1 Multidimensional root-finding

1.1 Introduction

In this part, we consider the problems of solving a system of n equations in n unknowns,

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0, \\ f_1(x_1, x_2, \ldots, x_n) &= 0, \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0. \end{aligned}$$

For simplicity, we express this system of nonlinear equations in vector form,

$$\mathbf{F}(\mathbf{x}) = \mathbf{0},$$

where $\mathbf{F}\colon \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}^n$ is a vector-valued function of n variables presented by vectors

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad \mathbf{F} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

The component functions f_1, f_2, …, f_n are functions of x_1, x_2, …, x_n in general.

1.2 Limit, Continuity and Differentiability in $\mathbb{R}^n$

Recall the notion of limit can be easily generalized to vector-valued functions and functions of several variables. Given a function $f\colon \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}$ and a point $\mathbf{x}_0 \in \mathcal{D}$, we write

$$\lim_{\mathbf{x}\to\mathbf{x}_0} f(\mathbf{x}) = L$$

if, for any $\epsilon > 0$, there exists a $\delta > 0$ such that

$$|f(\mathbf{x}) - L| < \epsilon$$

whenever $\mathbf{x} \in \mathcal{D}$ and

$$0 \le \|\mathbf{x} - \mathbf{x}_0\| \le \delta.$$

where $\|\cdot\|$ denotes any appropriate norm in $\mathbb{R}^n$. And f is said to be continuous at $\mathbf{x}_0 \in \mathcal{D}$ if

$$\lim_{\mathbf{x}\to\mathbf{x}_0} f(\mathbf{x}) = f(\mathbf{x}_0)$$

Furthermore, f is said to be differentiable at $\mathbf{x}_0 \in \mathcal{D}$ if there is a constant vector $\mathbf{a}$ such that

$$\lim_{\mathbf{x}\to\mathbf{x}_0} \frac{\left|f(\mathbf{x}) - f(\mathbf{x}_0) - \mathbf{a}^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_0)\right|}{\|\mathbf{x} - \mathbf{x}_0\|} = 0$$

Similarly, for $\mathbf{F}\colon \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{x}_0 \in \mathcal{D}$, we write

$$\lim_{\mathbf{x}\to\mathbf{x}_0} \mathbf{F}(\mathbf{x}) = \mathbf{L}$$

if and only if

$$\lim_{\mathbf{x}\to\mathbf{x}_0} f_i(\mathbf{x}) = L_i \qquad \text{for all} \quad i = 1, 2, \ldots, n.$$

And $\mathbf{F}$ is said to be continuous at $\mathbf{x}_0 \in \mathcal{D}$ if

$$\lim_{\mathbf{x}\to\mathbf{x}_0} \mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x}_0)$$

Furthermore, $\mathbf{F}$ is said to be differentiable at $\mathbf{x}_0$ if there exists a linear transformation

$$\mathbf{T}\colon \mathbb{R}^n \to \mathbb{R}^n$$

such that

$$\lim_{\mathbf{x}\to\mathbf{x}_0} \frac{\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}_0) - \mathbf{T}(\mathbf{x}-\mathbf{x}_0)\|}{\|\mathbf{x}-\mathbf{x}_0\|} = 0$$

where the matrix associated with the linear transformation $\mathbf{T}$ is the matrix of partial derivatives, namely, the Jacobian matrix $\mathbf{J}$.

1.3 Fixed-point iteration

Now suppose we have the following function

$$\mathbf{F}(\mathbf{x}) = \mathbf{G}(\mathbf{x}) - \mathbf{x} \qquad \text{where} \qquad \mathbf{G}\colon \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}^n$$

In class, we have discuss why fixed-point iteration can be used to find the root of a single function of the form

$$f(x) = g(x) - x$$

where g is Lipschitz continuous with Lipschitz constant $c \in [0, 1)$. In the following assignment, we have learned that if the function g instead is continuously differentiable and maps an interval $\mathcal{I}$ into itself, then g has a fixed point in $\mathcal{I}$. Furthermore, if there is a constant $c < 1$ such that

$$|g'(x)| < c \qquad \text{for all } x \in \mathcal{I}.$$

then the fixed-point iteration will converge to the fixed point. The existence and uniqueness of fixed points of vector-valued functions of several variables and the convergence result are in line with the single-variable case. Below are the analogues of the two theorems in higher dimensions.

