

Practical Implementation of Unconstrained Optimization Methods Tested on Quadratic Functions

Awatif M. A. El Siddieg
Present address :Prince Sattam Bin Abdul-Aziz University Faculty of Sciences and Humanilities Studies
Math.Dep.
.Hotat Bani Tamim Kingdom of Saudi Arabia, P.O.Box 13
* E-mail address : wigdan1965@hotmail.com

Abstract:

In this work we give a detailed look to the Practical implementation of unconstrained optimization tested on quadratic functions. Section (1) speak about the theory of optimization problems, introduce definitions and theorems of linear programming problems , definitions and theorems of quadratic programming problems . Section (2) introduce some methods of unconstrained optimization. In section(3) we look at methods for approximating solutions to equations, solving unconstrained optimization problems .

Section(4) gives practical implementation of some method s using Matlab programs.

Section (1)

Introduction:

In this section we give theory of optimization problems, introduce definitions and theorems of linear programming problems , definitions and theorems of quadratic programming problems .

Theoretical background:

Definition(1): Linear programming problems:

Optimization might be defined as the science of determining the(best) solution to certain mathematically defined problems , which are often models of physical reality. It involves the study of optimality criteria for problems.

Example(1):

minimize :4a + 5b + 6(a + b) = 10a + 11b

subject to : $a + b \geq 11$

$a - b \leq 5$

$7a + 12b \geq 35$

$a \geq 0, b \geq 0$

the minimum occurs at a - b = 5 and a + b = 11

i.e. a = 8 and b = 3 with c = a + b = 11 and the value of the objective function 10a + 11b = 80 + 33 = 113.

Definition (2) :

A point $\underline{x} \in \Re^n$ is said to be a relative minimum point or a local $\Re^n$ if $\exists$ an $\mathcal{E} > 0$ such that $f(\underline{x}) \geq f(\underline{x}^*)$ $\forall \underline{x} \in \Re^n$ a strict relative minimum point of over $\Re^n$.

Definition (3): (constrained optimization problems)

The general form of a constrained optimization problem the form[9] :

$$\min_{\underline{x} \in \Re^n} f(\underline{x})$$

Subject to $$c_i(\underline{x}) = 0 , \ i = 1,2, \dots \dots , p \qquad (1)$$

$$c_i(\underline{x}) \geq 0, i = p + 1 , \dots \dots , n \qquad (2)$$

Where c_i is the i^{th} constraint function . the constraints $c_i(\underline{x}) = 0$ are termed equality constraints and the set of such . constraints is denoted by (E) and the constraints $c_i(\underline{x}) \geq 0$ are termed inequality constraints denoted by (I)

Proposition (1): (First order necessary condition)

Let S be a subset of $\mathfrak{R}^n$, and let f $\in \mathscr{C}^{(1)}$ be a function on S . If $\underline{x}^*$ is a relative any minimum point of f over $\underline{p}$, then for any f $\in \mathfrak{R}^n$, that is a feasible direction at $\underline{x}^*$, we have $\nabla f(\underline{x}^*)\underline{p}^T \geq 0$[1] .

Corollary (1) :

Let S be a subset of $\mathfrak{R}^n$, let $f \in \mathscr{C}^{(1)}$, be a function on $\mathfrak{R}^n$. If $\underline{x}^*$ is a relative minimum point of f and if $\underline{x}^*$ is an interior point of $\mathfrak{R}^n$, then

$$\nabla f(\underline{x}^*) = \underline{0}.$$

Descent directions at a point :

Consider the Taylor expansion of $f(\underline{x})$ about $\underline{x}'$ up to the first order term

$$f\left(\underline{x}' + \alpha \underline{p}\right) = f(\underline{x}') + \alpha \underline{p}^T g\left(x' + \alpha\theta \underline{p}\right)$$

Where,$0< \theta < 1$, $\alpha > 0$.

Since $f(\underline{x})$ is smooth enough (i.e, all the partial derivatives are continuous), then $\underline{p}^T g(\underline{x})$ *is* $< 0, \forall \underline{x}$,then sufficiently close to $\underline{x}'$ (by continuity). Thus if α is taken sufficiently small.

$$\underline{p}^T g\left(\underline{x}' + \alpha\theta \underline{p}\right) < 0$$

More precisely $\exists \overline{\alpha} > 0$ such that

$$\underline{p}^T g\left(\underline{x}' + \alpha\theta \underline{p}\right) < 0 \qquad \forall \alpha \in [0, \overline{\alpha}]$$

Thus $f\left(\underline{x}' + \alpha \underline{p}\right) < f(\underline{x}')$

we notice that if $\underline{p}^T g' < 0$, then the value of f decreases (locally if we move in the direction $\underline{p}$) .

Such a direction $\underline{p}$ is called a descent direction at $\underline{x}'$, and it chamcterized . by :

$$\underline{p}^T g' < 0$$

An example of a descent direction at $\underline{x}'$ is $\underline{p} = -g'$since

$$-g'^{T} g' < 0 \text{ provided } \underline{g'} \neq \underline{0}$$

Definition (5): (Definite and semi definite matrices)

Let C be symmetric matrix we say that C is positive definite if $\underline{x}^T \, C\underline{x} > 0$ $\forall \underline{x} \in \Re^n$,$\underline{x} \neq \underline{0}$, *C* is called positive semi definite if

$\underline{x}^T C\underline{x} \geq 0$ *for* $\forall \, \underline{x} \in \Re^n$

Section (2)

Introduction: In this section we introduce some unconstrained optimization methods tested on quadratic functions .

