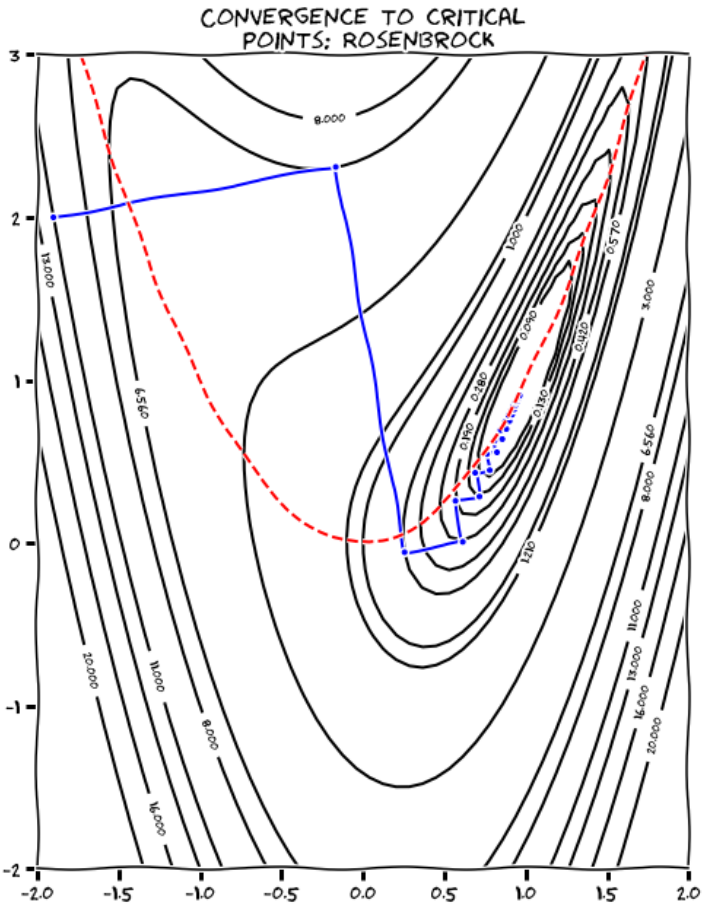
Course notes for MATH 524: Non-Linear Optimization

Francisco Blanco-Silva



Department of Mathematics, University of South Carolina
E-mail address: blanco@math.sc.edu
URL: people.math.sc.edu/blanco

2010 *Mathematics Subject Classification.* 49M, 65K, 90C

This version of the notes was completed on November 12, 2017.

Contents

List of Figures v

Chapter 1. Review of Optimization from Vector Calculus 1
The Theory of Optimization 6
Exercises 7

Chapter 2. Existence and Characterization of Extrema for Unconstrained Optimization 11
1. Anatomy of a function 11
2. Existence results 21
3. Characterization results 22
Examples 22
Exercises 24

Chapter 3. Numerical Approximation for Unconstrained Optimization 29
1. Newton-Raphson's Method 29
2. Secant Methods 39
3. The Method of Steepest Descent 44
4. Effective Algorithms for Unconstrained Optimization 51
Exercises 54

Chapter 4. Existence and Characterization of Extrema for Constrained Optimization 63
1. Necessary Conditions 66
2. Sufficient Conditions 68
Key Examples 69
Exercises 70

Chapter 5. Numerical Approximation for Constrained Optimization 73
1. Projection Methods for Linear Equality constrained programs 73
2. Linear Programing: The simplex method 78
3. The Frank-Wolfe Method 86
Exercises 89

Index 91

Bibliography 93

Appendix A. Rates of Convergence 95

Appendix B. Basic `sympy` commands for Calculus 97
1. Function operations 97
2. Derivatives, Gradients, Hessians 98
3. Integration 99
4. Sequences, series 100
5. Power series, series expansions 100

Appendix C. Basic graphing in Python 101
1. `matplotlib` 101

List of Figures

1.1 Details of the graph of $\mathcal{R}_{1,1}$ 3
1.2 Global minima in unbounded domains 5
1.3 Contour plots for problem 1.4 8

2.1 Detail of the graph of $\mathcal{W}_{0.5,7}$ 13
2.2 Convex sets. 18
2.3 Convex Functions. 19

3.1 Newton-Raphson iterative method 30
3.2 Initial guess must be carefully chosen in Newton-Raphson 31
3.3 Newton-Raphson fails for some functions 32
3.4 Newton-Raphson method 37
3.5 Secant iterative method 40
3.6 The Method of Steepest Descent: Polynomial function 47
3.7 The Method of Steepest Descent: Rosenbrock Function 48
3.8 The BFGS method: Rosenbrock function 55
3.9 Newton method in `desmos.com` 57

