Numerical Mathematics & Computing, 7 Ed.
§13.1 Minimization of Functions

Ward Cheney & David Kincaid

Cengage§c

http://www.cengagebrian.com
http://www.ma.utexas.edu/CNA/NMC7

Minimization of Functions

Applications

- Economics and finance: expenditure minimization problem
- Mechanics
- Electrical engineering
- Civil engineering
- Operations research
- Control engineering
- Geophysics
- Molecular modeling: protein structure prediction

An Example

- An **engineering design problem** leads to a function

$$F(x, y) = \cos(x^2) + e^{(y-6)^2} + 3(x + y)^4$$

  in which x and y are parameters to be selected and $F(x, y)$ is a function related to the cost of manufacturing and is to be **minimized**.

- Methods for **locating optimal points** (x, y) in such problems are developed now.

Finding the minima of a function in Calculus

- Problems of **maximization** are covered by the theory of minimization because the maxima of F occur at points where $-F$ has its minima.
- An important application of calculus is the problem of **finding the local minima of a function**.
- In **calculus**, the principal technique for minimization is to differentiate the function whose minimum is sought, set the derivative equal to zero, and locate the points that satisfy the resulting equation.

- This technique can be used on **functions of one or several variables**.
- For example, if a minimum value of $F(x_1, x_2, x_3)$ is sought, we look for the points where **all three partial derivatives are simultaneously zero**:

$$\frac{\partial F x_1}{\partial v} = \frac{\partial F x_2}{\partial v} = \frac{\partial F x_1}{\partial v} = 0$$

- This procedure cannot be readily accepted as a **general-purpose** numerical method because it requires differentiation followed by the solution of one or more equations in one or more variables.
- This task may be as **difficult to carry out** as a direct frontal attack on the original problem.

Unconstrained and Constrained Minimization Problems

- The **minimization problem** has two forms:
  unconstrained
  constrained.
- In an **unconstrained** minimization problem, a function F is defined from the n-dimensional space $\mathbb{R}^n$ into the real line $\mathbb{R}$, and a point $\mathbf{z} \in \mathbb{R}^n$ is sought with the property that

$$F(\mathbf{z}) \leq F(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathbb{R}^n$$

- It is convenient to write points in $\mathbb{R}^n$ simply as $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, and so on.
- If it is necessary to display the components of a point, we write

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$$

Constrained Minimization Problems

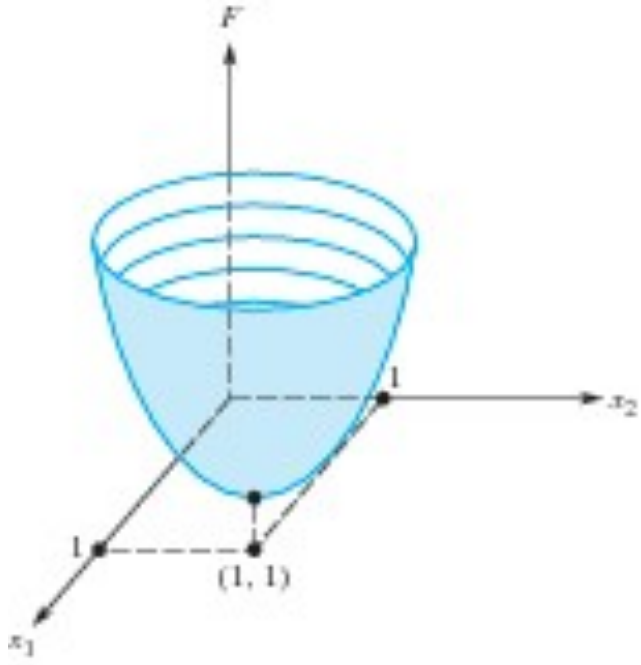
- In a **constrained** minimization problem, a subset $\mathbf{K}$ in $\mathbb{R}^n$ is prescribed, and a point $\mathbf{z} \in \mathbf{K}$ is sought for which

$$F(\mathbf{z}) \leq F(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathbf{K}$$

- Such problems are **more difficult** because of the need to keep the points within the set $\mathbf{K}$.
- Sometimes the set $\mathbf{K}$ is defined in a complicated way.

- Consider the **elliptic paraboloid**

$$F(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 4$$

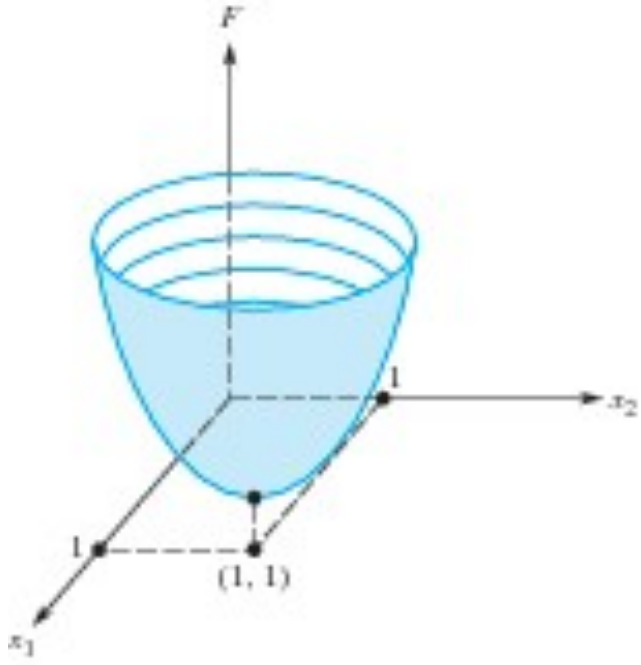


Elliptic paraboloid

- The unconstrained minimum?
- If $\mathbf{K} = \{(x_1, x_2): x_1 \leq 0, x_2 \leq 0\}$ the constrained minimum?

- Consider the **elliptic paraboloid**

$$F(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 4$$



Elliptic paraboloid

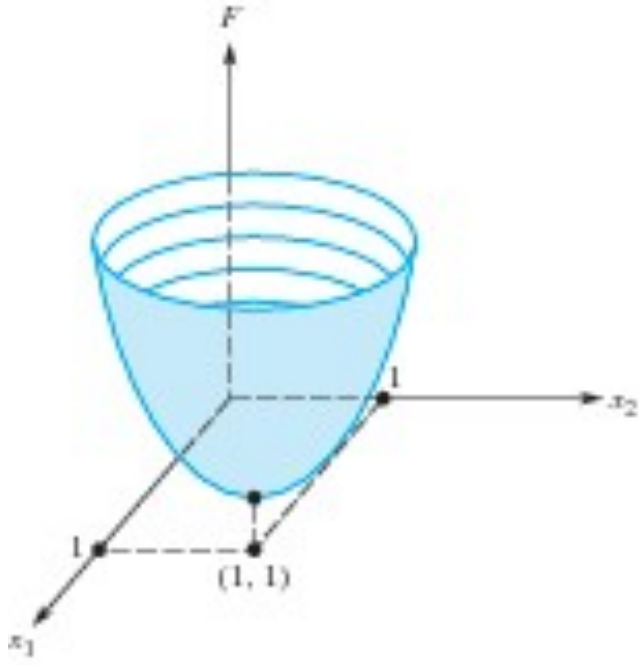
- The unconstrained minimum occurs at (1, 1) because

$$F(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2 + 2$$

  - If $\mathbf{K} = \{(x_1, x_2): x_1 \leq 0, x_2 \leq 0\}$, the constrained minimum?

- Consider the **elliptic paraboloid**

$$F(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 4$$



Elliptic paraboloid

- The unconstrained minimum occurs at (1, 1) because

$$F(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2 + 2$$

- If $\mathbf{K} = \{(x_1, x_2): x_1 \leq 0, x_2 \leq 0\}$ the constrained minimum is 4 at (0,0).

One-Variable Case

- The **special case** in which a **function F is defined on** $\mathbb{R}$ is considered first because the more general problem with n variables is often solved by a sequence of one-variable problems.
- Suppose that $F: \mathbb{R} \to \mathbb{R}$ and that we seek a point $z \in \mathbb{R}$ with the property that $F(z) \leq F(x)$ for all $x \in \mathbb{R}$.
- Note that if **no assumptions** are made about F, this problem is **insoluble** in its general form!

- For instance, the function

$$f(x) = \frac{1}{1+x^2}$$

has **no minimum point**.

- Even for relatively well-behaved functions, such as

$$F(x) = x^2 + \sin(53x)$$

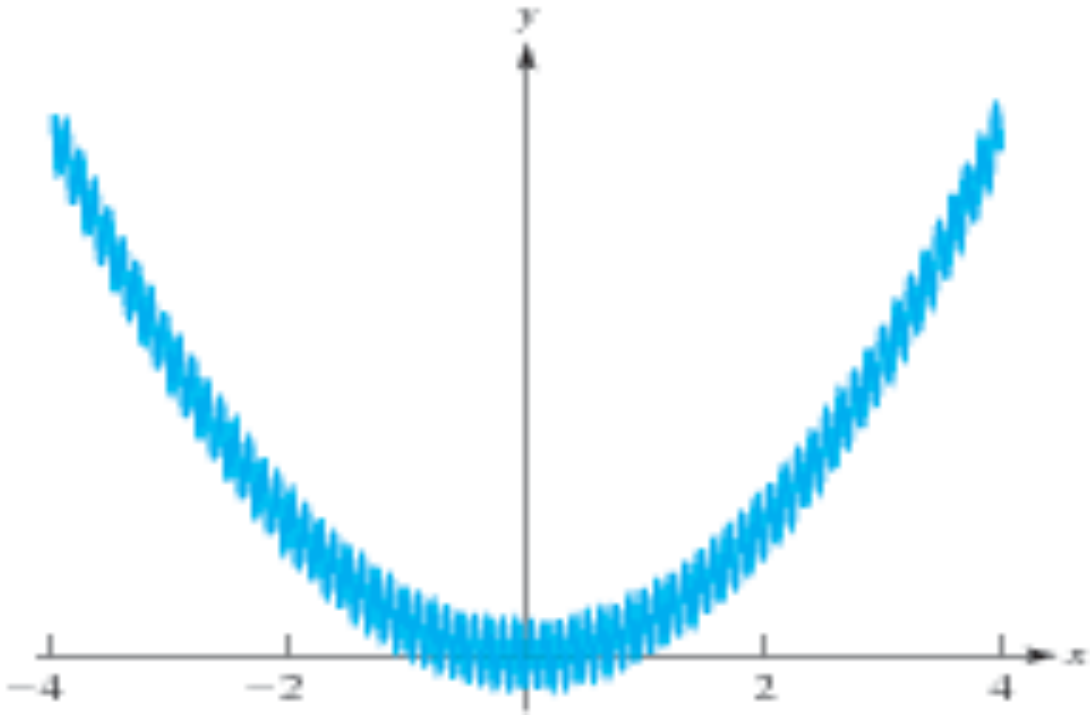
numerical methods may encounter some difficulties

because of the **large number of purely local minima**.