Definition(6): (Unconstrained optimization problems)

The problem takes the form:

$$\text{minimize } f\left(\underline{x}\right).$$

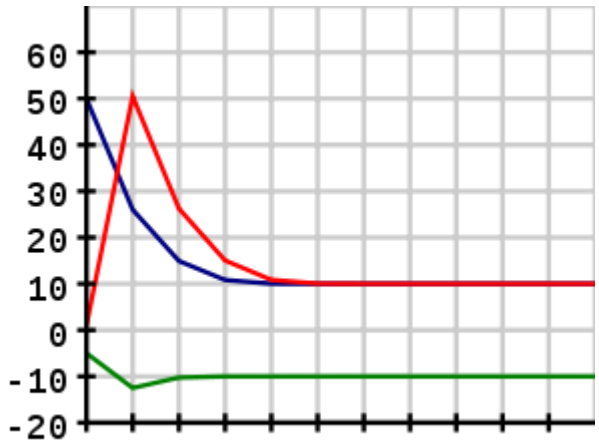
$$\text{Subject to } \underline{x} \in \Re^n$$

Where f is a continuous real valued function

Definition(7):

Quadratic programming (QP) is a special type of mathematical optimization problem. It is the problem of optimizing (minimizing or maximizing) a quadratic function of several variables subject to linear constraints on these variables .

1-Babylonian method:



Graph charting the use of the Babylonian method for approximating the square root of 100 (10) using starting values $x_0 = 50$, $x_0 = 1$, and $x_0 = -5$. Note that using a negative starting value yields the negative root [3].

Perhaps the first algorithm used for approximating $\sqrt{S}$ is known as the "Babylonian method", named after the Babylonians, or "Heron's method", named after the first-century Greek mathematician Hero of Alexandria who gave the first explicit description of the method. It can be derived from (but predates by many centuries) Newton's method. The basic idea is that if x is an overestimate to the square root of a non-negative real number S then S/x will be an underestimate and so the average of these two numbers may reasonably be expected to provide a better approximation (though the formal proof of that assertion depends on the inequality of arithmetic and geometric means that shows this average is always an overestimate of the square root, as noted in the article on square roots, thus assuring convergence)[3]. This is a quadratically convergent algorithm, which means that the number of correct digits of the approximation roughly doubles with each iteration. It proceeds as follows:

1. Begin with an arbitrary positive starting value x_0 (the closer to the actual square root of S, the better).
2. Let x_{n+1} be the average of x_n and S / x_n (using the arithmetic mean to approximate the geometric mean).
3. Repeat step 2 until the desired accuracy is achieved.

It can also be represented as:

$$x_0 \approx \sqrt{S}.$$
$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right),$$
$$\sqrt{S} = \lim_{n\to\infty} x_n.$$

This algorithm works equally well in the p-adic numbers, but cannot be used to identify real square roots with p-adic square roots; it is easy, for example, to construct a sequence of rational numbers by this method that converges to +3 in the reals, but to −3 in the 2-adics [5].

Example (2): Calculate $\sqrt{S}$, where S = 125348, to 6 significant figures, use the rough estimation method above to get x_0. The number of digits in S is D = 6 = 2 ∙2 + 2. So, n = 2 and the rough estimate is:

$$x_0 = 6 \cdot 10^2 = 600.000.$$
$$x_1 = \frac{1}{2}\left(x_0 + \frac{S}{x_0}\right) = \frac{1}{2}\left(600.000 + \frac{125348}{600.000}\right) = 404.457.$$
$$x_2 = \frac{1}{2}\left(x_1 + \frac{S}{x_1}\right) = \frac{1}{2}\left(404.457 + \frac{125348}{404.457}\right) = 357.187.$$
$$x_3 = \frac{1}{2}\left(x_2 + \frac{S}{x_2}\right) = \frac{1}{2}\left(357.187 + \frac{125348}{357.187}\right) = 354.059.$$

$$x_4 = \frac{1}{2}\left(x_3 + \frac{S}{x_3}\right) = \frac{1}{2}\left(354.059 + \frac{125348}{354.059}\right) = 354.045.$$

$$x_5 = \frac{1}{2}\left(x_4 + \frac{S}{x_4}\right) = \frac{1}{2}\left(354.045 + \frac{125348}{354.045}\right) = 354.045.$$

Therefore, $\sqrt{125348} \approx 354.045$.

Convergence:

Let the relative error in x_n be defined by

$$\varepsilon_n = \frac{x_n}{\sqrt{S}} - 1$$

and thus

$$x_n = \sqrt{S} \cdot (1 + \varepsilon_n).$$

Then it can be shown that

$$\varepsilon_{n+1} = \frac{\varepsilon_n^2}{2(1 + \varepsilon_n)}$$

and thus that

$$0 \leq \varepsilon_{n+2} \leq \min\left\{\frac{\varepsilon_{n+1}^2}{2}, \frac{\varepsilon_{n+1}}{2}\right\}$$

and consequently that convergence is assured provided that x_0 and S are both positive.

Worst case for convergence:

If using the rough estimate above with the Babylonian method, then the worst cases are:

$$\begin{aligned}
&S = 1; \quad x_0 = 2; \quad x_1 = 1.250; \quad \varepsilon_1 = 0.250.\\
&S = 10; \quad x_0 = 2; \quad x_1 = 3.500; \quad \varepsilon_1 < 0.107.\\
&S = 10; \quad x_0 = 6; \quad x_1 = 3.833; \quad \varepsilon_1 < 0.213.\\
&S = 100; \quad x_0 = 6; \quad x_1 = 11.333; \quad \varepsilon_1 < 0.134.
\end{aligned}$$

Thus in any case,

$$\varepsilon_1 \leq 2^{-2}.$$

$$\varepsilon_2 < 2^{-5} < 10^{-1}.$$

$$\varepsilon_3 < 2^{-11} < 10^{-3}.$$

$$\varepsilon_4 < 2^{-23} < 10^{-6}.$$

$$\varepsilon_5 < 2^{-47} < 10^{-14}.$$

$$\varepsilon_6 < 2^{-95} < 10^{-28}.$$

$$\varepsilon_7 < 2^{-191} < 10^{-57}.$$

$$\varepsilon_8 < 2^{-383} < 10^{-115}.$$

Remember that rounding errors will slow the convergence. It is recommended to keep at least one extra digit beyond the desired accuracy of the x_n being calculated to minimize round off error.