4.1 Can you tell what are the global maximum and minimum values of f in S? 64
4.2 Cones for $(0,0)$, $(-1,-1)$ and $(0,-1/2)$. 65
4.3 Feasibility region for (P) in example 4.4 68

5.1 Illustration of the simplex method for Example 5.4 81
5.2 Set up for Example 5.9 88
5.3 Frank-Wolfe iteration to solve the program (P) in Example 5.9. 89

C.1 Basic rendering of functions with `pyplot` 103
C.2 Tinkering with color, style and width of lines in `pyplot` 103
C.3 Contour plots with `pyplot` 104

CHAPTER 1

Review of Optimization from Vector Calculus

The starting point of these notes is the concept of *optimization* as developed in MATH 241 (see e.g. [**6**, Chapter 14])

DEFINITION. If $f(x, y)$ is differentiable in an open region containing the point (x_0, y_0), we define the *gradient vector* of $f(x, y)$ at (x_0, y_0) as the vector

$$\nabla f(x_0, y_0) = \left[\frac{\partial f(x_0, y_0)}{\partial x}, \frac{\partial f(x_0, y_0)}{\partial y}\right].$$

Given any vector $\boldsymbol{v} = [v_1, v_2]$ with $\|\boldsymbol{v}\| = \left(v_1^2 + v_2^2\right)^{1/2} = 1$ (what we call a *unit vector* or a *direction*), we define the *directional derivative* of f in the direction $\boldsymbol{v}$ at (x_0, y_0) by

$$D_{\boldsymbol{v}} f(x_0, y_0) = \langle \nabla f(x_0, y_0), \boldsymbol{v} \rangle = v_1 \frac{\partial f(x_0, y_0)}{\partial x} + v_2 \frac{\partial f(x_0, y_0)}{\partial y}.$$

REMARK 1.1. The gradient has many interesting properties. Assume $f(x, y)$ is a differentiable function.

Fastest Increase: At any point (x, y), the function f increases most rapidly in the direction of the gradient vector $\boldsymbol{v} = \nabla f(x, y)$. The derivative in that direction is $D_{\boldsymbol{v}} f(x, y) = \|\nabla f(x, y)\|$.

Fastest Decrease: At any point (x, y), the function f decreases most rapidly in the direction $\boldsymbol{v} = -\nabla f(x, y)$. The derivative in that direction is $D_{\boldsymbol{v}} f(x, y) = -\|\nabla f(x, y)\|$.

Zero Change: Any direction $\boldsymbol{v}$ perpendicular to a non-zero gradient is a direction of *zero change* in f at (x, y): $D_{\boldsymbol{v}} f(x, y) = 0$.

Tangents to Level Curves: At every point (x, y) in the domain of f, the gradient $\nabla f(x, y)$ is perpendicular to the level curve through (x, y).

DEFINITION. Let $D \subseteq \mathbb{R}^2$ be a region on the plane containing the point (x_0, y_0). We say that the real-valued function $f\colon D \to \mathbb{R}$ has a *local minimum* at (x_0, y_0) if $f(x_0, y_0) \leq f(x, y)$ for all domain points (x, y) in an open disk centered at (x_0, y_0). In that case, we also say that $f(x_0, y_0)$ is a *local minimum value* of f in D.

Emphasis was made to find conditions on the function f to guarantee existence and characterization of minima:

THEOREM 1.1. *Let $D \subseteq \mathbb{R}^2$ and let $f\colon D \to \mathbb{R}$ be a function for which first partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ exist in D. If $(x_0, y_0) \in D$ is a local minimum of f, then $\nabla f(x_0, y_0) = 0$.*

The local minima of these functions are among the zeros of the equation $\nabla f(x, y) = 0$, the so-called *critical points* of f. More formally:

DEFINITION. An interior point of the domain of a function $f(x, y)$ where both directional derivatives are zero, or where at least one of the directional derivatives do not exist, is a *critical point* of f.