- Recall that a point z is a **local minimum** point of a function F if there is some neighborhood of z in which all points satisfy

$$F(z) \leq F(x)$$

$$F(x) = x^2 + \sin(53x)$$



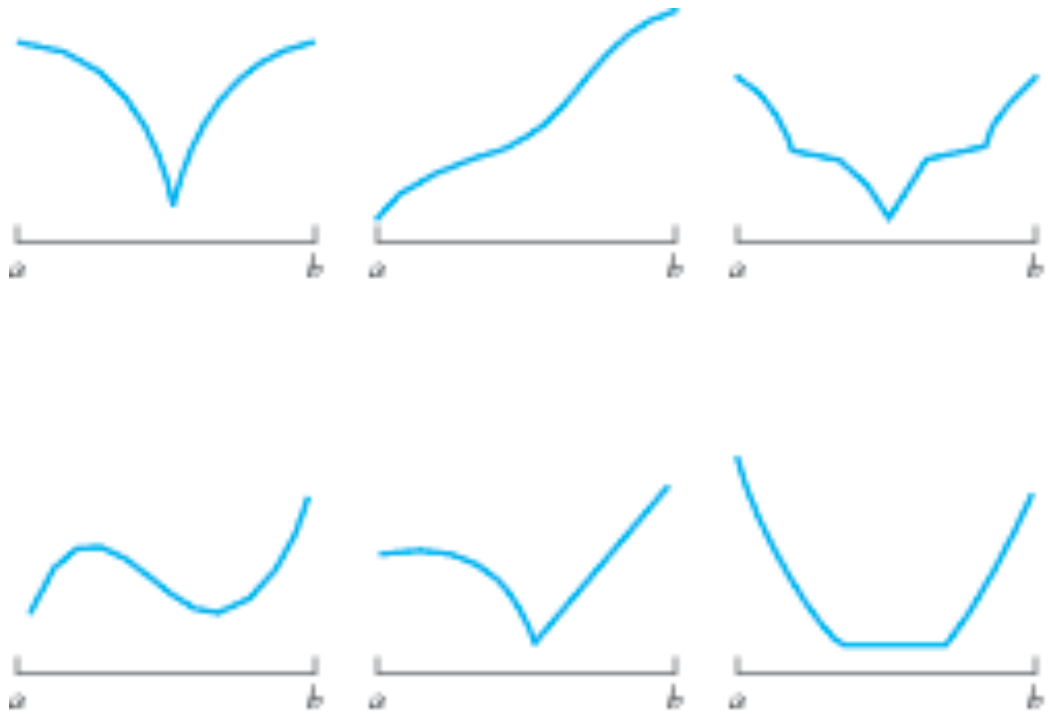
We can use mathematical software to find local minimum values for the function

$$F(x) = x^2 + \sin(53x)$$

- First, we define the function, find a local minimum value in the interval $\left[-\frac{1}{2}, \frac{1}{2}\right]$, and plot the curve.
- The point that is computed may not be a global minimum point!
- To **try to find the global minimum point**, we could use various starting values to find **local minimum values** and then find the **minimum of them**.
- In fact, we find a **local minimum** -0.999122 at $t = -0.0296166$, which is the **global minimum** for this function.

Unimodal Functions F

- In attacking a minimization problem, one **reasonable assumption** is that on some interval $[a, b]$ given to us in advance, F has **only a single local minimum**.
- This property is often expressed by saying that F is **unimodal** on $[a,b]$.
- **Caution**: In statistics, **unimodal** refers to a single local maximum.
- Some unimodal functions are sketched in the following figures.



Examples of unimodal and nonunimodal functions

Yes or No question

A continuous unimodal function strictly decrease up to the minimum point and strictly increase thereafter.

- An **important property of a continuous unimodal function**, which might be surmised from these figures is that it is **strictly decreasing up to the minimum point and strictly increasing thereafter**.
- To be convinced of this, let x^* **be the minimum point of** F **on** $[a, b]$ and suppose, for instance, that F is not strictly decreasing on the interval $[a, x^*]$.
- Then points x_1 and x_2 that satisfy $a \leq x_1 < x_2 \leq x^*$ and $F(x_1) \leq F(x_2)$ must exist.
- Now let x^{**} **be a minimum point of** F on the interval $[a, x_2]$.
- Recall that a continuous function on a closed finite interval attains its minimum value.

- We can assume that $x^{**} \neq x_2$ because if x^{**} were initially chosen as x_2, it could be replaced by x_1 inasmuch as $F(x_1) \leq F(x_2)$.
- But now we see that x^{**} **is a local minimum point of F in the interval** $[a, b]$ because it is a minimum point of F on $[a, x_2]$, but it is not x_2 itself.
- The presence of two local minimum points, of course, contradicts the unimodality of F.

Fibonacci Search Algorithm

- Now we pose a problem concerning the **search for a minimum point x^* of a continuous unimodal function F on a given interval $[a, b]$**.

Fibonacci Search Algorithm

- Now we pose a problem concerning the **search for a minimum point x^* of a continuous unimodal function F on a given interval $[a, b]$**.
- **How accurately can the true minimum pointx^* be computed with only n evaluations of F?**

Fibonacci Search Algorithm

- Now we pose a problem concerning the **search for a minimum point x^* of a continuous unimodal function F on a given interval $[a, b]$**.
- **How accurately can the true minimum pointx^* be computed with only n evaluations of F?**
- With no evaluations of F, the best that can be said is that $x^* \in [a, b]$; taking the **midpoint**

$$\widehat{x} = \tfrac{1}{2}(b + a)$$

Fibonacci Search Algorithm

- Now we pose a problem concerning the **search for a minimum point x^* of a continuous unimodal function F on a given interval $[a, b]$**.
- **How accurately can the true minimum pointx^* be computed with only n evaluations of F?**
- With no evaluations of F, the best that can be said is that $x^* \in [a, b]$; taking the **midpoint**

$$\widehat{x} = \tfrac{1}{2}(b + a)$$

as the best estimate gives an **error** of

$$|x^* - \widehat{x}| \leq \tfrac{1}{2}(b - a)$$

Fibonacci Search Algorithm

- Now we pose a problem concerning the **search for a minimum point x^* of a continuous unimodal function F on a given interval $[a, b]$.**
- **How accurately can the true minimum pointx^* be computed with only n evaluations of F?**
- With no evaluations of F, the best that can be said is that $x^* \in [a, b]$; taking the **midpoint**

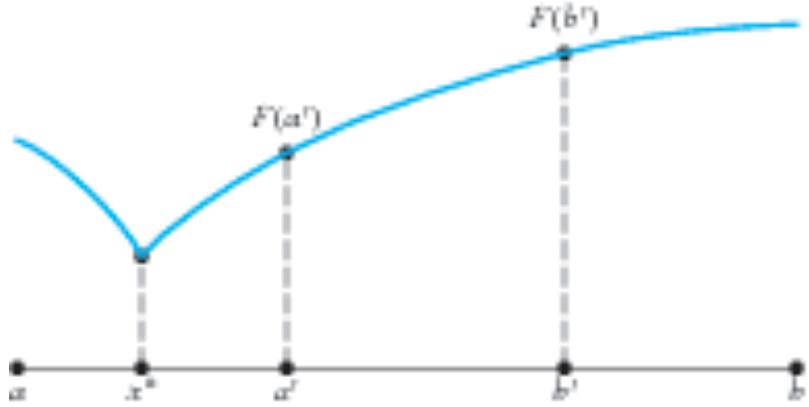
$$\widehat{x} = \tfrac{1}{2}(b + a)$$

as the best estimate gives an **error** of

$$|x^* - \widehat{x}| \leq \tfrac{1}{2}(b - a)$$

- One evaluation by itself does not improve this situation, so the best estimate and the error remain the same as in the previous case.
- Consequently, we need **at least two function evaluations** to obtain a better estimate.

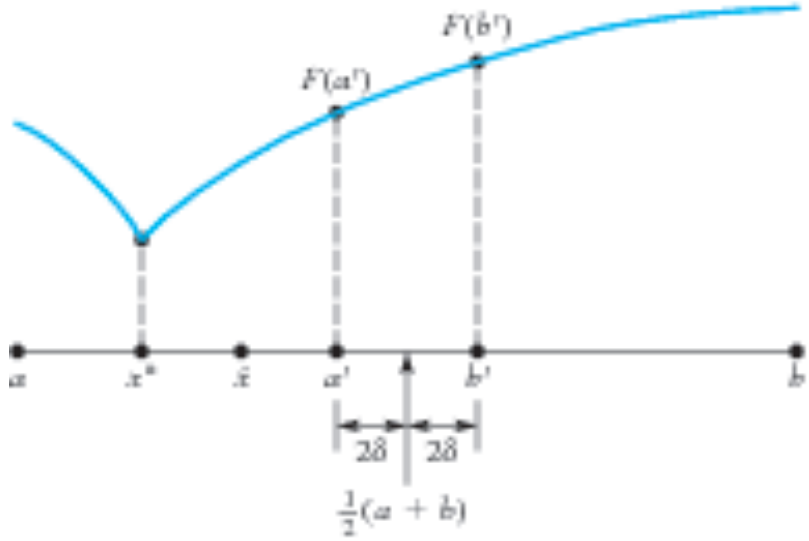
- Suppose that F is evaluated at a' and b' with the results shown.



Fibonacci search algorithm: F evaluated at a' and b'

- If
$$F(a') < F(b')$$
then because F is increasing to the right of x^*, we can be sure that $x^* \in [a, b']$.
- On the other hand, similar reasoning for the case
$$F(a') \geq F(b')$$
shows that $x^* \in [a', b]$.

- Thus, F should be evaluated at two nearby points on either side of the midpoint.



Fibonacci search algorithm:F evaluated on either side of the midpoint

- Suppose that

$$a' = \frac{1}{2}(a+b) - 2\delta, \qquad b' = \frac{1}{2}(a+b) + 2\delta$$

- Taking the **midpoint** of the appropriate subinterval $[a, b']$ or $[a', b]$ as the best estimate $\widehat{x}$ of x^*, we find that the **error does not exceed**

$$\frac{1}{4}(b-a) + \delta$$

- For $n = 3$, **two evaluations** are first made at the $\frac{1}{3}$ and $\frac{2}{3}$ points of the initial interval $[a, b]$; that is,

$$a' = a + \frac{1}{3}(b - a), \qquad b' = a + \frac{2}{3}(b - a)$$

- From the two values $F(a')$ and $F(b')$, it can be determined whether $x^* \in [a, b']$ or $x^* \in [a', b]$.
- The two cases are, of course, similar.