2- Bakhshali approximation: This method for finding an approximation to a square root was described in an ancient Indian mathematical manuscript called the Bakhshali manuscript. It is equivalent to two iterations of the

Babylonian method beginning with N [2]. The original presentation goes as follows: To calculate $\sqrt{S}$, let N^2 be the nearest perfect square to S. Then, calculate:

$$d = S - N^2$$
$$P = \frac{d}{2N}$$
$$A = N + P$$
$$\sqrt{S} \approx A - \frac{P^2}{2A}$$

This can be also written as:

$$\sqrt{S} \approx N + \frac{d}{2N} - \frac{d^2}{8N^3 + 4Nd} = \frac{8N^4 + 8N^2d + d^2}{8N^3 + 4Nd} = \frac{N^4 + 6N^2S + S^2}{4N^3 + 4NS}$$

Example (3) by discussion:

Consider the perfect square $2809 = 53^2$. Use the duplex method to find the square root of 2,809.

Solution:

- Set down the number in groups of two digits.
- Define a divisor, a dividend and a quotient to find the root.
- Given 2809. Consider the first group, 28.
  - Find the nearest perfect square below that group.
  - The root of that perfect square is the first digit of our root.
  - Since $28 > 25$ and $25 = 5^2$, take 5 as the first digit in the square root.
  - For the divisor take double this first digit ($2 \cdot 5$), which is 10.
- Next, set up a division framework with a colon.

- 28: 0 9 is the dividend and 5: is the quotient.
- Put a colon to the right of 28 and 5 and keep the colons lined up vertically. The duplex is calculated only on quotient digits to the right of the colon.

- Calculate the remainder. 28: minus 25: is 3:
  - Append the remainder on the left of the next digit to get the new dividend.
  - Here, append 3 to the next dividend digit 0, which makes the new dividend 30. The divisor 10 goes into 30 just 3 times. (No reserve needed here for subsequent deductions.)
- Repeat the operation.
  - The zero remainder appended to 9. Nine is the next dividend.
  - This provides a digit to the right of the colon so deduct the duplex, $3^2 = 9$.
  - Subtracting this duplex from the dividend 9, a zero remainder results.
  - Ten into zero is zero. The next root digit is zero. The next duplex is $2(3\ 0) = 0$.
  - The dividend is zero. This is an exact square root, 53.

Example (4): (analysis and square root framework):

Find the square root of 2809.

Solution :

Set down the number in groups of two digits.

The number of groups gives the number of whole digits in the root.

Put a colon after the first group, 28, to separate it.

From the first group, 28, obtain the divisor, 10, since

$28 > 25=5^2$ and by doubling this first root, 2x5=10.

Gross dividend: 28: 0 9. Using mental math:

Divisor: 10) 3 0 Square: 10) 28: $_3$0 9

Duplex, Deduction: 25: xx 09 Square root: 5: 3. 0

Dividend: 30 00

Remainder: 3: 0 0 0 0

Square Root, Quotient: 5: 3. 0

3 -Taylor series:

If N is an approximation to $\sqrt{S}$, a better approximation can be found by using the Taylor series of the square root function:

$$\sqrt{N^2+d} = \sum_{n=0}^{\infty} \frac{(-1)^n (2n)! d^n}{(1-2n)n!^2 4^n N^{2n-1}} = N + \frac{d}{2N} - \frac{d^2}{8N^3} + \frac{d^3}{16N^5} - \frac{5d^4}{128N^7} + \cdots$$

As an iterative method, the order of convergence is equal to the number of terms used. With 2 terms, it is identical to the Babylonian method; With 3 terms, each iteration takes almost as many operations as the Bakhshali approximation, but converges more slowly. Therefore, this is not a particularly efficient way of calculation.

Section(3)

Unconstrained optimization methods:

Introduction : In this section we are going to look at some unconstrained optimization algorithms and test them on non linear functions.

1 - Newton's Method:

Introduction: Newton's Method attempts to construct a sequence x_n from an initial guess x_0 that converges towards x_*such that $f'(x_*) = 0$. This is x_* called a stationary point of f.

The second order Taylor expansion $f_T(x)$ of the function around x_n (where $\Delta x = x - x_n$) is: $f_T(x_n + \Delta x) = f_T(x) = f(x_n) + f'(x_n)\Delta x + \frac{1}{2}f''(x_n)\Delta x^2$, attains its extremum when its derivative with respect to Δx is equal to zero, i.e. when Δx solves the linear equation:

$$f'(x_n) + f''(x_n)\Delta x = 0.$$

(Considering the right-hand side of the above equation as a quadratic in Δx, with constant coefficients.)

Thus, provided that $f(x)$ is a twice-differentiable function well approximated by its second order Taylor expansion and the initial guess x_0 is chosen close enough to x_*, the sequence (x_n) defined by: $\Delta x = x - x_n = -\frac{f'(x_n)}{f''(x_n)}$

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}, \quad n = 0, 1, \ldots$$

will converge towards a root of f', i.e. x_* for which $f'(x_*) = 0$.

Newton's method (also known as the Newton–Raphson method), named after Isaac Newton and Joseph Raphson, is a method for finding successively better approximations to the roots (or zeroes) of a real-valued function. The algorithm is first in the class of Householder's methods, succeeded by Halley's method.

2- The Newton-Raphson method in one variable:

Given a function $f(x)$ and its derivative $f'(x)$, we begin with a first guess x_0 for a root of the function. Provided the function is reasonably well-behaved a better approximation x_1 is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Geometrically, x_1 is the intersection with the x-axis of a line tangent to f at $f(x_0)$.

The process is repeated until a sufficiently accurate value is reached:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

The idea of the method is as follows: one starts with an initial guess which is reasonably close to the true root, then the function is approximated by its tangent line (which can be computed using the tools of calculus), and one computes the x-intercept of this tangent line (which is easily done with elementary algebra). This x-intercept will typically be a better approximation to the function's root than the original guess, and the method can be iterated [5], [9].