We employed the *Second Derivative Test for Local Extreme Values* to characterize some minima:

THEOREM 1.2. *Suppose that $f\colon \mathbb{R}^2 \to \mathbb{R}$ and its first and second partial derivatives are continuous throughout a disk centered at the point (x_0, y_0), and that $\nabla f(x_0, y_0) = 0$. If the two following conditions are satisfied, then $f(x_0, y_0)$ is a local minimum value:*

$$\frac{\partial^2 f(x_0, y_0)}{\partial x^2} > 0 \tag{1}$$

$$\det \underbrace{\begin{bmatrix} \dfrac{\partial^2 f(x_0, y_0)}{\partial x^2} & \dfrac{\partial^2 f(x_0, y_0)}{\partial x \partial y} \\ \dfrac{\partial^2 f(x_0, y_0)}{\partial y \partial x} & \dfrac{\partial^2 f(x_0, y_0)}{\partial y^2} \end{bmatrix}}_{\operatorname{Hess} f(x_0, y_0)} > 0 \tag{2}$$

REMARK 1.2. The restriction of this result to univariate functions is even simpler: Suppose f'' is continuous on an open interval that contains x_0. If $f'(x_0) = 0$ and $f''(x_0) > 0$, then f has a local minimum at x_0.

EXAMPLE 1.1 (Rosenbrock Functions). Given strictly positive parameters $a, b > 0$, consider the Rosenbrock function

$$\mathcal{R}_{a,b}(x, y) = (a - x)^2 + b(y - x^2)^2.$$

It is easy to see that Rosenbrock functions are polynomials (prove it!). The domain is therefore the whole plane. Figure 1.1 illustrates a contour plot with several level lines of $\mathcal{R}_{1,1}$ on the domain $D = [-2, 2] \times [-1, 3]$, as well as its graph.

It is also easy to verify that the image is the interval $[0, \infty)$. Indeed, note first that $\mathcal{R}_{a,b}(x, y) \geq 0$ for all $(x, y) \in \mathbb{R}^2$. Zero is attained: $\mathcal{R}_{a,b}(a, a^2) = 0$. Note also that $\mathcal{R}_{a,b}(0, y) = a^2 + by^2$ is a polynomial of degree 2, therefore unbounded.

Let's locate all local minima:

- The gradient and Hessian are given respectively by

$$\nabla \mathcal{R}_{a,b}(x, y) = \left[2(x - a) + 4bx(x^2 - y), b(y - x^2)\right]$$

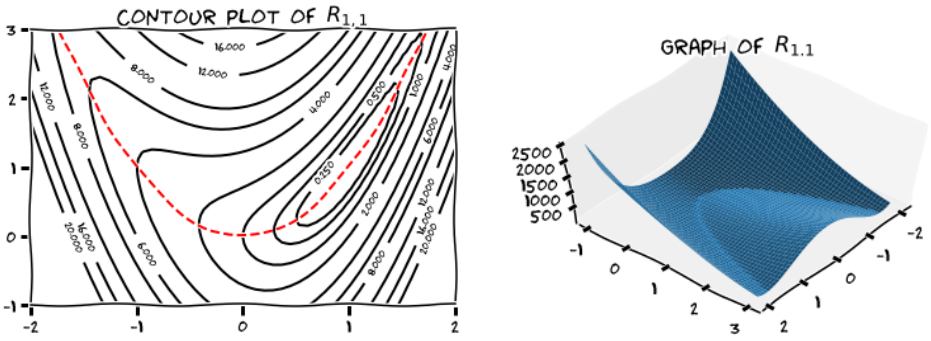


FIGURE 1.1. Details of the graph of $\mathcal{R}_{1,1}$

$$\text{Hess}\mathcal{R}_{a,b}(x,y) = \begin{bmatrix} 12bx^2 - 4by + 2 & -4bx \\ -4bx & 2b \end{bmatrix}$$

- The search for critical points $\nabla\mathcal{R}_{a,b} = \mathbf{0}$ gives only the point (a, a^2).
- $\frac{\partial^2\mathcal{R}_{a,b}}{\partial x^2}(a, a^2) = 8ba^2 + 2 > 0$.
- The Hessian at that point has positive determinant:

$$\det \text{Hess}\mathcal{R}_{a,b}(a, a^2) = \det \begin{bmatrix} 8ba^2 + 2 & -4ab \\ -4ab & 2b \end{bmatrix} = 4b > 0$$

There is only one local minimum at (a, a^2).

The second step was the notion of *global (or absolute) minima*: points (x_0, y_0) that satisfy $f(x_0, y_0) \leq f(x, y)$ for any point (x, y) in the domain of f. We always started with the easier setting, in which we placed restrictions on the domain of our functions:

THEOREM 1.3. *A continuous real-valued function always attains its minimum value on a* compact *set K. If the function is also differentiable in the* interior *of K, to search for global minima we perform the following steps:*

Interior Candidates: *List the critical points of f located in the interior of K.*

Boundary Candidates: *List the points in the boundary of K where f may have minimum values.*

Evaluation/Selection: *Evaluate f at all candidates and select the one(s) with the smallest value.*

EXAMPLE 1.2. A flat circular plate has the shape of the region

$$x^2 + y^2 \leq 1.$$

The plate, including the boundary, is heated so that the temperature at the point (x, y) is given by $f(x, y) = 100(x^2 + 2y^2 - x)$ in Celsius degrees. Find the temperature at the coldest point of the plate.