Let $\mathcal{D} \subseteq \mathbb{R}^n$ be closed and $\mathbf{G}\colon \mathcal{D} \to \mathcal{D}$ be Lipschitz continuous, that is, there is a constant c

$$\|\mathbf{G}(\mathbf{x}_1) - \mathbf{G}(\mathbf{x}_2)\| \le c\,\|\mathbf{x}_1 - \mathbf{x}_2\| \qquad \text{for all } \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}.$$

If this c, which is known as a Lipschitz constant, is

$$0 \le c < 1$$

then the function $\mathbf{G}$ has a unique fixed point $\mathbf{x}^* \in \mathcal{D}$ and the sequence defined by

$$\mathbf{x}_0 \in \mathcal{D} \qquad \text{and} \qquad \mathbf{x}_k = \mathbf{G}(\mathbf{x}_{k-1}) \qquad \text{for} \quad k = 1, 2, 3, \ldots$$

converges to $\mathbf{x}^*$

Let $\mathcal{D} \subseteq \mathbb{R}^n$ be closed and $\mathbf{G}\colon \mathcal{D} \to \mathcal{D}$ be continuously differentiable. Furthermore, if there exists a constant $c < 1$ and a matrix norm $\|\cdot\|$ such that

$$\|\mathbf{J}_{\mathbf{G}}(\mathbf{x})\| \le c \qquad \text{for all } \mathbf{x} \in \mathcal{D}.$$

then $\mathbf{G}$ has a unique fixed point $\mathbf{x}^*$ in $\mathcal{D}$, and fixed-point iteration is guaranteed to converge to $\mathbf{x}^*$ for any initial point $\mathbf{x}_0 \in \mathcal{D}$.

1.4 Newton's method

We have learned in class that Newton's method can achieve quadratic convergence. Recall that this method uses the following iteration formula:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad k = 0, 1, 2, \dots \qquad \text{where } x_0 \text{ is an initial point.}$$

Here we present Newton's method to system of nonlinear equations. Let us define

$$\mathbf{G}(\mathbf{x}) = \mathbf{x} - [\mathbf{J}_{\mathbf{F}}(\mathbf{x})]^{-1}\,\mathbf{F}(\mathbf{x})$$

where $\mathbf{J}_{\mathbf{F}}(\mathbf{x})$ is the Jacobian matrix of $\mathbf{F}$ evaluated at $\mathbf{x}$. Similar to what we did in class for the single-variable case. In order to show quadratic convergence, all we need is to show the derivative, here the Jacobian of $\mathbf{G}$, is zero at the root $\mathbf{x} = \mathbf{x}^*$ given $\mathbf{F}$ has an appropriate level of smoothness. If we define

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) \end{bmatrix} \quad \text{and} \quad \mathbf{G}(\mathbf{x}) = \begin{bmatrix} g_1(x_1, x_2, \dots, x_n) \\ g_2(x_1, x_2, \dots, x_n) \\ \vdots \\ g_n(x_1, x_2, \dots, x_n) \end{bmatrix}$$

where f_i and g_i are the component functions of $\mathbf{F}$ and $\mathbf{G}$, then we have

$$\begin{aligned} \frac{\partial g_i}{\partial x_k} &= \frac{\partial}{\partial x_k}\left[x_i - \sum_{j=1}^{n} a_{ij}(\mathbf{x}) f_j(\mathbf{x})\right] \qquad \text{where } a_{ij} \text{ is the } ij\text{-th element of } [\mathbf{J}_{\mathbf{F}}(\mathbf{x})]^{-1} \\ &= \delta_{ik} - \sum_{j=1}^{n} a_{ij}(\mathbf{x}) \frac{\partial f_j}{\partial x_k} - \sum_{j=1}^{n} f_j(\mathbf{x}) \frac{\partial a_{ij}}{\partial x_k} \end{aligned}$$