- Let us suppose that $F(a') \geq F(b')$, so that our minimum point x^* must be in $[a', b]$, as shown.

- The third (final) evaluation is made close to b', for example, at $b' + \delta$ (where $\delta > 0$).
- If $F(b') \geq F(b' + \delta)$, then $x^* \in [b', b]$.
- Taking the midpoint of this interval, we obtain $\widehat{x} = \frac{1}{2}(b' + b)$ as our estimate of x^* and find that

$$|\widehat{x} - x^*| \leq \frac{1}{6}(b - a)$$

- On the other hand, if $F(b') < F(b' + \delta)$, then $x^* \in [a', b' + \delta]$.
- Again we take the **midpoint**

$$\widehat{x} = \frac{1}{2}(a' + b' + \delta)$$

and find that

$$|\widehat{x} - x^*| \leq \tfrac{1}{6}(b - a) + \frac{1}{2}\delta$$

- So if we ignore the small quantity $\delta/2$, our accuracy is

$$\frac{1}{6}(b - a)$$

in using three evaluations of F.

- By continuing the search pattern outlined, we find an estimate $\widehat{x}$ of x^* with only n evaluations of F and with an **error not exceeding**

$$\frac{1}{2}\left(\frac{b-a}{\lambda_n}\right) \tag{1}$$

where λ_n is the $(n+1)$st member of the **Fibonacci sequence**:

$$\begin{cases} \lambda_1 = 1, \quad \lambda_2 = 1 \\ \lambda_k = \lambda_{k-1} + \lambda_{k-2} \qquad (k \geq 3) \end{cases} \tag{2}$$

- For example, elements λ_1 through λ_8 are 1, 1, 2, 3, 5, 8, 13, and 21.

- In the **Fibonacci search algorithm**, we initially determine the number of steps N for a desired accuracy $\epsilon > \delta$ by selecting N to be the subscript of the smallest Fibonacci number greater than $\frac{1}{2}(b-a)/\epsilon$.
- We define a **sequence of intervals**, starting with the given interval $[a, b]$ of length $\ell = b - a$, and, for $k = N,\ N-1, \ldots, 3$, use these **formulas for updating**:

$$\Delta = \left(\frac{\lambda_{k-2}}{\lambda_k}\right)(b-a)$$

$$a' = a + \Delta \qquad b' = b - \Delta \tag{3}$$

$$\begin{cases} a = a', & \text{if } F(a') \geq F(b') \\ b = b', & \text{if } F(a') < F(b') \end{cases}$$

- At the **step** $k = 2$, we set

$$a' = \frac{1}{2}(a+b) - 2\delta, \qquad b' = \frac{1}{2}(a+b) + 2\delta$$

$$\begin{cases} a = a', & \text{if } F(a') \geq F(b') \\ b = b', & \text{if } F(a') < F(b') \end{cases}$$

and we have the final interval $[a, b]$, from which we compute

$$\widehat{x} = \frac{1}{2}(a+b)$$

- This algorithm requires **only one function evaluation per step** after the initial step.

- To verify the algorithm, consider the situation shown in this figure.

Fibonacci search algorithm: Verify using a typical situation

- Since $\lambda_k = \lambda_{k-1} + \lambda_{k-2}$, we have

$$\ell' = \ell - \Delta = \ell - \left(\frac{\lambda_{k-2}}{\lambda_k}\right)\ell = \left(\frac{\lambda_{k-1}}{\lambda_k}\right)\ell \tag{4}$$

- The **length of the interval of uncertainty** has been **reduced by the factor** $(\lambda_{k-1}/\lambda_k)$.
- The next step yields

$$\Delta' = \left(\frac{\lambda_{k-3}}{\lambda_{k-1}}\right)\ell' \tag{5}$$

and Δ' is actually the distance between a' and b'.

- Therefore, one of the preceding points at which the function was evaluated is at one end or the other of $[a, b]$; that is,

$$
\begin{aligned}
b' - a' = \ell = 2\Delta &= \left(\frac{\lambda_k - 2\lambda_{k-2}}{\lambda_k}\right)\ell \\
&= \left(\frac{\lambda_{k-1} - \lambda_{k-2}}{\lambda_k}\right)\ell = \left(\frac{\lambda_{k-3}}{\lambda_k}\right)\ell \\
&= \left(\frac{\lambda_{k-3}}{\lambda_{k-1}}\right)\ell' = \Delta'
\end{aligned}
$$

  by (2), (4), and (5).

- It is clear by (4) that after $N-1$ function evaluations, the next-to-last interval has length $(1/\lambda_N)$ times the length of the initial interval $[a, b]$.
- So the final interval is $(b-a)(1/\lambda_N)$ wide, and the maximum error (1) is established.
- The final step is similar to that outlined, and F is evaluated at a point 2δ away from the midpoint of the next-to-last interval.
- Finally, set

$$\widehat{x} = \frac{1}{2}(b+a)$$

from the last interval $[a, b]$.

- One disadvantage of the Fibonacci search is that the algorithm is rather complicated.
- Also, the desired precision must be given in advance, and the number of steps to be computed for this precision must be determined before beginning the computation.
- Thus, the initial evaluation points for the function F depend on N, the number of steps.

Golden Section Search Algorithm

- A similar algorithm that is free of these drawbacks is described next.
- It has been termed the **Golden Section Search** because it depends on a **ratio** ρ known to the early Greeks as the **golden section ratio**:

$$\rho = \frac{1}{2}(1 + \sqrt{5}) \approx 1.61803\ 39887$$

- ρ satisfies the equation

$$\rho^2 = \rho + 1$$

which has roots

$$\frac{1}{2}(1 + \sqrt{5}) \approx 1.61803\ldots$$

$$\frac{1}{2}(1 - \sqrt{5}) \approx -0.61803\ldots.$$

- In each step of this iterative algorithm, an interval $[a, b]$ is available from the previous work.

- It is an **interval that is known to contain the minimum point** x^*, and our objective is to **replace it by a smaller interval** that is also known to contain x^*.
- In each step, two values of F are needed:

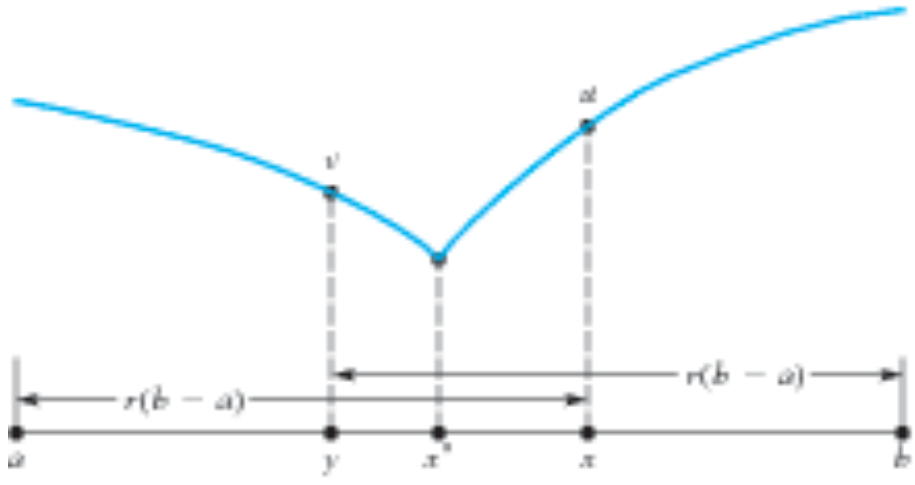
$$\begin{cases} x = a + r(b-a), & u = F(x) \\ y = a + r^2(b-a), & v = F(y) \end{cases} \tag{6}$$

where $r = 1/\rho$ and $r^2 + r = 1$, which has roots

$$\frac{1}{2}\left(-1 + \sqrt{5}\right) \approx 0.61803\ldots$$

$$\frac{1}{2}\left(-1 - \sqrt{5}\right) \approx -1.61803\ldots.$$

- There are two cases to consider: **either** $u > v$ **or** $u \leq v$
- Let us take the first.
- The next figure depicts this situation.
- Since F is assumed continuous and unimodal, the minimum of F must be in the interval $[a, x]$.
- This interval is the input interval at the beginning of the next step.



Golden section search algorithm: $u > v$

- There are two cases to consider: **either** $u > v$ **or** $u \leq v$
- Let us take the first.
- The next figure depicts this situation.
- Since F is assumed continuous and unimodal, the minimum of F must be in the interval $[a, x]$.
- This interval is the input interval at the beginning of the next step.
- Observe now that within the interval $[a, x]$, one evaluation of F is already available, namely, at y.
- Also note that

$$a + r(x - a) = y$$

  because $x - a = r(b - a)$.
- In the next step, therefore, y will play the role of x, and we shall need the value of F at the point

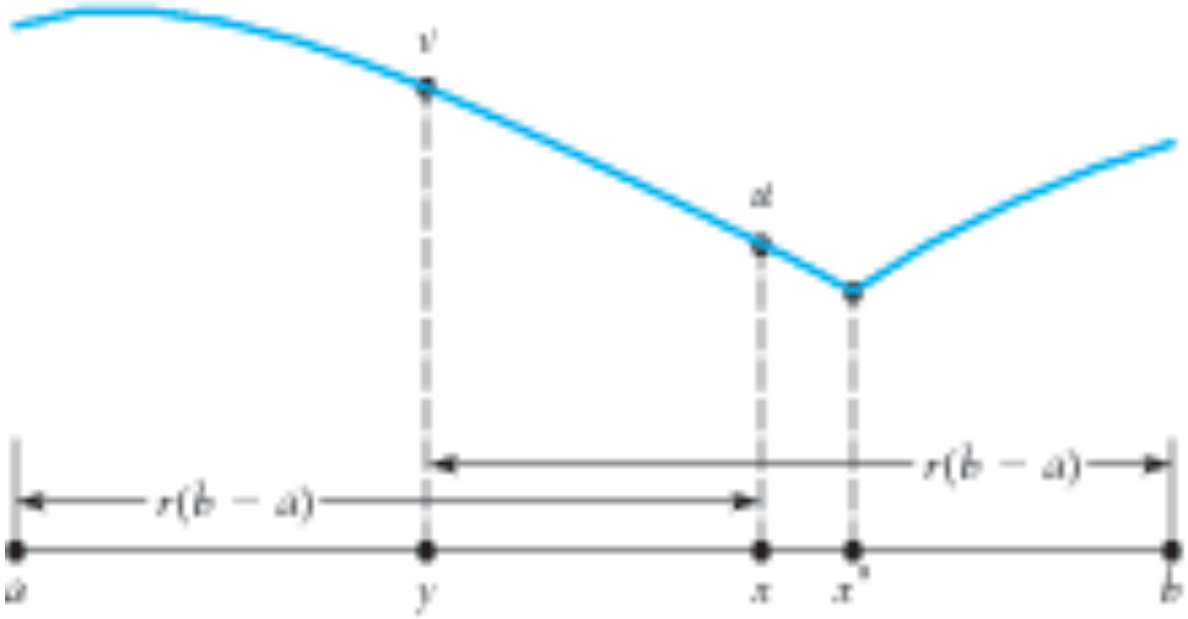
$$a + r^2(x - a)$$

- So what must be done in this step is to carry out the following replacements **in order**:

$$
\begin{aligned}
b &\leftarrow x \\
x &\leftarrow y \\
u &\leftarrow v \\
y &\leftarrow a + r^2(b-a) \\
v &\leftarrow F(y)
\end{aligned}
$$