We start by searching for critical points. The equation $\nabla f(x,y) = 0$ gives $x = \frac{1}{2}$, $y = 0$. The point $(\frac{1}{2}, 0)$ is clearly inside of the plate. This is our first candidate.

The border of the plate can be parameterized by $\varphi(t) = (\cos t, \sin t)$ for $t \in [0, 2\pi)$. The search for minima in the boundary of the plate can then be coded as an optimization problem for the function $h(t) = (f \circ \varphi)(t) = 100(\cos^2 t + 2\sin^2 t - \cos t)$ on the interval $[0, 2\pi)$. Note that $h'(t) = 0$ for $t \in \{0, \frac{2}{3}\pi\}$ in $[0, 2\pi)$. We thus have two more candidates:

$$\varphi(0) = (1, 0) \qquad \varphi(\tfrac{2}{3}\pi) = \big(-\tfrac{1}{2}, \tfrac{1}{2}\sqrt{3}\big)$$

Evaluation of the function at all candidates gives us the solution to this problem:

$$f(\tfrac{1}{2}, 0) = -25^\circ\text{C}.$$

On a second setting, we remove the restriction of boundedness of the function. In this case, global minima will only be guaranteed for very special functions.

EXAMPLE 1.3. Any polynomial $p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$ with even degree $n \geq 2$ and positive leading coefficient satisfies $\lim_{|x|\to\infty} p_n(x) = +\infty$. To see this, we may write

$$a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 = a_n x^n \big(1 + \tfrac{a_{n-1}}{a_n x} + \cdots + \tfrac{a_0}{a_n x^n}\big)$$

The behavior of each of the factors as the absolute value of x goes to infinity leads to our claim.

$$\lim_{|x|\to\infty} a_n x^n = +\infty,$$
$$\lim_{|x|\to\infty} \big(1 + \tfrac{a_{n-1}}{a_n x} + \cdots + \tfrac{a_0}{a_n x^n}\big) = 1.$$

It is clear that a polynomial of this kind must attain a minimum somewhere in its domain. The critical points will lead to them.

EXAMPLE 1.4. Find the global minima of the function

$$f(x) = \log(x^4 - 2x^2 + 2).$$

Note first that the domain of f is the whole real line, since $x^4 - 2x^2 + 2 = (x^2-1)^2 + 1 \geq 1$ for all $x \in \mathbb{R}$. Note also that we can write $f(x) = (g \circ h)(x)$ with $g(x) = \log(x)$ and $h(x) = x^4 - 2x^2 + 1$. Since g is one-to-one and increasing, we can focus on h to obtain the requested solution. For instance, $\lim_{|x|\to\infty} f(x) = +\infty$, since $\lim_{|x|\to\infty} h(x) = +\infty$. This guarantees the existence of global minima. To look for it, h again points to the possible locations by solving for its critical points: $h'(x) = 0$. We have then that f attains its minima at $x = \pm 1$.

We learned other useful characterizations for extrema, when the domain could be expressed as solutions of equations:

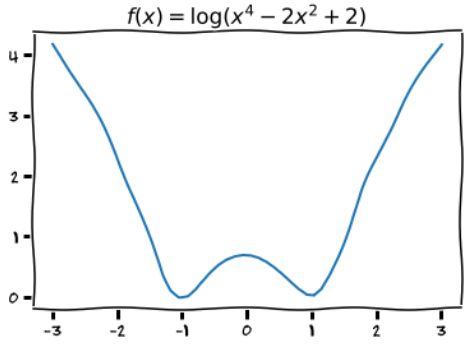


FIGURE 1.2. Global minima in unbounded domains

THEOREM 1.4 (Orthogonal Gradient). *Suppose $f(x, y)$ is differentiable in a region whose interior contains a smooth curve C: $\boldsymbol{r}(t) = \big(x(t), y(t)\big)$. If P_0 is a point on C where f has a local extremum relative to its values on C, then ∇f is orthogonal to C at P_0.*

This result leads to the *Method of Lagrange Multipliers*

THEOREM 1.5 (Lagrange Multipliers on one constraint). *Suppose that $f(x, y)$ and $h_1(x, y)$ are differentiable and $\nabla h_1 \neq 0$ when $h_1(x, y) = 0$. To find the local extrema of f subject to the constraint $h_1(x, y) = 0$ (if these exist), find the values of x, y and λ that simultaneously satisfy the equations*

$$\nabla f = \lambda \nabla h_1, \text{ and } h_1(x, y) = 0$$

EXAMPLE 1.5. Find the minimum value of the expression $3x + 4y$ for values of x and y on the circle $x^2 + y^2 = 1$.