Note the first summation is equal to δ_{ik} since $\dfrac{\partial f_j}{\partial x_k}$ is the jk-th element of $\mathbf{J}_{\mathbf{F}}(\mathbf{x})$ by definition, thus the summation is equal to ik-th element of

$$[\mathbf{J}_{\mathbf{F}}(\mathbf{x})]^{-1}\,\mathbf{J}_{\mathbf{F}}(\mathbf{x}) = \mathbf{I}$$

The second summation vanishes at $\mathbf{x}^*$ since $\mathbf{F}(\mathbf{x}^*) = \mathbf{0}$. Hence we reach the conclusion

$$\begin{aligned} \nabla g_i(\mathbf{x}^*) &= \mathbf{0} \qquad \text{for all } i = 1, 2, \dots, n \\ \implies \mathbf{J}_{\mathbf{G}}(\mathbf{x}^*) &= \mathbf{0} \end{aligned}$$

Since $\mathbf{J}_{\mathbf{G}}$ is the zero matrix at $\mathbf{x}^*$, and $\mathbf{G}$ is continuously differentiable at $\mathbf{x}^*$ for a sufficiently smooth $\mathbf{F}$ that has an invertible $\mathbf{J}_{\mathbf{F}}(\mathbf{x}^*)$, there is a constant $c < 1$ and a norm $\|\cdot\|$ such that

$$\|\mathbf{J}_{\mathbf{G}}(\mathbf{x})\| \le c \qquad \text{for all } \mathbf{x} \in \mathcal{D}.$$

This shows that Newton's method in this case can have quadratic convergence near root $\mathbf{x}^*$. To apply a multidimensional version of Newton's method, we choose an initial point $\mathbf{x}_0$ in the hope of it being sufficiently close to $\mathbf{x}^*$. Then we iterate as follows:

$$\begin{aligned} \mathbf{y}_k &= -\mathbf{F}(\mathbf{x}_k) \\ \mathbf{w}_k &= \left[\mathbf{J}_{\mathbf{F}}(\mathbf{x}_k)\right]^{-1} \mathbf{y}_k \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \mathbf{w}_k \end{aligned}$$

where the vector $\mathbf{w}_k$ is computed by solving the system

$$\mathbf{J}_{\mathbf{F}}(\mathbf{x}_k)\mathbf{w}_k = \mathbf{y}_k$$

using a method such as Gaussian elimination with back substitution. The above scheme can be applied to a function $\mathbf{F}\colon \mathbb{R}^n \to \mathbb{R}^m$ with only a slight modification to how we solve $\mathbf{w}_k$. When

$$m < n,$$

we use the pseudoinverse, by which we can find one of potentially many roots of $\mathbf{F}$. When

$$m > n,$$

we can try the least-square solution, but convergence of this scheme become doubtful.

1.5 Quasi-Newton

When n becomes large, Newton's method becomes very expensive due the computation of the Jacobian matrix $\mathbf{J}_{\mathbf{F}}(\mathbf{x}_k)$ in each iteration, and the computation of $\mathbf{w}_k$ by solving

$$\mathbf{J}_{\mathbf{F}}(\mathbf{x}_k)\mathbf{w}_k = \mathbf{y}_k$$

Furthermore, it is not possible to take information from one iteration to the next since the Jacobian matrix is changing every iteration. An alternative is to modify Newton's method by using approximate derivatives as in the secant method for a single nonlinear equation. The secant method uses the approximation