Suppose f : [a, b] → R is a differentiable function defined on the interval [a, b] with values in the real numbers R. The formula for converging on the root can be easily derived. Suppose we have some current approximation x_n. Then we can derive the formula for a better approximation, x_{n+1} by referring to the diagram on the right. We know from the definition of the derivative at a given point that it is the slope of a tangent at that point.

That is

$$f'(x_n) = \frac{\Delta y}{\Delta x} = \frac{f(x_n) - 0}{x_n - x_{n+1}}.$$

Here, f' denotes the derivative of the function f. Then by simple algebra we can derive

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

We start the process off with some arbitrary initial value x_0. (The closer to the zero, the better. But, in the absence of any intuition about where the zero might lie, a "guess and check" method might narrow the possibilities to a reasonably small interval by appealing to the intermediate value theorem.) The method will usually converge, provided this initial guess is close enough to the unknown zero, and that $f'(x_0) \neq 0$. Furthermore, for a zero of multiplicity 1, the convergence is at least quadratic (see rate of convergence) in a neighborhood of the zero, which intuitively means that the number of correct digits roughly at least doubles in every step. More details can be found in the analysis section below.

The Householder's methods are similar but have higher order for even faster convergence. However, the extra computations required for each step can slow down the overall performance relative to Newton's method, particularly if f or its derivatives are computationally expensive to evaluate [8].

Proof of quadratic convergence for Newton's iterative method:

According to Taylor's theorem, any function f(x) which has a continuous second derivative can be by an expansion about a point that is close to a root of f(x). Suppose this root is α .Then the expansion of f(α) about x_n is:

$$f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + R_1 \qquad (1)$$

where the Lagrange form of the Taylor series expansion remainder is

$$R_1 = \frac{1}{2!} f''(\xi_n)(\alpha - x_n)^2,$$

where ξ_n is in between x_n and α.

Since $\alpha$is the root, (1) becomes:

$$0 = f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2} f''(\xi_n)(\alpha - x_n)^2 \tag{2}$$

Dividing equation (2) by $f'(x_n)$and rearranging gives

$$\frac{f(x_n)}{f'(x_n)} + (\alpha - x_n) = \frac{-f''(\xi_n)}{2f'(x_n)} (\alpha - x_n)^2 \tag{3}$$

Remembering that x_{n+1} is defined by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{4}$$

one finds that

$$\underbrace{\alpha - x_{n+1}}_{\epsilon_{n+1}} = \frac{-f''(\xi_n)}{2f'(x_n)} (\underbrace{\alpha - x_n}_{\epsilon_n})^2.$$

That is,

$$\epsilon_{n+1} = \frac{-f''(\xi_n)}{2f'(x_n)} \epsilon_n{}^2. \tag{5}$$

Taking absolute value of both sides gives

$$|\epsilon_{n+1}| = \frac{|f''(\xi_n)|}{2\,|f'(x_n)|}\,{\epsilon_n}^2 \tag{6}$$

Equation (6) shows that the rate of convergence is quadratic if 1- $f'(x) \neq 0; \forall x \in I$, where I is the interval $[\alpha - r, \alpha + r]$ for some $r \geq |(\alpha - x_0)|$; following conditions are satisfied:

1. $f''(x)$ is finite , $\forall x \in I$;
2. x_0 sufficiently close to the root α

The term sufficiently close in this context means the following:

(a) Taylor approximation is accurate enough such that we can ignore higher order terms,

(b) $$\frac{1}{2}\left|\frac{f''(x_n)}{f'(x_n)}\right| < C\left|\frac{f''(\alpha)}{f'(\alpha)}\right|, \text{ for some } C < \infty,$$

(c)

$$C\left|\frac{f''(\alpha)}{f'(\alpha)}\right|\epsilon_n < 1, \text{ for } n \in \mathrm{Z}^+ \cup \{0\} \text{ and } C \text{ satisfying condition (b)}.$$

Finally, (6) can be expressed in the following way:

$$|\epsilon_{n+1}| \leq M{\epsilon^2}_n$$

where M is the supremum of the variable coefficient of ϵ^2 on the interval I defined in the condition 1, that is:

$$M = \sup_{x \in I} \frac{1}{2} \left| \frac{f''(x)}{f'(x)} \right|.$$

The initial point x_0 has to be chosen such that conditions (1) through (3) are satisfied, where the third condition requires that $M\,|\epsilon_0| < 1.$

Minimization and maximization problems (Newton's method in optimization):

Newton's method can be used to find a minimum or maximum of a function. The derivative is zero at a minimum or maximum, so minima and maxima can be found by applying Newton's method to the derivative. The iteration becomes:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}.$$

Example(5) : Let's start by computing $\sqrt{5}$. Of course, this is easy if you have a calculator, but it is a simple example which will illustrate a more general method.

First, we'll think about the problem in a slightly different way. We are looking for $\sqrt{5}$ which is a solution of the equation $f(x) = x^2 - 5 = 0$.

The problem is that it is difficult to generate a numerical solution to this equation. But remember in the section on approximations , we saw how to approximate a function near a given point by its tangent line. The idea here will

be to actually solve the approximate equation which is easy since it is a linear one.

If we think for a minute, we know that $\sqrt{5}$ is between 2 and 3 so let's just choose to use the linear approximation at $x_0 = 2$. We know that $f(x) = x^2 - 5$ so that $f'(x) = 2x$. The linear approximation is then

$$\begin{aligned} f(x) \approx l(x) &= f(x_0) + f'(x_0)(x - x_0) \\ f(x) \approx l(x) &= -1 + 4(x - 2) = 4x - 9 \end{aligned}$$

Notice that the linear equation is easy to solve. We will then approximate the solution to $f(x) = 0$ by the solution to $l(x) = 4x - 9 = 0$ which is $x = \frac{9}{4} = 2.25$. If you have a look on a calculator, you will see that $\sqrt{5} = 2.236068...$. So you can see that we have found a fairly good approximation.