- The other case is similar.
- If $u \leq v$, the picture might be as in the next figure.
- In this case, the minimum point must lie in $[y, b]$.
- Within this interval, one value of F is available, namely, at x
- Observe that

$$y + r^2(b - y) = x$$



Golden section search algorithm: $u \leq v$

- Thus, x should now be given the role of y, and the value of F is to be computed at $y + r(b - y)$.
- The following ordered replacements accomplish this:

$$
\begin{aligned}
a &\leftarrow y \\
y &\leftarrow x \\
v &\leftarrow u \\
x &\leftarrow a + r(b - a) \\
u &\leftarrow F(x)
\end{aligned}
$$

Analysis

- If $[a, b]$ is the starting interval in the search for a minimum of F, then at the beginning, with one evaluation of F, we can be sure only that the minimum point, x^*, is in an **interval** of width $b - a$.
- In the **golden section search**, the corresponding **lengths in successive steps** are $r(b - a)$ for **two evaluations** of F, $r^2(b - a)$ for three evaluations of F, and so on.
- After n **steps**, the minimum point has been pinned down to an **interval of length** $r^{n-1}(b - a)$.

- **How does this compare with the Fibonacci search algorithm using n evaluations?**
- The corresponding width of interval, at the last step of this algorithm, is $\lambda_n^{-1}(b-a)$.
- Now, the Fibonacci algorithm should be better, because it is designed to do as well as possible with a prescribed number of steps.
- So we expect the ratio r^{n-1}/λ_n^{-1} to be greater than 1.
- But it approaches 1.17 as $n \to \infty$.
- Thus, one may conclude that the extra complexity of the Fibonacci algorithm, together with the disadvantage of having the algorithm itself depend on the number of evaluations permitted, mitigates against its use in general.

Summary

- Performance is independent of the function that is being minimized
- No advantage has been taken of any smoothness that the function F may possess.
- This **procedure is quite slow**. Slowness in this context refers to the **large number of function evaluations** that are needed to achieve reasonable precision.
- It can be surmised that this slowness is attributable to the **extreme generality of the algorithm**.

Quadratic Interpolation Algorithm

- We do not know x^* and do not want to involve derivatives in our algorithms, a natural stratagem is to interpolate F by a quadratic polynomial.
- Any three values $(x_i, F(x_i))$, $i = 1, 2, 3$, can be used for this purpose.
- The minimum point of the resulting quadratic function may be a better approximation to x^* than is x_1, x_2, or x_3.
- Writing an algorithm that carries out this idea iteratively is **not trivial**, and **many unpleasant cases** must be handled.

Outline of an algorithm

- Computing begins by evaluating the two numbers

$$\begin{cases} u = F(x) \\ v = F(y) \end{cases}$$

- Now let

$$z = \begin{cases} 2x - y, & \text{if } u < v \\ 2y - x, & \text{if } u \geq v \end{cases}$$

- In either case, the number

$$w = F(z)$$

is to be computed.

- In the main calculation, a quadratic polynomial q is determined to interpolate F at the three current points x, y, and z.
- The formulas are discussed below.
- Next, the point t where $q'(t) = 0$ is determined.
- Under ideal circumstances, t is a **minimum point** of q and an **approximate minimum point** of F.
- So one of the x, y, or z should be replaced by t.
- We are interested in examining $q''(t)$ to determine the shape of the curve q near t.

Process

- At this stage, we have three points x, y, and z together with corresponding function values u, v, and w.
- In the main iteration step of the algorithm, one of these points and its accompanying function value are replaced by a new point and new function value.
- The process is repeated until a success or failure is reached.

- For the complete description of this algorithm, the formulas for t and $q''(t)$ must be given.
- They are obtained as follows:

$$\begin{cases} a = \dfrac{v-u}{y-x} \\ b = \dfrac{w-v}{z-y} \\ c = \dfrac{b-a}{z-x} \\ t = \dfrac{1}{2}\left[x+y-\dfrac{a}{c}\right] \\ q''(t) = 2c \end{cases}$$

- The **solution case** occurs if

$$q''(t) > 0, \qquad \max\{|t-x|, |t-y|, |t-z|\} < \varepsilon$$

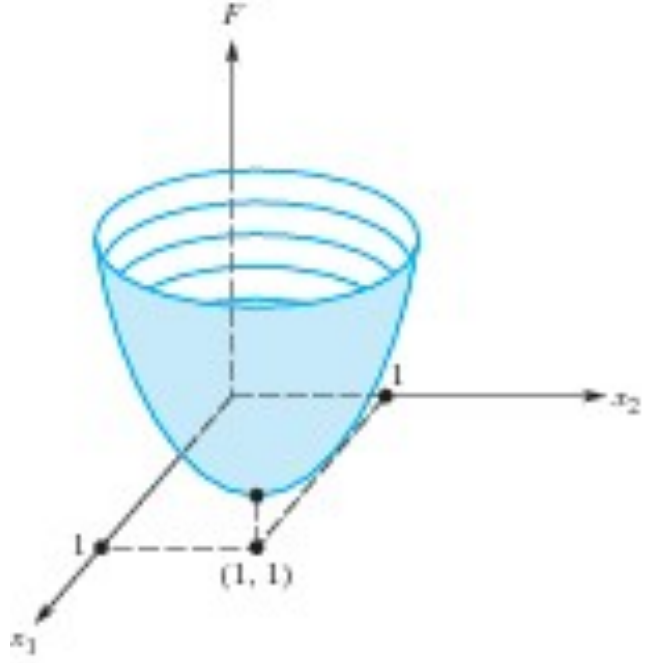
- The condition $q''(t) > 0$ indicates, of course, that q' is **increasing** in the vicinity of t, so t is indeed a minimum point of q.
- The second condition indicates that this estimate, t, of the minimum point of F is within distance ε of each of the three points x, y, and z.
- In this case, t is accepted as a solution.

Functions in Matlab

fminunc: Unconstrained Minimization
https://www.mathworks.com/help/optim/ug/fminunc.html

fmincon: Find minimum of constrained nonlinear multivariable function

$$F(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 4$$



The unconstrained minimum occurs at (1, 1)

§13.2 Minimization of Functions: Multivariate Case

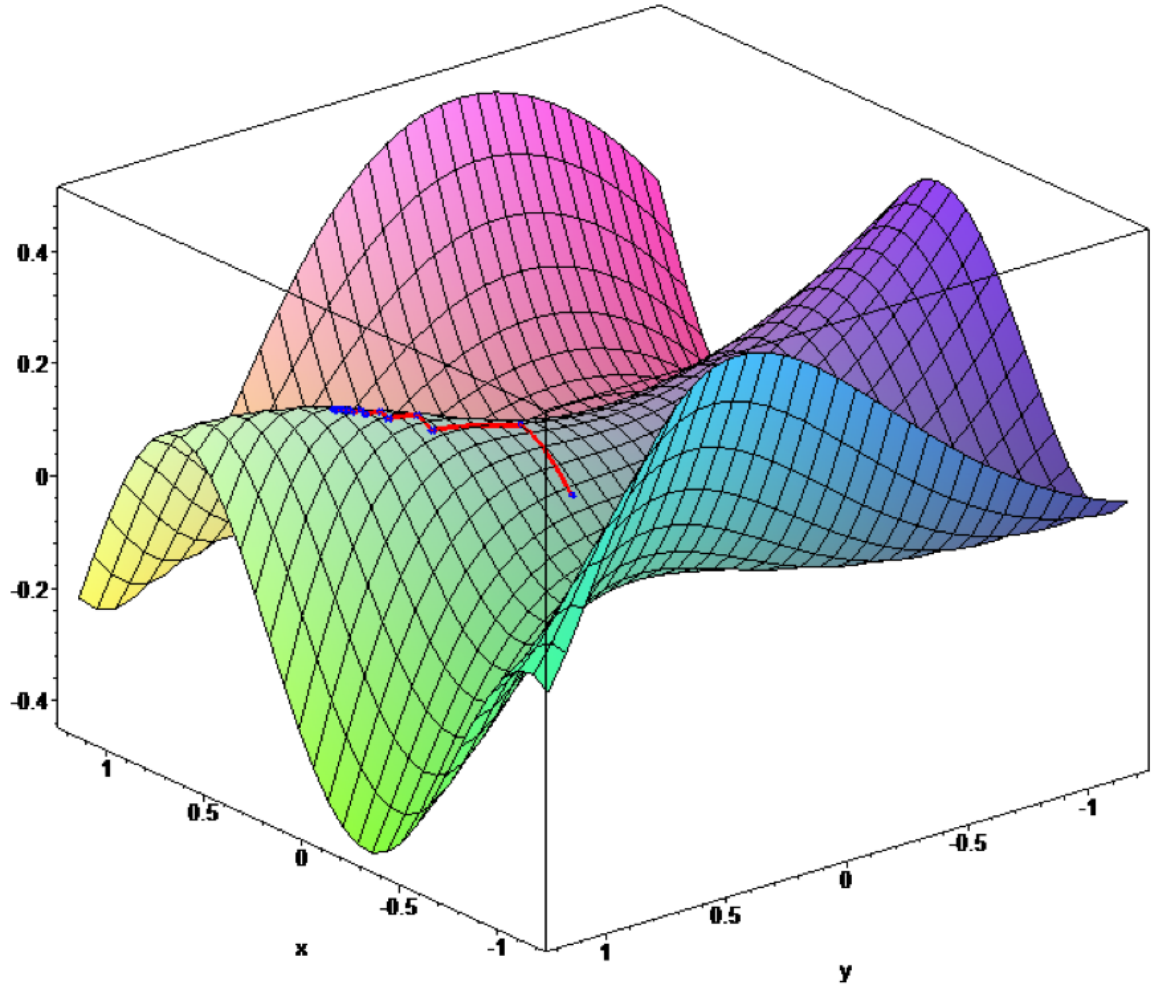
- Now we consider a real-valued function of n real variables F: $\mathbb{R}^n \to \mathbb{R}$.
- As before, a point $\mathbf{x}^*$ is sought such that

$$F(\mathbf{x}^*) \leq F(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathbb{R}^n$$

- Some of the theory of multivariate functions must be developed to understand the rather sophisticated minimization algorithms in current use.