$$f'(x_1) \approx \underbrace{\frac{f(x_1) - f(x_0)}{x_1 - x_0}}_{A_1}$$

as a replacement for $f'(x_1)$ in the single-variable Newton's method in the first iteration. Note the approximation A_1 is a scalar such that

$$A_1(x_1 - x_0) = f(x_1) - f(x_0)$$

Along this line of thinking, we replace the Jacobian matrix $\mathbf{J}_{\mathbf{F}}(\mathbf{x}_1)$ by a matrix $\mathbf{A}_1$ that satisfies

$$\mathbf{A}_1\left(\mathbf{x}_1 - \mathbf{x}_0\right) = \mathbf{F}\left(\mathbf{x}_1\right) - \mathbf{F}\left(\mathbf{x}_0\right)$$

Any nonzero vector in $\mathbb{R}^n$ can be written as a sum of a scalar multiple of $\mathbf{x}_1 - \mathbf{x}_0$ and a scalar multiple of a vector in the *orthogonal complement* of $\mathbf{x}_1 - \mathbf{x}_0$. So to uniquely define the matrix $\mathbf{A}_1$, we also need to specify how $\mathbf{A}_1$ maps the orthogonal complement of $\mathbf{x}_1 - \mathbf{x}_0$. However, no information is available about the change in $\mathbf{F}$ in a direction orthogonal to $\mathbf{x}_1 - \mathbf{x}_0$. So we specify there is no update in this direction, that is,

$$\mathbf{A}_1\mathbf{z} = \mathbf{J}_{\mathbf{F}}(\mathbf{x}_0)\mathbf{z} \qquad \text{whenever} \qquad \left(\mathbf{x}_1 - \mathbf{x}_0\right)^{\mathrm{T}}\mathbf{z} = 0$$

It is clear that the following matrix satisfies both conditions:

$$\mathbf{A}_1 = \mathbf{J}_\mathbf{F}(\mathbf{x}_0) + \frac{\Big[\mathbf{F}(\mathbf{x}_1) - \mathbf{F}(\mathbf{x}_0) - \mathbf{J}_\mathbf{F}(\mathbf{x}_0)\,(\mathbf{x}_1 - \mathbf{x}_0)\Big]\,(\mathbf{x}_1 - \mathbf{x}_0)^\mathrm{T}}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2^2}$$

So we use this $\mathbf{A}_1$ in place of $\mathbf{J}_\mathbf{F}(\mathbf{x}_1)$ to determine $\mathbf{x}_2$, and then $\mathbf{A}_2$ to $\mathbf{x}_3$, and etc. In general

$$\begin{aligned}
\mathbf{s}_k &= \mathbf{x}_k - \mathbf{x}_{k-1}\\
\mathbf{y}_k &= \mathbf{F}(\mathbf{x}_k) - \mathbf{F}(\mathbf{x}_{k-1})\\
\mathbf{A}_k &= \mathbf{A}_{k-1} + \frac{\mathbf{y}_k - \mathbf{A}_{k-1}\mathbf{s}_k}{\|\mathbf{s}_k\|_2^2}\mathbf{s}_k^\mathrm{T}\\
\mathbf{w}_k &= \mathbf{A}_k^{-1}\mathbf{y}_k\\
\mathbf{x}_{k+1} &= \mathbf{x}_k - \mathbf{w}_k
\end{aligned}$$