We start by modeling this problem to adapt the technique of Lagrange multipliers:

$$f(x, y) = \underbrace{3x + 4y}_{\text{target}} \qquad h_1(x, y) = \underbrace{x^2 + y^2 - 1}_{\text{constraint}}$$

Look for the values of x, y and λ that satisfy the equations $\nabla f = \lambda \nabla h_1$, $h_1(x, y) = 0$

$$3 = 2\lambda x, \qquad 4 = 2\lambda y \qquad 1 = x^2 + y^2$$

Equivalently, $\lambda \neq 0$ and x, y satisfy

$$x = \frac{3}{2\lambda}, \qquad y = \frac{2}{\lambda}, \qquad 1 = \frac{9}{4\lambda^2} + \frac{4}{\lambda^2}$$

These equations lead to $\lambda = \pm\frac{5}{2}$, and there are only two possible candidates for minimum. Evaluation of f on those gives that the minimum is at the point $\big(-\frac{3}{5}, -\frac{4}{5}\big)$.

This method can be extended to more than two dimensions, and more than one constraint. For instance:

THEOREM 1.6 (Lagrange Multipliers on two constraints). *Suppose that $f(x,y,z)$, $h_1(x,y,z)$, $h_2(x,y,z)$ are differentiable with ∇h_1 not parallel to ∇h_2. To find the local extrema of f subject to the constraint $h_1(x,y,z) = h_2(x,y,z) = 0$ (if these exist), find the values of x, y, λ_1 and λ_2 that simultaneously satisfy the equations*

$$\nabla f = \lambda_1 \nabla h_1 + \lambda_2 \nabla h_2, \qquad h_1(x,y,z) = 0, \qquad h_2(x,y,z) = 0$$

EXAMPLE 1.6. The cylinder $x^2+y^2 = 1$ intersects the plane $x+y+z = 1$ in an ellipse. Find the points on the ellipse that lies closest to the origin.

We again model this as a Lagrange multipliers problem:

$$f(x,y,z) = \overbrace{x^2+y^2+z^2}^{\text{target}},$$
$$h_1(x,y,z) = \underbrace{x^2+y^2-1}_{\text{constraint}}, \qquad h_2(x,y,z) = \underbrace{x+y+z-1}_{\text{constraint}}.$$

The gradient equation $\nabla f = \lambda_1 \nabla h_1 + \lambda_2 \nabla h_2$ gives

$$2x = 2\lambda_1 x + \lambda_2, \qquad 2y = 2\lambda_1 y + \lambda_2, \qquad 2z = \lambda_2$$

These equations are satisfied simultaneously only in two scenarios:

(a) $\lambda_1 = 1$, $\lambda_2 = 0$ and $z = 0$
(b) $\lambda_1 \neq 1$ and $x = y = z/(1-\lambda_1)$

Resolving each case we find four candidates:

$$(1,0,0), \quad (0,1,0), \quad (\sqrt{2}/2, \sqrt{2}/2, 1-\sqrt{2}), \quad (-\sqrt{2}/2, -\sqrt{2}/2, 1+\sqrt{2}).$$

The first two are our solution.

The Theory of Optimization

The purpose of these notes is the development of a theory to deal with optimization in a more general setting.

- We start in an Euclidean d–dimensional space with the usual topology based on the distance

  $$\mathrm{d}(\boldsymbol{x},\boldsymbol{y}) = \|\boldsymbol{x}-\boldsymbol{y}\| = \langle \boldsymbol{x}-\boldsymbol{y}, \boldsymbol{x}-\boldsymbol{y}\rangle^{1/2} = \sqrt{\sum_{k=1}^{d} (x_k - y_k)^2}.$$

  For instance, the *open ball* of radius $r > 0$ centered at a point $\boldsymbol{x}^\star$ is the set $B_r(\boldsymbol{x}^\star) = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}-\boldsymbol{x}^\star\| < r\}$.
- Given a real-valued function $f\colon D \to \mathbb{R}$ on a domain $D \subseteq \mathbb{R}^d$, we define the concept of *extrema* and *extreme Values*:

  DEFINITION. Given a real-valued function $f\colon D \to \mathbb{R}$ on a domain $D \subseteq \mathbb{R}^d$, we say that a point $\boldsymbol{x}^\star \in D$ is a:
  global minimum: $f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$.
  global maximum: $f(\boldsymbol{x}^\star) \geq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$.
  strict global minimum: $f(\boldsymbol{x}^\star) < f(\boldsymbol{x})$ for all $\boldsymbol{x} \in D \setminus \{\boldsymbol{x}^\star\}$.

strict global maximum: $f(\boldsymbol{x}^\star) > f(\boldsymbol{x})$ for all $\boldsymbol{x} \in D \setminus \{\boldsymbol{x}^\star\}$.
local minimum: There exists $\delta > 0$ so that $f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in B_\delta(\boldsymbol{x}^\star) \cap D$.
local maximum: There exists $\delta > 0$ so that $f(\boldsymbol{x}^\star) \geq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in B_\delta(\boldsymbol{x}^\star) \cap D$.
strict local minimum: There exists $\delta > 0$ so that $f(\boldsymbol{x}^\star) < f(\boldsymbol{x})$ for all $\boldsymbol{x} \in B_\delta(\boldsymbol{x}^\star) \cap D$, $\boldsymbol{x} \neq \boldsymbol{x}^\star$.
strict local maximum: There exists $\delta > 0$ so that $f(\boldsymbol{x}^\star) > f(\boldsymbol{x})$ for all $\boldsymbol{x} \in B_\delta(\boldsymbol{x}^\star) \cap D$, $\boldsymbol{x} \neq \boldsymbol{x}^\star$.

In this setting, the objective of *optimization* is the search for extrema in the following two scenarios:

Unconstrained Optimization: if D is an open set (usually the whole space $\mathbb{R}^d$).
Constrained Optimization: if D can be described as a set of *constraints*: $\boldsymbol{x} \in D$ if there exist $m, n \in \mathbb{N}$ and functions $g_k \colon \mathbb{R}^d \to \mathbb{R}$ $(1 \leq k \leq m)$, $h_j \colon \mathbb{R}^d \to \mathbb{R}$ $(1 \leq j \leq n)$ so that

$$g_k(\boldsymbol{x}) \leq 0 \quad (1 \leq k \leq m)$$
$$h_j(\boldsymbol{x}) = 0 \quad (1 \leq j \leq n)$$

For each of these problems, we follow a similar program:

Existence of extrema: Establish results that guarantee the existence of extrema depending on the properties of D and f.
Characterization of extrema: Establish results that describe conditions for points $\boldsymbol{x} \in D$ to be extrema of f.
Tracking extrema: Design robust numerical algorithms that find the extrema for scientific computing purposes.

The development of existence of solutions of any optimization problem, as well as the characterization results for unconstrained optimization will be covered in chapter 2. The design of algorithms to track extrema in the unconstrained setting will be covered in chapter 3. Chapter 4 is devoted to characterization results for constrained optimization, and Chapter 5 for the design of algorithms in that setting.

Exercises

PROBLEM 1.1 (Advanced). State and prove similar statements as in Definition 1, Theorems 1.1, 1.2 and 1.3, but for *local* and *global maxima*.

PROBLEM 1.2 (Basic). Find and sketch the domain of the following functions.

(a) $f(x, y) = \sqrt{y - x - 2}$
(b) $f(x, y) = \log\left(x^2 + y^2 - 4\right)$
(c) $f(x, y) = \frac{(x-1)(y+2)}{(y-x)(y-x^3)}$

(d) $f(x, y) = \log(xy + x - y - 1)$

PROBLEM 1.3 (Basic). Find and sketch the level lines $f(x, y) = c$ on the same set of coordinate axes for the given values of c.

(a) $f(x, y) = x + y - 1$, $c \in \{-3, -2, -1, 0, 1, 2, 3\}$.
(b) $f(x, y) = x^2 + y^2$, $c \in \{0, 1, 4, 9, 16, 25\}$.
(c) $f(x, y) = xy$, $c \in \{-9, -4, -1, 0, 1, 4, 9\}$

PROBLEM 1.4 (CAS). Use a Computer Algebra System of your choice to produce contour plots of the given functions on the given domains.