Example

$$F(x,y) = \sin\left(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3\right)\cos(2x + 1 - e^y)$$



Taylor Series for F: Gradient Vector and Hessian Matrix

- If the function F possesses partial derivatives of certain low orders (which is usually assumed in the development of these algorithms), then at any given point $\mathbf{x}$, a **gradient vector** $\mathbf{G}(\mathbf{x}) = (G_i)_n$ is defined with components

$$G_i = G_i(\mathbf{x}) = \frac{\partial F(\mathbf{x})}{\partial x_i} \qquad (1 \leq i \leq n) \tag{1}$$

  and a **Hessian matrix** $\mathbf{H}(\mathbf{x}) = (H_{ij})_{n \times n}$ is defined with components

$$H_{ij} = H_{ij}(\mathbf{x}) = \frac{\partial^2 F(\mathbf{x})}{\partial x_i \, \partial x_j} \qquad (1 \leq i,\ j \leq n) \tag{2}$$

- We interpret $\mathbf{G}(\mathbf{x})$ as an n-component vector and $\mathbf{H}(\mathbf{x})$ as an $n \times n$ matrix, both depending on $\mathbf{x}$.

- A result in calculus states that the **order** in which partial derivatives are taken is immaterial if all partial derivatives that occur are continuous.
- In the special case of the Hessian matrix, if the second partial derivatives of F are all continuous, then $\mathbf{H}$ is a **symmetric matrix**; that is, $\mathbf{H} = \mathbf{H}^T$ because

$$H_{ij} = \frac{\partial^2 F}{\partial x_i \, \partial x_j} = \frac{\partial^2 F}{\partial x_j \, \partial x_i} = H_{ji}$$

- Using the gradient and Hessian, we can write the first few terms of the Taylor series for F as

$$F(\mathbf{x}+\mathbf{h}) = F(\mathbf{x}) + \sum_{i=1}^{n} G_i(\mathbf{x})h_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} h_i H_{ij}(\mathbf{x})h_j + \cdots \qquad (3)$$

- (3) can also be written in an elegant matrix-vector form:

$$F(\mathbf{x}+\mathbf{h}) = F(\mathbf{x}) + \mathbf{G}(\mathbf{x})^T\mathbf{h} + \frac{1}{2}\mathbf{h}^T\mathbf{H}(\mathbf{x})\mathbf{h} + \cdots \qquad (4)$$

- Here, $\mathbf{x}$ is the fixed point of expansion in $\mathbb{R}^n$, and $\mathbf{h}$ is the variable in $\mathbb{R}^n$ with components $h_1, h_2, \ldots, h_n$.
- The three dots indicate higher-order terms in $\mathbf{h}$ that are not needed in this discussion.

Example 1

To illustrate Formula (4), let us compute the first three terms in the Taylor series for the function

$$F(x_1, x_2) = \cos(\pi x_1) + \sin(\pi x_2) + e^{x_1 x_2}$$

taking $(1, 1)$ as the point of expansion.

- Partial derivatives are

$$\frac{\partial F}{\partial x_1} = -\pi \sin(\pi x_1) + x_2 e^{x_1 x_2}, \qquad \frac{\partial F}{\partial x_2} = \pi \cos(\pi x_2) + x_1 e^{x_1 x_2}$$

$$\frac{\partial^2 F}{\partial x_1^2} = -\pi^2 \cos(\pi x_1) + x_2^2 e^{x_1 x_2}, \qquad \frac{\partial^2 F}{\partial x_2 \, \partial x_1} = (x_1 x_2 + 1) e^{x_1 x_2}$$

$$\frac{\partial^2 F}{\partial x_1 \, \partial x_2} = (x_1 x_2 + 1) e^{x_1 x_2}, \qquad \frac{\partial^2 F}{\partial x_2^2} = \pi^2 \sin(\pi x_2) + x_1^2 e^{x_1 x_2}$$

- Note the equality of cross derivatives; that is,

$$\partial^2 F/\partial x_1\,\partial x_2 = \partial^2 F/\partial x_2\,\partial x_1$$

- At the particular point $\mathbf{x} = [1, 1]^T$, we have

$$F(\mathbf{x}) = -1+e, \qquad \mathbf{G}(\mathbf{x}) = \begin{bmatrix} e \\ -\pi + e \end{bmatrix}, \qquad \mathbf{H}(\mathbf{x}) = \begin{bmatrix} \pi^2 + e & 2e \\ 2e & e \end{bmatrix}$$

- So by (4),

$$F(1+h_1,1+h_2) = -1+e+[e,-\pi+e]\begin{bmatrix} h_1 \\ h_2 \end{bmatrix} + \frac{1}{2}[h_1,h_2]\begin{bmatrix} \pi^2+e & 2e \\ 2e & e \end{bmatrix}\begin{bmatrix} h_1 \\ h_2 \end{bmatrix} + \cdots$$

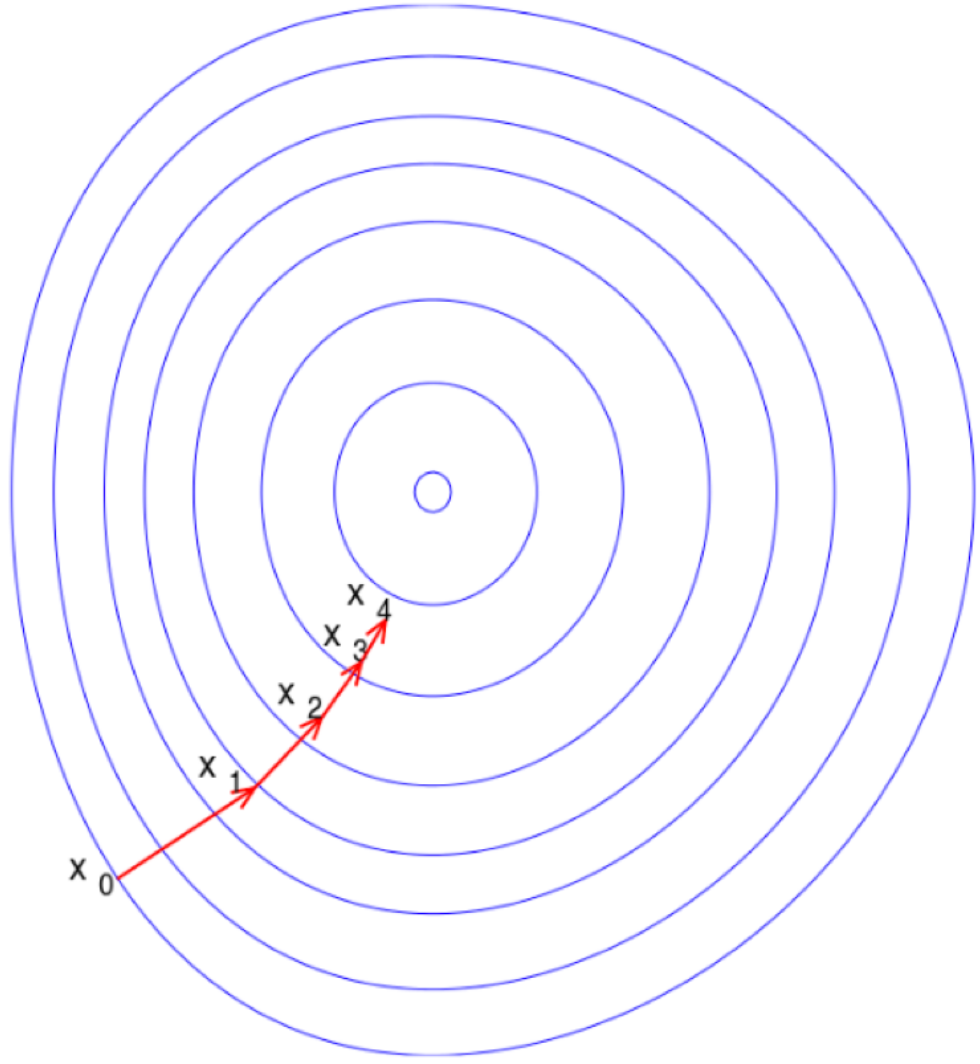
- Equivalently, by (3),

$$F(1+h_1,1+h_2) = -1+e+eh_1+(-\pi+e)h_2 + \frac{1}{2}\left[(\pi^2+e)h_1^2+(2e)h_1h_2+(2e)h_2h_1+eh_2^2\right] + \cdots$$

Steepest Descent Procedure

- Thus, $\mathbf{x}^T\mathbf{H}\mathbf{x}$ can be interpreted as the sum of all n^2 terms in a square array of which the (i, j) element is $x_i H_{ij} x_j$.
- A crucial property of the gradient vector $\mathbf{G}(\mathbf{x})$ is that it points in the direction of the most rapid increase in the function F, which is the direction of **steepest ascent**.
- Conversely, $-\mathbf{G}(\mathbf{x})$ points in the direction of the **steepest descent**.

Illustration of steepest descent (also called gradient descent on a series of level sets