However, simply approximating $\mathbf{J}_\mathbf{F}(\mathbf{x}_k)$ by $\mathbf{A}_k$ does not reduce the cost by much, because we still have to incur the same cost of solving the linear system

$$\mathbf{A}_k\mathbf{w}_k = \mathbf{y}_k$$

So to truly cut the cost, we have to find a way to efficiently update the inverse of $\mathbf{A}_k$ from previous available information. This can be done by employing a matrix inversion formula known as the Sherman and Morrison' formula:

$$\left(\mathbf{B} + \mathbf{a}\mathbf{b}^\mathrm{T}\right)^{-1} = \mathbf{B}^{-1} - \frac{\mathbf{B}^{-1}\mathbf{a}\mathbf{b}^\mathrm{T}\mathbf{B}^{-1}}{1 + \mathbf{b}^\mathrm{T}\mathbf{A}^{-1}\mathbf{a}}$$

where $\mathbf{B}$ is invertible and $1 + \mathbf{b}^\mathrm{T}\mathbf{A}^{-1}\mathbf{a} \neq 0$. So by letting

$$\mathbf{B} = \mathbf{A}_{k-1}, \qquad \mathbf{a} = \frac{\mathbf{y}_k - \mathbf{A}_{k-1}\mathbf{s}_k}{\|\mathbf{s}_k\|_2^2,} \qquad \text{and} \qquad \mathbf{b} = \mathbf{s}$$

the Sherman and Morrison's formula gives the following formula for the inverse

$$\mathbf{A}_k^{-1} = \mathbf{A}_{k-1}^{-1} + \frac{\left(\mathbf{s}_k - \mathbf{A}_{k-1}^{-1}\mathbf{y}_k\right)\mathbf{s}_k^\mathrm{T}\mathbf{A}_{k-1}^{-1}}{\mathbf{s}_k^\mathrm{T}\mathbf{A}_{k-1}^{-1}\mathbf{y}_k}$$

Combining the above approximation for the Jacobian matrix and this inversion formula gives what is known as *Broyden's Method.* Note that it is not necessary to compute the approximation for the Jacobian matrix, namely, $\mathbf{A}_k$ for $k \geq 1$; $\mathbf{A}_k^{-1}$ is computed directly.

2 Optimization

2.1 Introduction

In this part, we consider the general problem of finding the minimum for

$$f\colon \mathbb{R}^n \to \mathbb{R}$$

with/without some constraints:

$$g(\mathbf{x}) = 0 \qquad \text{and/or} \qquad h(\mathbf{x}) \geq 0$$

where $\mathbf{x} \in \mathbb{R}^n$.

2.2 Nonsmoothness

If the function f is sufficiently smooth and convex in the absence of any constraint, we could turn the optimisation problem into a root-finding problem by finding $\mathbf{x}^*$ such that

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

So our discussion on Newton's method and Quasi-Newton's method are applicable with little or no modification. In fact, finding a minimum of a function of many variables is easier in comparison with multidimensional root-finding. This is because the component functions of an n-dimensional gradient vector are not completely arbitrary functions. Rather, they obey so-called integrability conditions. So those methods are usually fast for well-behaved functions, however, they are still not very robust. That is, if the function is not sufficiently smooth or the initial guess is not sufficiently close to the minimum, then they may converge very slowly or even diverge. The downhill simplex method is an alternative that makes no assumptions about f and requires no derivatives. It is less efficient but it is very easy to understand and implement. We are to cover it here so that we have an optional method for functions that are too problematic for Newton and Quasi-Newton to handle. In many cases, the downhill simplex is faster than simulated annealing but usually slower than Newton and Quasi-Newton. It is based on the geometric object known as a simplex. To define a simplex properly, consider the following definitions:
A *linear combination* of $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$

$$\sum_{i=1}^{n} \alpha_i \mathbf{x}_i = \alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \cdots + \alpha_n \mathbf{x}_n$$

is known as a convex combination if all coefficients sum to 1 and are non-negative, that is,

$$\sum_{i=1}^{n} \alpha_i = 1 \qquad \text{and} \qquad \alpha_i \geq 0 \qquad \text{for all } i = 1, \ldots, n$$