We can understand what we have done graphically. We are looking for a solution to $f(x) = 0$ which is where the graph of $f(x)$ crosses the X-axis. We approximate that point by the point where the tangent line crosses the X-axis

Now this is where the story becomes interesting since we can repeat what we have just done using the new approximate solution. That is, we will call $x_1 = \frac{9}{4}$ and consider the linear approximation at that point.

$$\begin{aligned} f(x) \approx l(x) &= f(x_1) + f'(x_1)(x - x_1) \\ f(x) \approx l(x) &= (\frac{81}{16} - 5) + \frac{9}{2}(x - \frac{9}{4}) = \frac{9}{2}x - \frac{161}{16} \end{aligned}$$

Now if we call x_2 the solution to $l(x) = 0$, we find that $x_2 = 2.236111$ which is an even better approximate solution to the equation. We could continue this process

generating better approximations to $\sqrt{5}$ at every step. This is the basic idea of a technique known as Newton's Method

The general method:

More generally, we can try to generate approximate solutions to the equation $f(x) = 0$ using the same idea. Suppose that x_0 is some point which we suspect is near a solution. We can form the linear approximation at x_0 and solve the linear equation instead.

That is, we will call x_1 the solution to $l(x) = f(x_0) + f'(x_0)(x - x_0) = 0$. In other words,

$$\begin{aligned} f(x_0) + f'(x_0)(x_1 - x_0) &= 0 \\ x_1 - x_0 &= -\frac{f(x_0)}{f'(x_0)} \\ x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \end{aligned}$$

If our first guess x_0 was a good one, the approximate solution x_1 should be an even better approximation to the solution of $f(x) = 0$. Once we have x_1, we can repeat the process to obtain x_2, the solution to the linear equation

$$l(x_2) = f(x_1) + f'(x_1)(x_2 - x_1) = 0$$

Solving in the same way, we see that

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

Maybe now you see that we can repeat this process indefinitely: from x_2, we generate x_3 and so on. If, after n steps, we have an approximate solution x_n, then the next step is

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Provided we have started with a good value for x_0, this will produce approximate solutions to any degree of accuracy.

3-The gradient –descent method:

The gradient –descent method is the simplest Newton –type for nonlinear optimization .The price for this simplicity is that the method is hopelessly inefficient at solving most problems . The method has theoretical uses , though , in proving the convergence of other methods, and in providing lower bounds on the performance of better algorithms . It is well known and has been widely used and discussed, so it is worthwhile being familiar with it if only to know not to use it on general problems . The gradient –descent is old , but not as old as Newton's method ,it is much simpler than Newton's method .

The gradientt –descent method computes the search direction form

$$p_k = -\nabla f(x_k)$$

and then uses a line search to determine $x_{k+1} = x_k + \alpha_k p_k$.

Hence the cost of computing the search direction is just the cost of computing the gradient . Since the gradient must be computed to determine if the solution has been found , it is reasonable to say that the search direction is available for free. The search direction is a descent direction if $\nabla f(x_k) \neq 0$, that is , it is a descent direction unless x_k

is a stationary point of the function f .

The formula for the search direction can be derived in two ways, both of which have connections of Newton's method

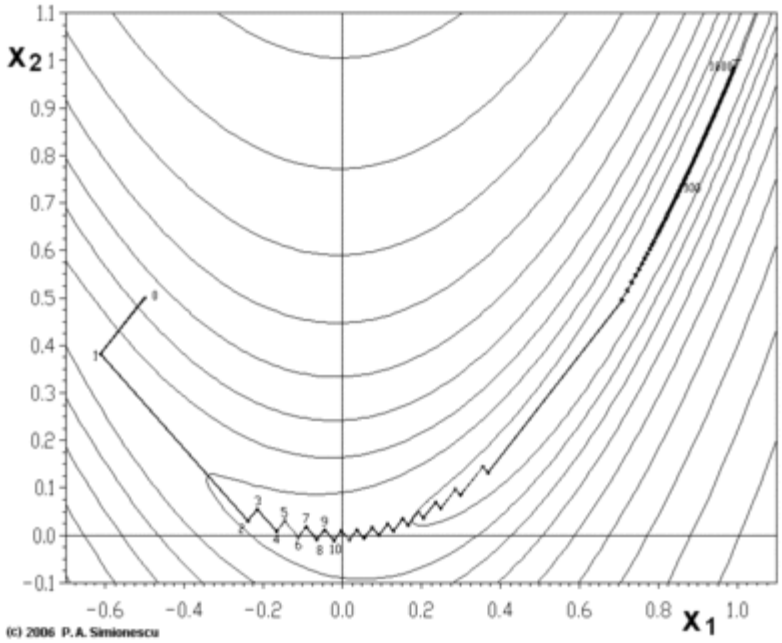
($((\nabla^2 f)p = -\nabla f)$ but with the Hessian approximated by the identity matrix $\nabla^2 \mathrm{f} \approx \mathrm{I}$, then the formula for the steepest − descent method is

obtained .

Gradient descent has problems with pathological functions such as the Rosenbrock function shown here:

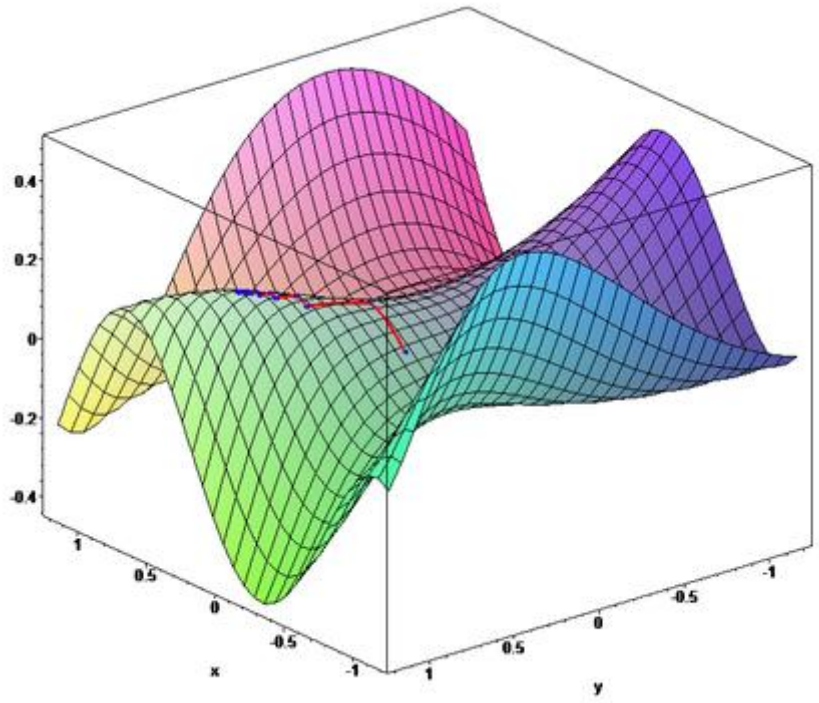
$$f(x_1, x_2) = (1 - {x_1}^2) + 100(x_2 - {x_1}^2)^2$$