(a) $f(x, y) = (\cos x)(\cos y)e^{-\sqrt{x^2+y^2}/4}$ on $[-2\pi, 2\pi] \times [-2\pi, 2\pi]$.
(b) $g(x, y) = \dfrac{xy(x^2 - y^2)}{x^2 + y^2}$ on $[-1, 1] \times [-1, 1]$
(c) $h(x, y) = y^2 - y^4 - x^2$ on $[-1, 1] \times [-1, 1]$
(d) $k(x, y) = e^{-y} \cos x$ on $[-2\pi, 2\pi] \times [-2, 0]$

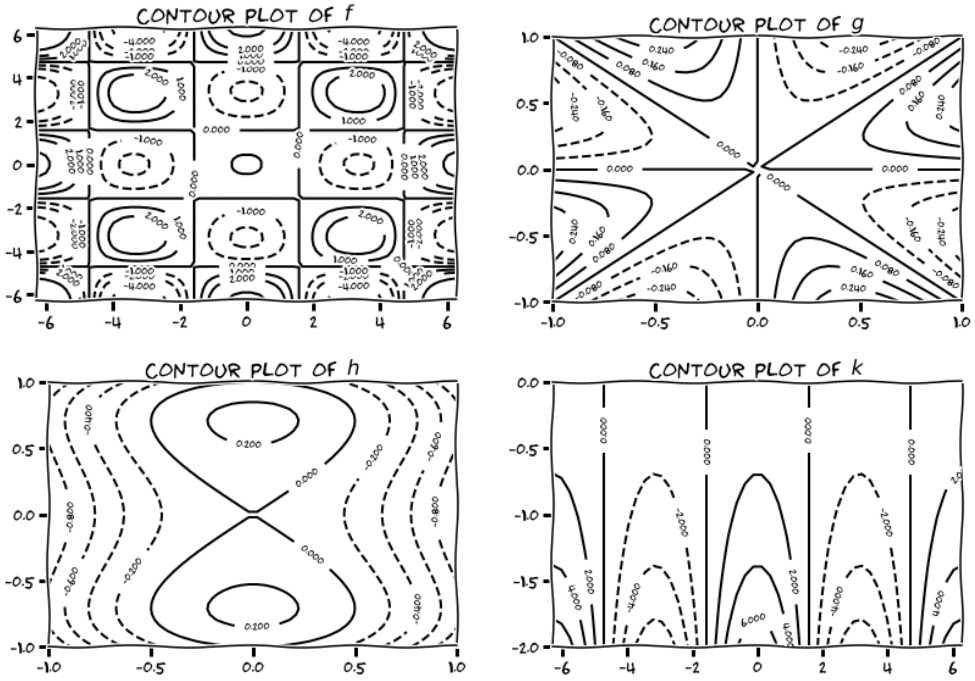


FIGURE 1.3. Contour plots for problem 1.4

PROBLEM 1.5 (Basic). Sketch the curve $f(x, y) = c$ together with ∇f and the tangent line at the given point. Write an equation for the tangent line.

(a) $f(x, y) = x^2 + y^2$, $c = 4$, $(\sqrt{2}, \sqrt{2})$.
(b) $f(x, y) = x^2 - y$, $c = 1$, $(\sqrt{2}, 1)$.
(c) $f(x, y) = xy$, $c = -1$, $(2, -1/2)$.

(d) $f(x,y) = x^2 - xy + y^2$, $c = 7$, $(-1,2)$.

PROBLEM 1.6 (Basic). For the function

$$f(x,y) = \frac{x-y}{x+y},$$

at the point $P_0 = (-1/2, 3/2)$, find the directions $\boldsymbol{v}$ and the directional derivatives $D_{\boldsymbol{v}} f(P_0)$ for which

(a) $D_{\boldsymbol{v}} f(P_0)$ is largest.
(b) $D_{\boldsymbol{v}} f(P_0)$ is smallest.
(c) $D_{\boldsymbol{v}} f(P_0) = 0$.
(d) $D_{\boldsymbol{v}} f(P_0) = 1$.
(e) $D_{\boldsymbol{v}} f(P_0) = -2$.

PROBLEM 1.7 (Intermediate). The derivative of $f(x,y)$ at $(1,2)$ in the direction $\frac{\sqrt{2}}{2}[1,1]$ is $2\sqrt{2}$ and in the direction $[0,-1]$ is -3. What is the derivative of f in the direction $\frac{\sqrt{5}}{5}[-1,-2]$?

PROBLEM 1.8 (Intermediate). Find the absolute maxima and minima of the function $f(x,y) = (4x - x^2)\cos y$ on the rectangular plate $1 \le x \le 3, -\frac{\pi}{4} \le y \le \frac{\pi}{4}$.

PROBLEM 1.9 (Basic). Find two numbers $a \le b$ such that

$$\int_a^b (24 - 2x - x^2)^{1/3}\, dx$$

has its largest value.

PROBLEM 1.10 (Basic). Find the points of the hyperbolic cylinder $x^2 - z^2 - 1 = 0$ in $\mathbb{R}^3$ that are closest to the origin.