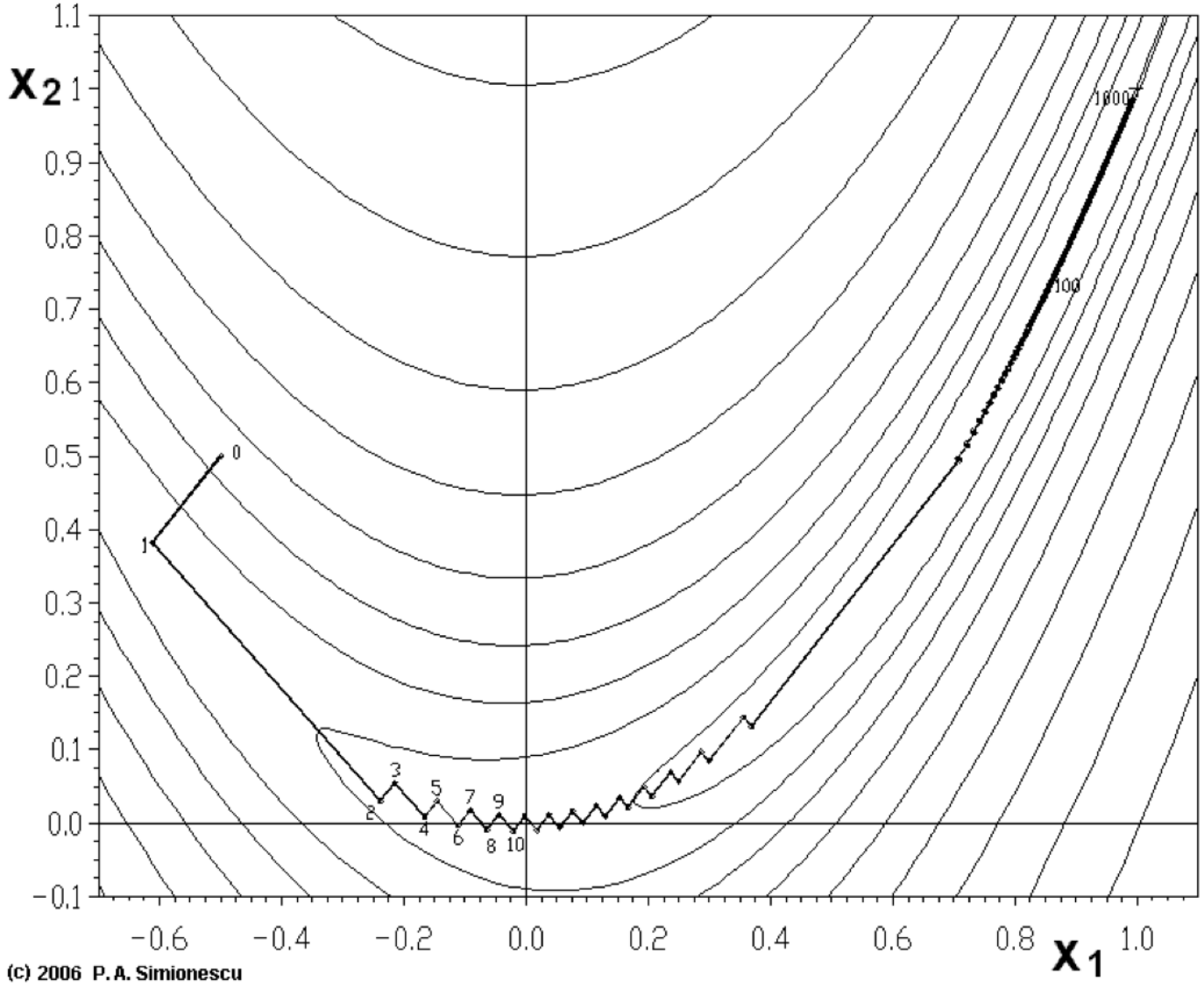
- On the basis of the foregoing discussion, a minimization procedure called **best-step steepest descent** can be described.
- At any given point $\mathbf{x}$, the gradient vector $\mathbf{G}(\mathbf{x})$ is calculated.
- Then a one-dimensional minimization problem is solved by determining the value t^* for which the function

$$\phi(t) = F(\mathbf{x} + t\mathbf{G}(\mathbf{x}))$$

  is a minimum.
- Then we replace $\mathbf{x}$ by $\mathbf{x} + t^*\mathbf{G}(\mathbf{x})$ and begin anew.

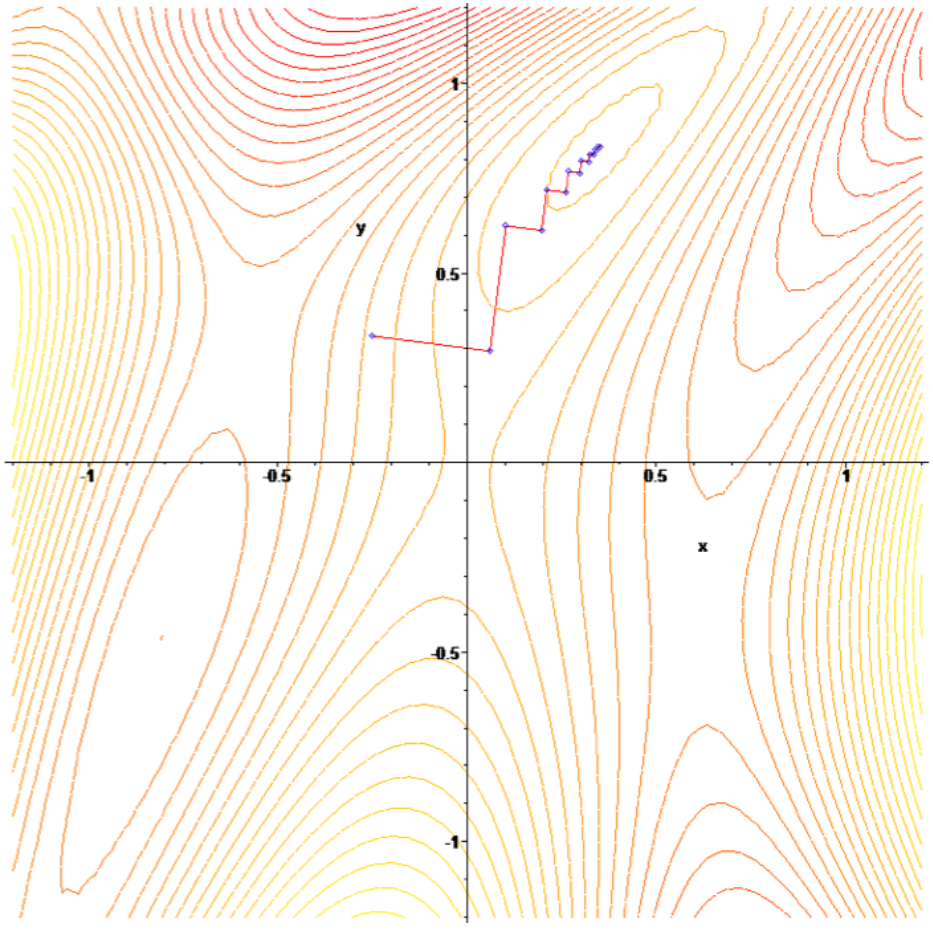
Example 1 Rosenbrock function

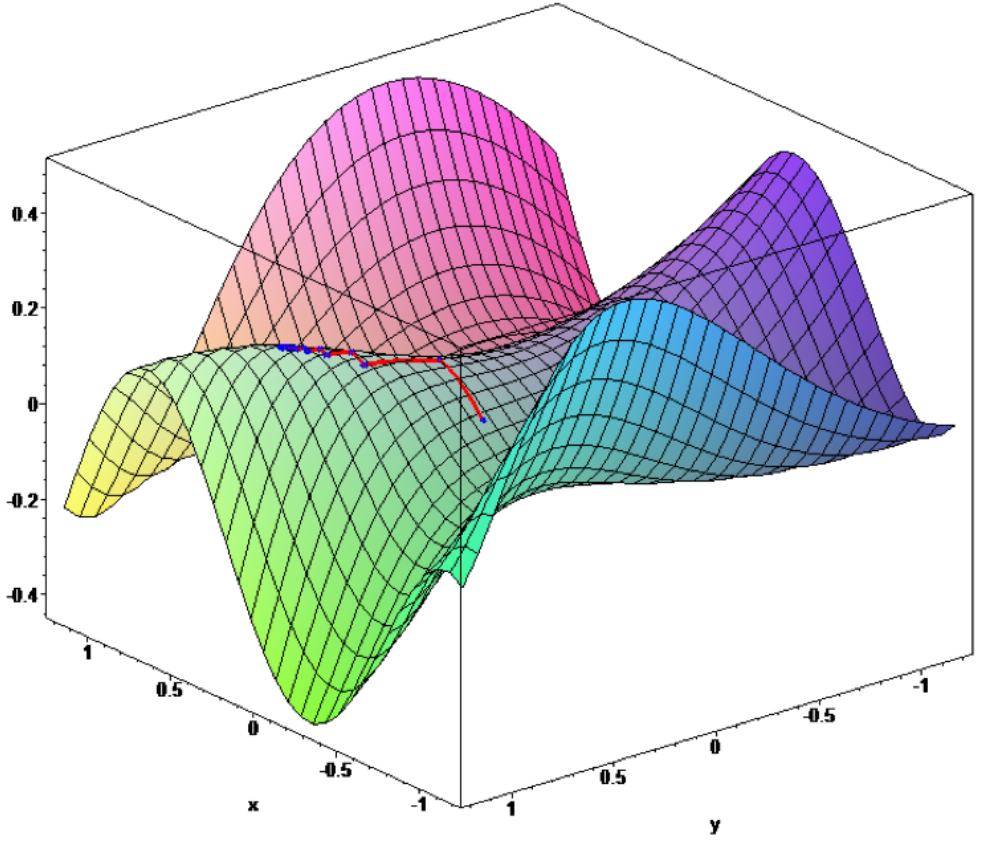
$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2.$$



Example 2

$$F(x,y) = \sin\left(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3\right)\cos(2x + 1 - e^y)$$





Summary

- The general method of steepest descent takes a step of any size in the direction of the negative gradient.
- It is not usually competitive with other methods, but it has the advantage of simplicity.
- One way of speeding it up is described in a computer exercise.

More Advanced Algorithms

- To explain more advanced algorithms, we consider a general real-valued function F of n variables.
- Suppose that we have obtained the first three terms in the Taylor series of F in the vicinity of a point $\mathbf{z}$.
- **How can they be used to guess the minimum point of F?**
- Obviously, we could ignore all terms beyond the quadratic terms and find the minimum of the resulting quadratic function:

$$F(\mathbf{x}+\mathbf{z}) = F(\mathbf{z}) + \mathbf{G}(\mathbf{z})^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}(\mathbf{z})\mathbf{x} + \cdots \tag{11}$$

- Here, $\mathbf{z}$ is fixed and $\mathbf{x}$ is the variable.

- To find the minimum of this quadratic function of $\mathbf{x}$, we must compute the first partial derivatives and set them equal to zero.
- Denoting this quadratic function by Q and simplifying the notation slightly, we have

$$Q(\mathbf{x}) = F(\mathbf{z}) + \sum_{i=1}^{n} G_i x_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} x_i H_{ij} x_j \tag{12}$$

from which it follows that

$$\frac{\partial Q}{\partial x_k} = G_k + \sum_{j=1}^{n} H_{kj} x_j \qquad (1 \leq k \leq n) \tag{13}$$

- The point $\mathbf{x}$ that is sought is thus a solution of the system of n equations

$$\sum_{j=1}^{n} H_{kj} x_j = -G_k \qquad (1 \leq k \leq n)$$

or, equivalently,

$$\mathbf{H}(\mathbf{z})\mathbf{x} = -\mathbf{G}(\mathbf{z}) \tag{14}$$

- The preceding analysis suggests the following iterative procedure for locating a minimum point of a function F:
- Start with a point $\mathbf{z}$ that is a current estimate of the minimum point.
- Compute the gradient and Hessian of F at the point $\mathbf{z}$.
- They can be denoted by $\mathbf{G}$ and $\mathbf{H}$, respectively.
- Of course, $\mathbf{G}$ is an n-component vector of numbers and $\mathbf{H}$ is an $n \times n$ matrix of numbers.
- Then solve the matrix equation

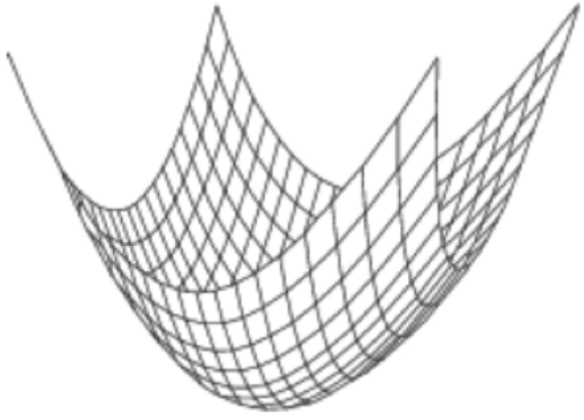
$$\mathbf{H}\mathbf{x} = -\mathbf{G}$$

  obtaining an n-component vector $\mathbf{x}$.
- Replace $\mathbf{z}$ by $\mathbf{z} + \mathbf{x}$ and return to the beginning of the procedure.