The convex hull of a set $\mathcal{S} \in \mathbb{R}^n$ is the set of all convex combinations of all points $\mathbf{x}_i \in \mathcal{S}$.
A simplex is a geometric object in $\mathbb{R}^n$ that represents the convex hull of $n+1$ points in $\mathbb{R}^n$, which are known as vertices of the simplex. So a simplex in $\mathbb{R}$ is a line segment, a simplex in $\mathbb{R}^2$ is a triangle, a simplex in $\mathbb{R}^3$ is a tetrahedron, and etc. The downhill simplex method iteratively generates a a sequence of smaller and smaller simplexes in $\mathbb{R}^n$ to enclose the minimum point $\mathbf{x}^*$. At each iteration, the vertices $\{\mathbf{x}_j\}_{j=1}^{n+1}$ of the simplex are ordered according their $f(\mathbf{x})$ values

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots f(\mathbf{x}_{n+1}).$$

The point $\mathbf{x}_1$ is referred as the *best* vertex, and $\mathbf{x}_{n+1}$ as the *worst* vertex. The downhill simplex method uses 4 possible operations to iterate from one simplex to the next:

reflection	*expansion*	*contraction*	*shrink*
$\alpha > 0$	$\beta > 1$	$0 < \gamma < 1$	$0 < \delta < 1$

Each operation is associated with a scalar parameter, and the centroid of n vertices other than the worst vertex

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}$$

The downhill simplex method iterates by the following scheme:

1. *Sort*:

   Evaluate f at all $n+1$ vertices of the current simplex and sort the vertices so that

   $$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots f(\mathbf{x}_{n+1})$$

2. *Reflection*:

   Compute the reflection point

   $$\mathbf{x}_r = \bar{\mathbf{x}} + \alpha(\bar{\mathbf{x}} - \mathbf{x}_{n+1})$$

   #Since $\mathbf{x}_{n+1}$ is the worst vertex, we want moving away from it. This so-called reflection point is just a test point along the line defined by $\bar{\mathbf{x}}$ and $\mathbf{x}_{n+1}$ away from the point $\mathbf{x}_{n+1}$.

3. *Conditional Execution*:

   If $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) < f(\mathbf{x}_n)$, then replace the worst vertex $\mathbf{x}_{n+1}$ with $\mathbf{x}_r$. *Next Iteration.*

   If $f(\mathbf{x}_r) < f(\mathbf{x}_1)$, then do the *Expansion* step.

   If $f(\mathbf{x}_n) \leq f(\mathbf{x}_r) < f(\mathbf{x}_{n+1})$, then jump to the *Outside Contraction* step.

   If $f(\mathbf{x}_{n+1}) \leq f(\mathbf{x}_r)$, then jump to the *Inside Contraction* step.

4. *Expansion*:

   Compute the expansion point

   $$\mathbf{x}_e = \bar{\mathbf{x}} + \beta(\mathbf{x}_r - \bar{\mathbf{x}})$$

   # If moving along the line defined by $\bar{\mathbf{x}}$ and $\mathbf{x}_{n+1}$ away from the point $\mathbf{x}_{n+1}$ gives a new best vertex, perhaps the minimum is just farther away in that region. So we move further into this region by pushing further along this direction.

5. *Conditional Execution*:

   If $f(\mathbf{x}_e) < f(\mathbf{x}_r)$, then replace the worst vertex $\mathbf{x}_{n+1}$ with $\mathbf{x}_e$; else replace $\mathbf{x}_{n+1}$ with $\mathbf{x}_r$; *Next Iteration.*

6. *Outside Contraction*:

   Compute the outside contraction point

   $$\mathbf{x}_{oc} = \bar{\mathbf{x}} + \gamma(\mathbf{x}_r - \bar{\mathbf{x}})$$

   # If the test point $\mathbf{x}_r$ is almost as bad as the worst vertex, then we contract along this direction to see whether we have pushed too far in this direction.