The Rosenbrock function has a narrow curved valley which contains the minimum. The bottom of the valley is very flat. Because of the curved flat valley the optimization is zig-zagging slowly with small stepsizes towards the minimum.



The "Zig-Zagging" nature of the method is also evident below, where the gradient descent method is applied to:

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



Gradient descent can also be used to solve a system of nonlinear equations. Below is an example that shows how to use the gradient descent to solve for three unknown variables, x_1, x_2, and x_3. This example shows one iteration of the gradient descent.

Example(6): Consider the non linear system of equations:

$$\begin{aligned} 3x_1 - cos(x_2x_3) - \frac{3}{2} &= 0 \\ 4{x_1}^2 - 625{x_2}^2 + 2x_2 - 1 &= 0 \\ e^{(-x_1x_2)} + 20x_3 + \frac{1}{3}(10\pi - 3) &= 0 \end{aligned}$$

suppose we have the function:

$$G(\underline{x}) = \begin{bmatrix} 3x_1 - cos(x_2x_3) - \frac{3}{2} \\ 4{x_1}^2 - 625{x_2}^2 + 2x_2 - 1 \\ e^{(-x_1x_2)} + 20x_3 + \frac{1}{3}(10\pi - 3) \end{bmatrix}$$

Where $\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, the objective function $F(\underline{x}) = \frac{1}{2}G^T(\underline{x})G(\underline{x})$

$$= \frac{1}{2}((3x_1 - cos(x_2x_3) - \frac{3}{2})^2 + (4x_1 - 625{x_2}^2 + 2x_2 - 1)^4 + e^{(-x_1x_2)} + 20x_3 + \frac{1}{3}(10\pi - 3)^2)$$

with the initial guess: $\underline{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

$$\underline{x}^{(1)} = \underline{x}^{(0)} - \gamma_0 \nabla F(\underline{x}^{(0)})$$

where $\nabla F(\underline{x}^{(0)}) = J_G(\underline{x}^{(0)})^T . G(\underline{x}^{(0)})$

the Jacobean matrix $J_G(\underline{x}^{(0)})$

$$J_G = \begin{bmatrix} 3 & sin(x_2x_3)x_3 & sin(x_2x_3)x_2 \\ 8x_1 & -1250x_2 + 2 & 0 \\ -x_2e^{(-x_1x_2)} & -x_1e^{(-x_1x_2)} & 20 \end{bmatrix}$$

Then evaluating these items at $\underline{x}^{(0)}$

$$J_G(\underline{x}^{(0)})^T = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 20 \end{bmatrix}, \qquad G(\underline{x}^{(0)}) = \begin{bmatrix} -25 \\ -1 \\ 10.472 \end{bmatrix}$$

so that $\underline{x}^{(1)} = 0 - \gamma_0 \begin{bmatrix} -7.5 \\ -2 \\ 209.44 \end{bmatrix}$

and $F(\underline{x}^{(0)}) = \frac{1}{2}((-2.5)^2 + (-1)^2 + (10.472)^2) = 58.450$

for suitable γ_0 must be found such that

$F(\underline{x}^{(1)}) \leq F(\underline{x}^{(0)})$, this can be done with any of a variety of line search algorithm.

One might also simply guess $\gamma_0 = 0.001$ which gives : $\underline{x}^{(1)} = \begin{bmatrix} 0.0075 \\ 0.002 \\ -0.209 \end{bmatrix}$

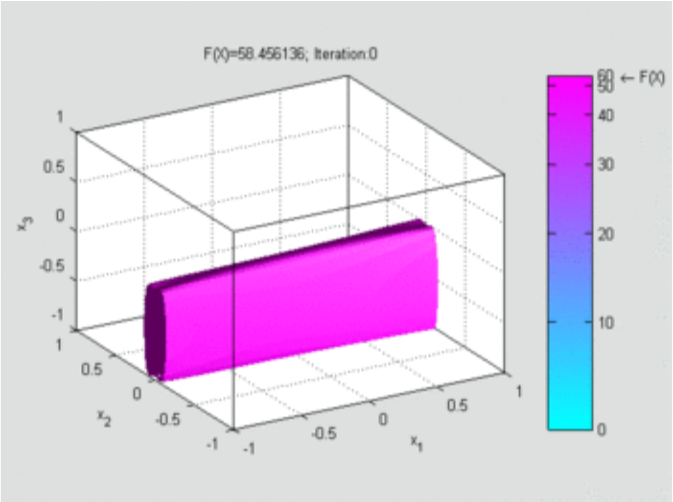
Evaluating at this value

$$\mathrm{F}(\underline{x}^{(1)}) = \frac{1}{2}((-2.48)^2 + (-1.00)^2 + (6.28)^2) = 23.306$$

The decrease from $F(\underline{x}^{(0)})$ =58.450

to the next steps value $F(\underline{x}^{(1)}) = 23.306$

is a sizable decrease in the objective function .Further steps would reduce its value until a solution to the system was found



An animation showing the first 83 iterations of gradient descent applied to this example. Surfaces are isosurfaces of $F\left(\underline{x}^{(n)}\right)$ at current guess $\underline{x}^{(n)}$, and arrows show the direction of descent. Due to a small and constant step size, the convergence is slow.