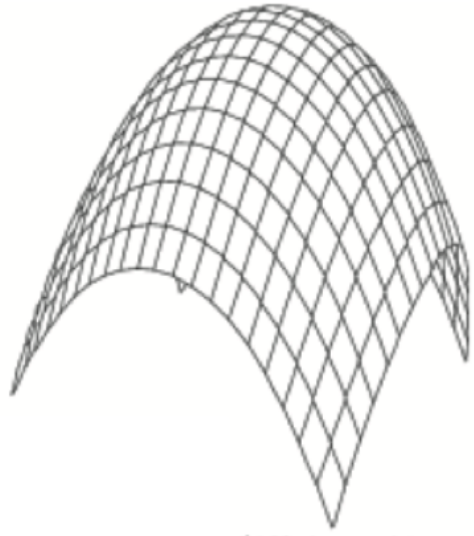
Minimum, Maximum, and Saddle Points

- There are many reasons for expecting trouble from the iterative procedure just outlined.
- One especially noisome aspect is that we can expect to find a point only where the first partial derivatives of F vanish; it need not be a minimum point.
- It is what we call a **stationary point**.
- Such points can be classified into three types:
  - **minimum point**
  - **maximum point**
  - **saddle point**

- In the following figures, they can be illustrated by simple quadratic surfaces familiar from analytic geometry:
  - **Minimum** $F(x, y) = x^2 + y^2$ at $(0, 0)$
  - **Maximum** $F(x, y) = 1 - x^2 - y^2$ at $(0, 0)$

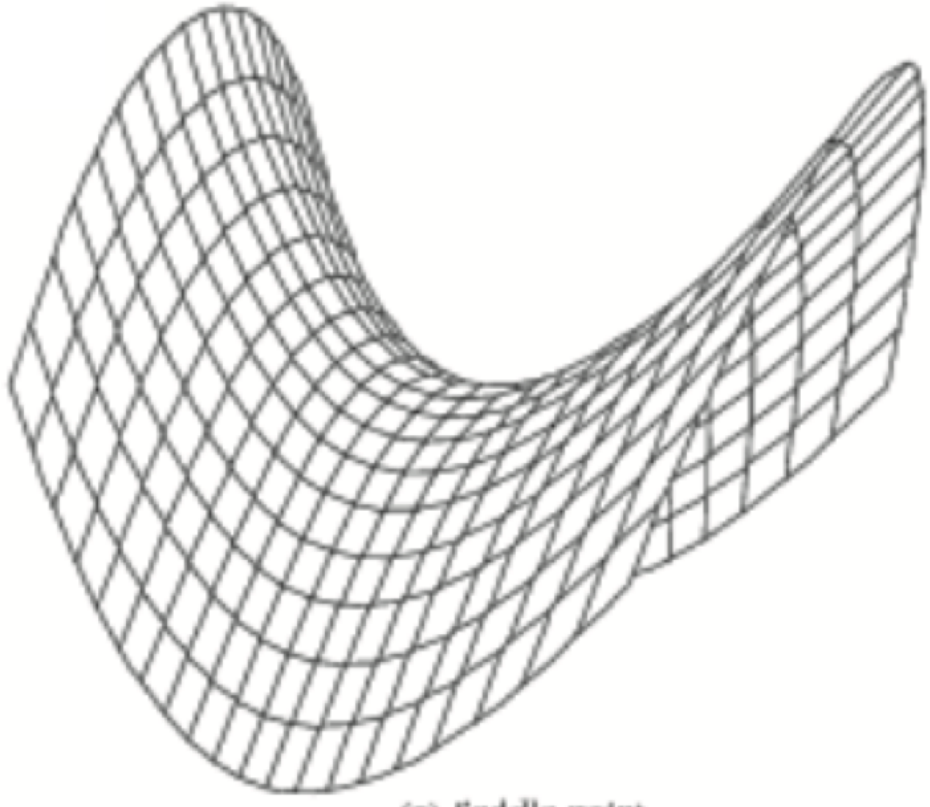


(a) Minimum point



(b) Maximum point

- **Saddle point** $F(x, y) = x^2 - y^2$ at $(0, 0)$



(c) Saddle point

Positive Definite Matrix

- If $\mathbf{z}$ is a stationary point of F, then

$$\mathbf{G}(\mathbf{z}) = 0$$

- Moreover, a criterion ensuring that Q, as defined in (12), has a minimum point is as follows:

Quadratic Function Theorem

Theorem 2

If the matrix $\mathbf{H}$ *has the property that* $\mathbf{x}^T\mathbf{H}\mathbf{x} > 0$ *for every nonzero vector* $\mathbf{x}$*, then the quadratic function* Q *has a minimum point.*

- A matrix that has this property is said to be **positive definite**.
- Notice that this theorem involves only second-degree terms in the quadratic function Q.

Quasi-Newton Methods

- Algorithms that converge faster than steepest descent in general and that are currently recommended for minimization are of a type called **quasi-Newton**.
- The principal example is an algorithm introduced in 1959 by Davidon, called the **variable metric algorithm**.
- These algorithms proceed iteratively, assuming in each step that a local quadratic approximation is known for the function F whose minimum is sought.

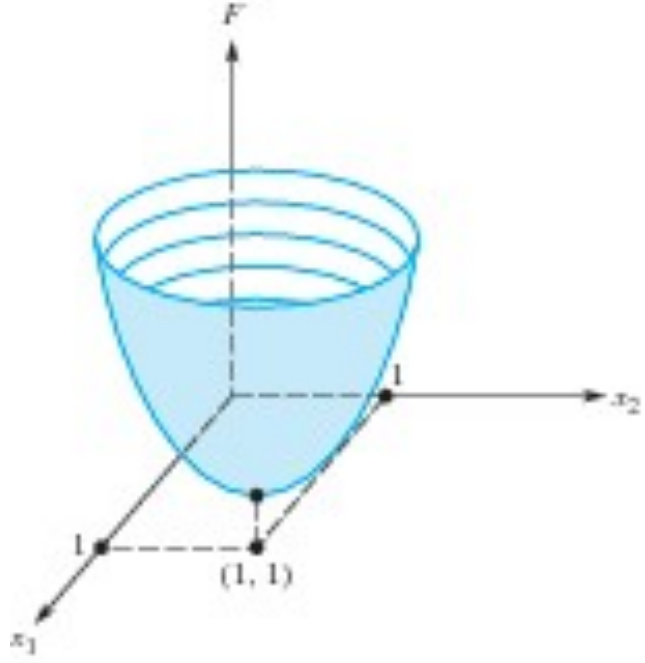
- The minimum of this quadratic function either provides the new point directly or is used to determine a line along which a one-dimensional search can be carried out.
- In implementation of the algorithm, the gradient can be either provided in the form of a procedure or computed numerically by finite differences.
- The Hessian **H** is not computed, but an estimate of its **LU** factorization is kept up to date as the process continues.

Functions in Matlab

fminunc: Unconstrained Minimization
https://www.mathworks.com/help/optim/ug/fminunc.html

fmincon: Find minimum of constrained nonlinear multivariable function

$$F(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 4$$



The unconstrained minimum occurs at (1, 1)

Nelder-Mead Algorithm

- For minimizing a function $F\colon \mathbb{R}^n \to \mathbb{R}$, another method called the **Nelder-Mead algorithm** is available.
- It is a method of **direct search** and proceeds without involving any derivatives of the function F and without any **line searches**.
- Before beginning the calculations, the user assigns values to three parameters: α, β, and γ.
- The default values are 1, $\frac{1}{2}$, and 1, respectively. In each step of the algorithm, a set of $n+1$ points in $\mathbb{R}^n$ is given: $\{x_0, x_1, \ldots, x_n\}$.

- This set is in **general position** in $\mathbb{R}^n$.
- This means that the set of n points $x_i - x_0$, with $1 \leq i \leq n$, is linearly independent.
- A consequence of this assumption is that the convex hull of the original set $\{x_0, x_1, \ldots, x_n\}$ is an **n-simplex**.

- For example, a **2-simplex** is a triangle in $\mathbb{R}^2$, and a **3-simplex** is a tetrahedron in $\mathbb{R}^3$.
- To make the description of the algorithm as simple as possible, we assume that the points have been relabeled (if necessary) so that $F(x_0) \geq F(x_1) \geq \cdots \geq F(x_n)$.
- Since we are trying to minimize the function F, the point x_0 is the worst of the current set, because it produces the highest value of F.

- We compute the point

$$u = \frac{1}{n} \sum_{i=1}^{n} x_i$$

- This is the **centroid** of the face of the current simplex opposite the worst vertex, x_0.
- Next, we compute a **reflected point** $v = (1 + \alpha)u - \alpha x_0$.
- If $F(v)$ is less than $F(x_n)$, then this is a favorable situation, and one is tempted to replace x_0 by v and begin anew.

- However, we first compute an **expanded reflected point** $w = (1 + \gamma)v - \gamma u$ and test to see whether $F(w)$ is less than $F(x_n)$.
- If so, we replace x_0 by w and begin anew.
- Otherwise, we replace x_0 by v as originally suggested and begin with the new simplex.