7. *Conditional Execution*:

   If $f(\mathbf{x}_{oc}) \leq f(\mathbf{x}_r)$, then replace the worst vertex $\mathbf{x}_{n+1}$ with $\mathbf{x}_{oc}$; *Next Iteration.*

   Else do the *Shrink* step.

8. *Insider Contraction*:

   Compute the inside contraction point

   $$\mathbf{x}_{ic} = \bar{\mathbf{x}} - \gamma(\mathbf{x}_r - \bar{\mathbf{x}})$$

   # If the test point $\mathbf{x}_r$ is even worse then the worst vertex, then we contract further along this direction to be on the same side as $\mathbf{x}_{n+1}$.

9. *Conditional Execution*:

   If $f(\mathbf{x}_{ic}) < f(\mathbf{x}_{n+1})$, then replace the worst vertex $\mathbf{x}_{n+1}$ with $\mathbf{x}_{ic}$; *Next Iteration.*

   Else do the *Shrink* step.

10. *Shrink*:

    Shrink the simplex toward the best vertex. Assign, for $2 \le j \le n+1$,

$$\mathbf{x}_j = \mathbf{x}_1 + \delta(\mathbf{x}_j - \mathbf{x}_1)$$

    Next Iteration.

    # If moving away from the worst vertex $\mathbf{x}_{n+1}$ alone is not useful, we have to consider moving towards the best vertex first before moving away from the worst vertex again. So this is a back and forth process like the movement of an ameba.

Even when f is sufficiently smooth and the gradient/Hessian is not expensive to evaluate , the downhill simplex method still provides a good initial guidance before Newton's method and Quasi-Newton's method are employed to refine the solution once we are sufficiently close.

3 Task

The followings are the actual tasks for this project:

Task 1 (5 points)
Consider the following system

$$x_1^2 - x_2 = 0; \qquad x_1^2 + x_2^2 = 1$$

(a) (1 point) Write a Matlab program to implement the fixed-point iteration.
(b) (1 point) Write a Matlab program to implement Newton's method.
(c) (1 point) Write a Matlab program to implement Broyden's method.
(d) (2 points) Which is the best method for the above system?

Task 2 (10 points)
Investigate the following system

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

then propose a scheme for solving this system. Write a report, 5-10 pages, including:

- Title Page
- Abstract
- Table of Contents
- Introduction
- Analysis
- Results and Discussion
- Conclusions
- References
- Appendices

Task 3 (5 points)
Let the following function denote some physical relationship between the variables y and x

$$y = f(x) = \alpha \exp\left(\frac{\beta}{x+\gamma}\right)$$

where α, β and γ are some unknown parameters $\in \mathbb{R}$. Suppose the following data are obtained from an experiment, and we want to find the best estimate for α, β and γ by following the principle of least squares.

x	y	x	y
50	34780	90	8261
55	28610	95	7030
60	23650	100	6005
65	19630	105	5147
70	16370	110	4427
75	13720	115	3820
80	11540	120	3307
85	9744	125	2872

Investigate the corresponding optimisation problem by trying various methods that we have discussed. Then propose a scheme for solving this problem. Submit only:

1. The pseudocode for your algorithm.
2. Some explanatory notes for your algorithm.
3. The solution to the problem.

[Hint: You need to do more than just combining methods to provide so-called *safeguards*.]

Task 4 (10 points)
An *academic summary* can be an invaluable resource for yourself later on. So the last task is to write one for the following paper

— Kuhn, H.W. and Tucker A.W. Nonlinear programming. In J. Neyman (ed.), *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, 1951.

and some sections of the following book

— Bertsekas, Dimitri P. Constrained optimization and Lagrange multiplier methods. Academic Press, 2014.

The topic of this *academic summary* is on

"Constrained Optimisation".

You shall cover 3-5 common algorithms with short explanatory notes as well as some necessary and sufficient conditions for a minimum. You shall aim for quality over quantity.

Click the box to download the paper : A version provided by Project Euclid

Click the box to download the book : An old version provided by the author