Multidimensional unconstrained optimization:

3- Analytical method:

Definition(4):

- If $f(x,y) < f(a,b), \forall\ (x,y) near\ (a,b), f(a,b)$

$is\ a\ local\ maximum.$

- If $f(x,y) > f(a,b), \forall\ (x,y) near\ (a,b), f(a,b)$

$is\ a\ local\ minimum.$

- If $f(x,y)$ has a local maximum or minimum at (a,b), and the first order partial derivatives of $f(x,y)$ exist at (a,b), , then
- $\frac{\partial f}{\partial x}\Big|_{(a,b)} = 0 \ \ , and\ \ \frac{\partial f}{\partial y}\Big|_{(a,b)} = 0$
- If $\frac{\partial f}{\partial x}\Big|_{(a,b)} = 0 \ \ , and\ \ \frac{\partial f}{\partial y}\Big|_{(a,b)} = 0$

Then (a,b), is a critical point or stationary point of $f(x,y)$

- If $\frac{\partial f}{\partial x}\Big|_{(a,b)} = 0 \ \ , and\ \ \frac{\partial f}{\partial y}\Big|_{(a,b)} = 0$

and the second order partial derivatives of $f(x,y)$ are continuous , then

- When $|H| > 0$ and $\frac{\partial^2 f}{\partial x^2}\Big|_{(a,b)} < 0$, $f(a,b)$ is a local maximum of $f(x,y)$.
- When $|H| > 0$ and $\frac{\partial^2 f}{\partial x^2}\Big|_{(a,b)} > 0$, $f(a,b)$ is a local maximum of $f(x,y)$.
- When $|H| < 0$, $f(a,b)$ is a saddle point.

Hessian of $f(x,y)$ $\qquad$ H = $\begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$

$|H| = \frac{\partial^2 f}{\partial x^2}.\frac{\partial^2 f}{\partial y^2} - \frac{\partial^2 f}{\partial x \partial y}.\frac{\partial^2 f}{\partial y \partial x}$

when $\frac{\partial^2 f}{\partial x \partial y}.$ is contiuous , $\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}.$

When $|H| > 0$, $\frac{\partial^2 f}{\partial x^2} . \frac{\partial^2 f}{\partial y^2} > 0$

Example(7):

$f(x,y)$ = -$2xy$ + $2x - x^2 - 2y^2$, find the optimum of $f(x,y)$

Solution : $\frac{\partial f}{\partial x}$ == $2y + 2 - 2x$, $\frac{\partial f}{\partial y} = 2x - 4\,y$

Let $\frac{\partial f}{\partial x}$ = **0** , -$2x$ +$2y$ =2 . **Let** $\frac{\partial f}{\partial y}$ = 0 , $2x - 4\,y$ = 0

Then x_* = **2** and y_* = 1 $\boldsymbol{i.e.}$ (2,1)is a critical point.

$\frac{\partial^2 f}{\partial x^2} = -2$, $\frac{\partial^2 f}{\partial y^2} = -4$, $\frac{\partial^2 f}{\partial x \partial y} = 2$ or $\frac{\partial^2 f}{\partial y \partial x} . = 2$

$|H| = \frac{\partial^2 f}{\partial x^2} . \frac{\partial^2 f}{\partial y^2} - \frac{\partial^2 f}{\partial x \partial y} . \frac{\partial^2 f}{\partial y \partial x}$ = (-2) . (- 4) - 2^2 = 4 > 0

$\frac{\partial^2 f}{\partial x^2} < 0$, $(x_*, y_*) = (2,1)$is alocal maximum.

Section(4)

A Matlab programs to the unconstrained optimization algorithms applied to some problems:

Example (1) :(Newton's method)

This example solves the system of two equations and two unknowns:

$$2x_1 - x_2 = e^{-x_1}$$
$$-x_1 + 2x_2 = e^{-x_2}.$$

Start your search for a solution at x_0 = [-5 -5].


```
x =
    0.5671
    0.5671
fval =
  1.0e-006 *
     -0.4059
     -0.4059
```

If the system of equations is linear, use\ (matrix left division) for better speed and accuracy. For example, to find the solution to the following linear system of equations:

$3x_1 + 11x_2 – 2x_3 = 7$
$x_1 + x_2 – 2x_3 = 4$
$x_1 – x_2 + x_3 = 19.$

Formulate and solve the problem as

```
A = [ 3 11 -2; 1 1 -2; 1 -1 1];
b = [ 7; 4; 19];
x = A\b
x =
   13.2188
   -2.3438
    3.4375
```

Example(2):(Implementation of the gradient -decent method)

Consider the problem of finding a solution to the following system of two nonlinear equations:

$g_1(x, y) = x^2 + y^2 - 1 = 0$, $g_2(x, y) = x^4 - y^4 + xy = 0.$

Newton's iteration scheme

$$(x_n+1, y_n+1)^T = (x_n, , y_n)^T – (Jg(, x_n, y_n))^{-1}(g_1(x_n y_n), (g_2(x_n y_n))^T$$

Requires the Jacobean $Jg(x, y)$. The components of Jg are :

$$(Jg)_{11} = 2x , (Jg)_{12} = 2y , (Jg)_{21} = 4x^3 + y , (Jg)_{22} = 4y^3 + x .$$

The iteration starts with some properly chosen starting value (x_0, y_0) . We stop the iteration if one of the following criteria is satisfied :

- The number of iterations exceeds an upper bound n_{max}
- The error $\varepsilon = g_1(x, y)| + |g_2(x, y)|$ is below a threshold ε_{max}