- Assume now that $F(v)$ is not less than $F(x_n)$.
- If $F(v) \leq F(x_1)$, then replace x_0 by v and begin again.
- Having disposed of all cases when $F(v) \leq F(x_1)$, we now consider two further cases.
- First, if $F(v) \leq F(x_0)$, then define $w = u + \beta(v - u)$.
- If $F(v) > F(x_0)$, compute $w = u + \beta(x_0 - u)$. With w now defined, test whether $F(w) < F(x_0)$.
- If this is true, replace x_0 by w and begin anew.
- However, if $F(w) \geq F(x_0)$, **shrink** the simplex by using

$$x_i \leftarrow \frac{1}{2}(x_i + x_n)$$

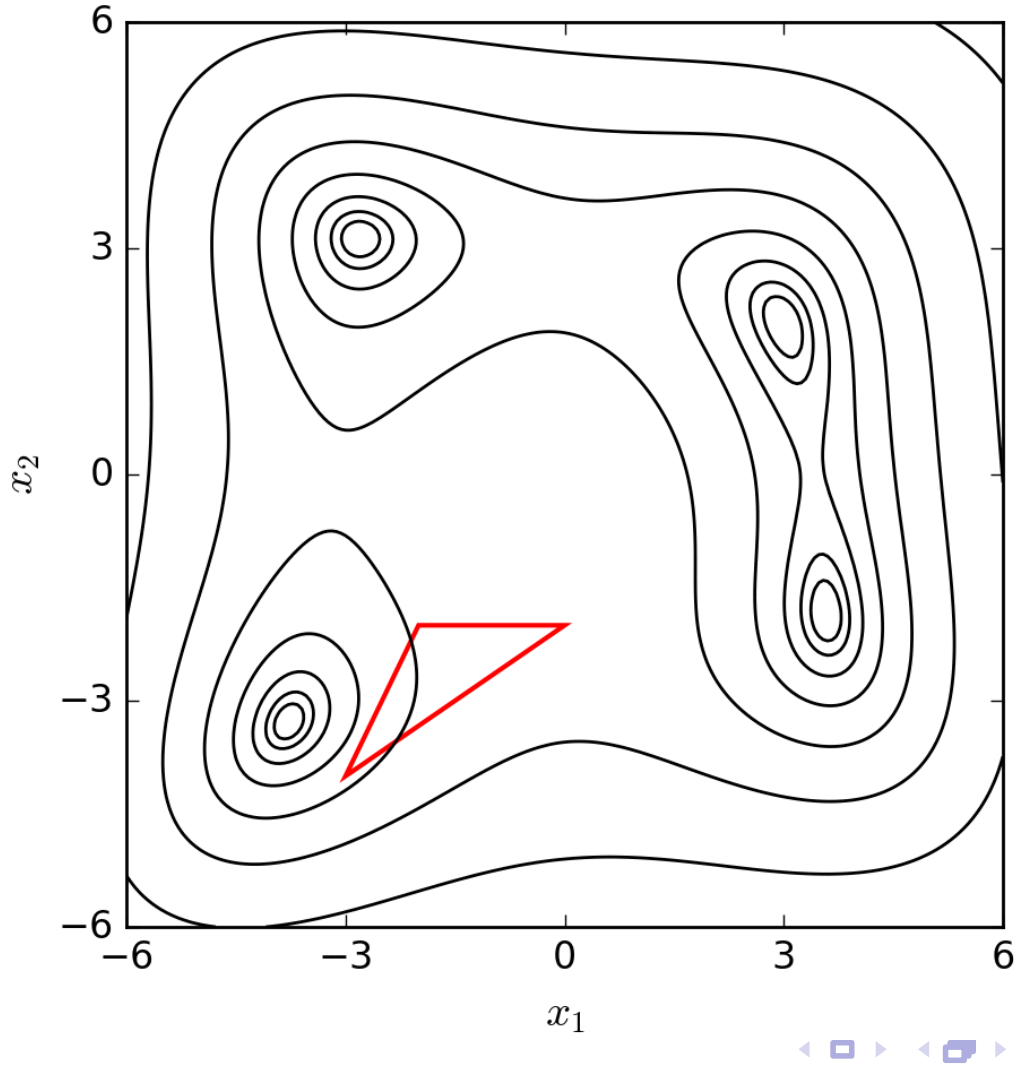
  for $0 \leq i \leq n - 1$.
- Then begin anew.

- The algorithm needs a stopping test in each major step.
- One such test is whether the relative **flatness** is small.
- That is the quantity

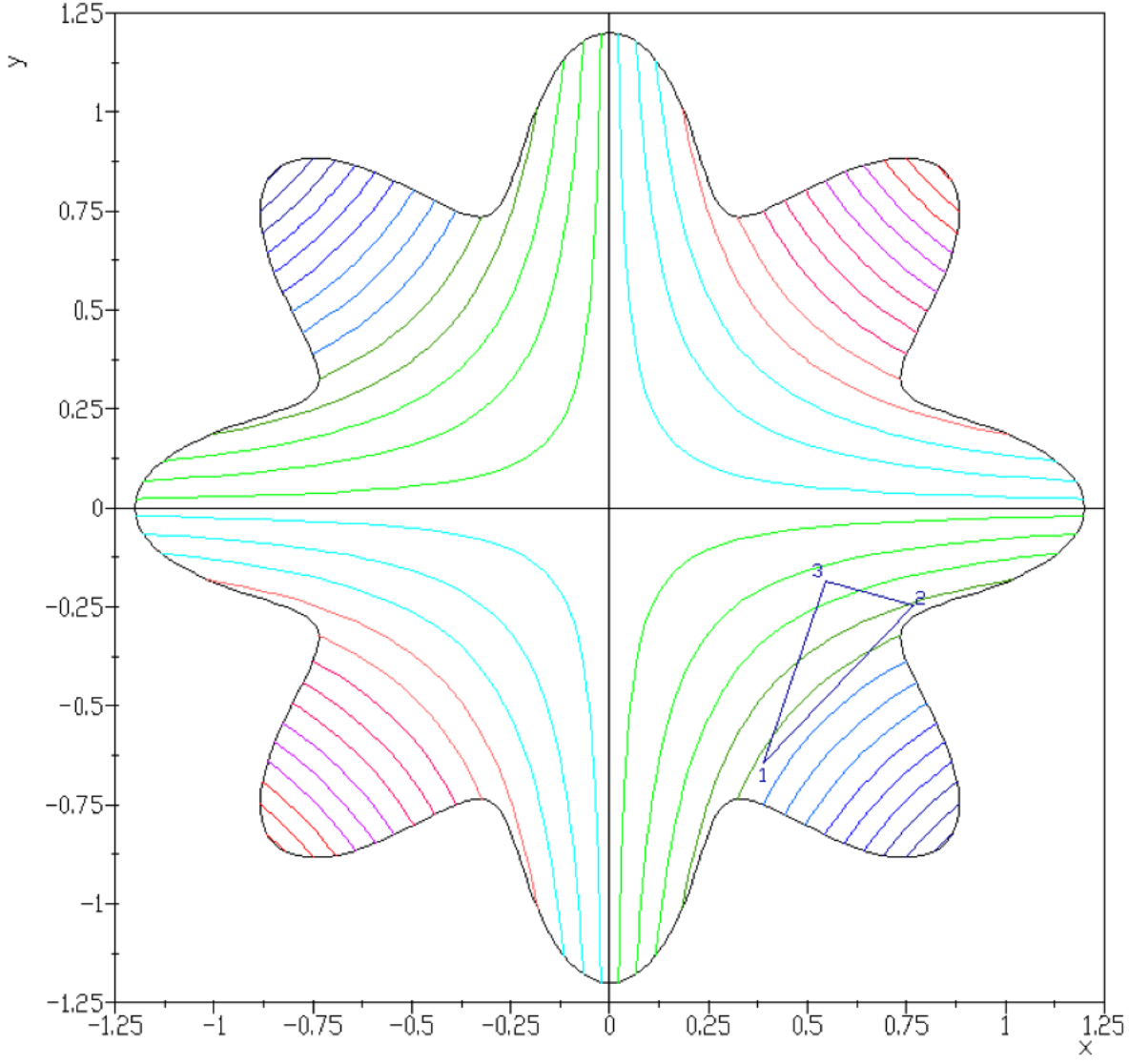
$$\frac{F(x_0) - F(x_n)}{|F(x_0)| + |F(x_n)|}$$

- Other tests to make sure progress is being made can be added.
- In programming the algorithm, one keeps the number of evaluations of f to a minimum.
- In fact, only three indices are needed: the indices of the greatest $F(x_i)$, the next greatest, and the least.

Nelder-Mead minimum search



Nelder-Mead minimum search



Method of Simulated Annealing

Method of Simulated Annealing

- This method has been proposed and found to be effective for the **minimization** of **difficult** functions, especially if they have many purely local minimum points.
- It involves no derivatives or **line searches**; indeed, it has found great success in minimizing discrete functions, such as arise in the **traveling salesman problem**.

- Suppose we are given a real-valued function of n real variables; that is, $F\colon \mathbb{R}^n \to \mathbb{R}$.
- We must be able to compute the values $F(x)$ for any x in $\mathbb{R}^n$.
- It is desired to locate a **global minimum point** of F, which is a point x^* such that $F(x^*) \leq F(x)$ for all x in $\mathbb{R}^n$.
- In other words, $F(x^*)$ is equal to $\inf_{x\in\mathbb{R}^n} F(x)$.
- The algorithm generates a sequence of points $x_1, x_2, x_3, \ldots$, and one hopes that $\min_{j\leq k} F(x_j)$ converges to $\inf F(x)$ as $k \to \infty$.

- It suffices to describe the computation that leads to x_{k+1}, assuming that x_k has been computed.
- We begin by generating a modest number of random points $u_1, u_2, \ldots, u_m$ in a large neighborhood of x_k.
- For each of these points, the value of F must be computed.
- The next point, x_{k+1}, in our sequence is chosen to be one of the points $u_1, u_2, \ldots, u_m$.
- This choice is made as follows.
- Select an index j such that

$$F(u_j) = \min\{F(u_1), F(u_2), \ldots, F(u_m)\}$$

- If $F(u_j) < F(x_k)$, then set $x_{k+1} = u_j$.

- In the other case, for each i, we assign a probability p_i to u_i by the formula

$$p_i = e^{\alpha[F(x_k)-F(u_i)]} \qquad (1 \leq i \leq m)$$

- Here, α is a positive parameter chosen by the user of the code.
- We normalize the probabilities by dividing each by their sum.
- That is, we compute

$$S = \sum_{i=1}^{m} p_i$$

and then carry out a replacement

$$p_i \leftarrow p_i/S$$

- Finally, a random choice is made among the points $u_1, u_2, \ldots, u_m$, taking account of the probabilities p_i that have been assigned to them.
- This randomly chosen u_i becomes x_{k+1}.

- The formula for the probabilities p_i is taken from the theory of thermodynamics.
- Presumably, other functions can serve in this role as well.
- **What is the purpose of the complicated choice for x_{k+1}?**
- Because of the possibility of encountering local minima, the algorithm must occasionally choose a point that is **uphill** from the current point.
- Then there is a chance that subsequent points might begin to move toward a different local minimum.
- An element of randomness is introduced to make this possible.

Simulated annealing searching for a maximum