phi'(\alpha^{(k-1)}) + 2u_2]$$

where

$$u_1 = \phi'(\alpha^{(k)}) + \phi'(\alpha^{(k-1)}) - 3[\phi(\alpha^{(k)}) - \phi(\alpha^{(k-1)})]/[\alpha^{(k)} - \alpha^{(k-1)}]$$

and

$$u_2 = [u_1^2 - \phi'(\alpha^{(k-1)})\, \phi'(\alpha^{(k)})]^{1/2}$$

- Quite powerful and not too difficult to implement on computers if the assumptions are OK