We choose $n_{max} = 100$ and $\varepsilon_{max} = 10^{-10}$. For the starting value $(x_0, y_0) = (1,1)$, the iteration is implemented in Matlab through the following commands (executed in a script file):

```
n= 0;

eps = 1;

x = [1;1];

while eps>1e -10 &n < 100

        g= [x (1)^2+x(2)^3-1;x(1)^4-x(2)^4+x(1)*x(2) ] ;

eps = abs (g (1) ) +abs (g (2) ) ;

Jg= [2*x (1) , 3*x (2) ^2; 4*x(2) ^3+x(1) ] ;

Y = x-Jg\ g ;

        x= y;

        n=n+1 ;

end

    n, x, eps,
```

The answer in the Matlab command window is :

```
 n  =

      6

x    =

    0.56497837204568

     0.87971040708669
```


eps =

1013964393477723e-013

Thus it took only 6 iterations to find a solution with an error below 10^{-13}. If the starting value is changed to $((x_0, y_0) = (-1,-1)$, the output is

n =

9

x =

-0.89443195117284

0.58479524791598

eps =

8. 626432901337466e – 014

giving another approximate solution .

x=y;

n=n+1;
end
n ,x ,eps,
Answer :
n = 85
x =
0.03349047167893
-0.56698094332619
eps =
8.552891728186296e-011

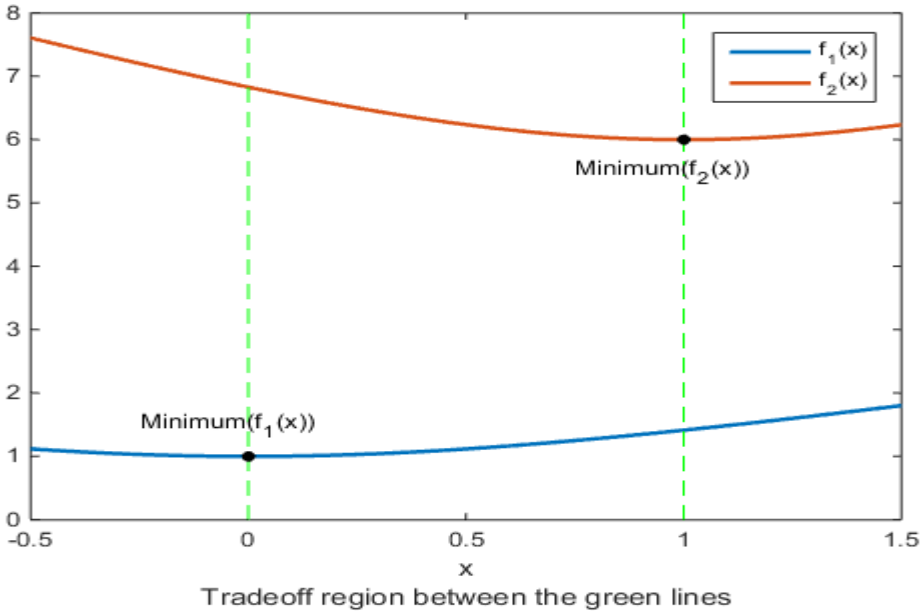
Example(3):

function f = simple_mult(x)

f(:,1) = sqrt(1+x.^2);
f(:,2) = 4 + 2*sqrt(1+(x-1).^2);

Both components are increasing as x decreases below 0 or increases above 1. In between 0 and 1, $f_1(x)$ is increasing and $f_2(x)$ is decreasing, so there is a tradeoff region.

```
t = linspace(-0.5,1.5);
F = simple_mult(t);
plot(t,F,'LineWidth',2)
hold on
plot([0,0],[0,8],'g--');
plot([1,1],[0,8],'g--');
plot([0,1],[1,6],'k.','MarkerSize',15);
text(-0.25,1.5,'Minimum(f_1(x))')
text(.75,5.5,'Minimum(f_2(x))')
hold off
legend('f_1(x)','f_2(x)')
xlabel({'x';'Tradeoff region between the green lines'})
```



Appendix

Difficulty in calculating derivative of a function:

Newton's method requires that the derivative be calculated directly. An analytical expression for the derivative may not be easily obtainable and could be expensive to evaluate. In these situations, it may be appropriate to

approximate the derivative by using the slope of a line through two nearby points on the function. Using this approximation would result in something like the secant method whose convergence is slower than that of Newton's method.

Failure of the method to converge to the root:

It is important to review the proof of quadratic convergence of Newton's Method [2] before implementing it. Specifically, one should review the assumptions made in the proof. For situations where the method fails to converge, it is because the assumptions made in this proof are not met.

References:

[1] Bonnans, J. Frédéric; Gilbert, J. Charles; Lemaréchal, Claude; Sagastizábal, Claudia A. (2006). Numerical optimization: Theoretical and practical aspects. Universitext (Second revised ed. of translation of 1997 French ed.). Berlin: Springer-Verlag. pp. xiv+490. doi:10.1007/978-3-540-35447-5. ISBN 3-540-35445-X. MR2265882.

[2] C. T. Kelley, Solving Nonlinear Equations with Newton's Method, no 1 in Fundamentals of Algorithms, SIAM, 2003. ISBN 0-89871-546-6.

[3] Endre Süli and David Mayers, An Introduction to Numerical Analysis, Cambridge University Press, 2003. ISBN 0-521-00794-1

[4] Kaw, Autar; Kalu, Egwu (2008). Numerical Methods with Applications (1st ed.)

[5] M. A. EL siddieg Awatif(2015)The Davidon Fletcher Powel Method Tested on Quadaratic Programming functions

[6] P. Deuflhard, Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms. Springer Series in Computational Mathematics, Vol. 35. Springer, Berlin, 2004. ISBN 3-540-21099-7.

[7] Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Chapter 9. Root Finding and Nonlinear Sets of Equations Importance Sampling". Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.. See especially Sections 9.4, 9.6, and 9.7.

[8] Tjalling J. Ypma, Historical development of the Newton-Raphson method, SIAM Review 37 (4), 531–551, 1995. doi:10.1137/1037125

[9] J. M. Ortega, W. C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables. Classics in Applied Mathematics, SIAM, 2000. ISBN 0-89871-461-3.