PROBLEM 1.11 (Intermediate). Find the extreme values of the function $f(x,y,z) = xy + z^2$ on the circle in which the plane $y - x = 0$ intersects the sphere $x^2 + y^2 + z^2 = 4$.

PROBLEM 1.12 (CAS). Write a routine (in your favorite CAS) that uses *symbolic computation*[1] to find the minimum of a differentiable real-valued function $f\colon \mathbb{R} \to \mathbb{R}$ over

(a) a closed interval $[a,b]$
(b) An interval of the form $[a,\infty)$, or $(-\infty,b]$

The routine should accept as input:

- the expression of the function f,
- the endpoints a, b.

[1] See Appendix B

CHAPTER 2

Existence and Characterization of Extrema for Unconstrained Optimization

In this chapter we will study different properties of functions and domains that guarantee existence of extrema for unconstrained optimization. Once we have them, we explore characterization of those points. We start with a reminder of the definition of continuous and differentiable functions, and then we proceed to introduce other functions with advantageous properties for optimization purposes.

1. Anatomy of a function

1.1. Continuity and Differentiability.

DEFINITION. We say that a function $\boldsymbol{f}\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ is continuous at a point $\boldsymbol{x}^\star \in \mathbb{R}^{d_1}$ if for all $\varepsilon > 0$ there exists $\delta > 0$ so that for all $\boldsymbol{x} \in \mathbb{R}^{d_1}$ satisfying $\|\boldsymbol{x} - \boldsymbol{x}^\star\|_{d_1} < \delta$, it is $\|f(\boldsymbol{x}) - f(\boldsymbol{x}^\star)\|_{d_2} < \varepsilon$.

EXAMPLE 2.1. Let $f\colon \mathbb{R}^2 \to \mathbb{R}$ be given by

$$f(x,y) = \begin{cases} \frac{2xy}{x^2+y^2}, & (x,y) \neq (0,0) \\ 0, & (x,y) = (0,0) \end{cases}$$

This function is trivially continuous at any point $(x,y) \neq (0,0)$. However, it fails to be continuous at the origin. Notice how we obtain different values as we approach $(0,0)$ through different generic lines $y = mx$ with $m \in \mathbb{R}$:

$$\lim_{x\to 0} f(x,mx) = \lim_{x\to 0} \frac{2mx^2}{(1+m^2)x^2} = \frac{2m}{1+m^2}.$$

DEFINITION. A function $\boldsymbol{T}\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ is said to be a *linear map* (or a *linear transformation*) if it satisfies

$$\boldsymbol{T}(\boldsymbol{x} + \lambda \boldsymbol{y}) = \boldsymbol{T}(\boldsymbol{x}) + \lambda \boldsymbol{T}(\boldsymbol{y}) \text{ for all } \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{d_1}, \lambda \in \mathbb{R}.$$

The *kernel* and *image* of a linear map are respectively given by

$$\ker T = \{\boldsymbol{x} \in \mathbb{R}^{d_1} : T(\boldsymbol{x}) = \boldsymbol{0}\},$$
$$\operatorname{im} T = \{\boldsymbol{y} \in \mathbb{R}^{d_2} : \text{ there exists } \boldsymbol{x} \in \mathbb{R}^{d_1} \text{ so that } \boldsymbol{y} = T(\boldsymbol{x})\}.$$

REMARK 2.1. For each real-valued linear map $T\colon \mathbb{R}^d \to \mathbb{R}$ there exists $\boldsymbol{a} \in \mathbb{R}^d$ so that $T(\boldsymbol{x}) = \langle \boldsymbol{a}, \boldsymbol{x} \rangle$ for all $\boldsymbol{x} \in \mathbb{R}^d$.

The kernel of a linear map in this case has a very simple expression:

$$\ker T = \ker\langle \boldsymbol{a}, \cdot\rangle = \{(x_1, x_2, \dots, x_d) \in \mathbb{R}^d : a_1x_1 + a_2x_2 + \cdots + a_dx_d = 0\}$$

The graph of a real-valued linear function can be identified with a hyperplane in $\mathbb{R}^{d+1}$:

$$\begin{aligned}\operatorname{Graph} T &= \{(\boldsymbol{x}, y) \in \mathbb{R}^{d+1} : y = T(\boldsymbol{x})\}\\ &= \{(x_1, x_2, \dots, x_d, y) \in \mathbb{R}^{d+1} : a_1x_1 + a_2x_2 + \cdots + a_dx_d = y\}\\ &= \ker\langle [a_1, a_2, \dots, a_d, -1], \cdot\rangle\end{aligned}$$

REMARK 2.2. For each linear map $\boldsymbol{T}\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ there exists a matrix $\boldsymbol{A}$ of size $d_1 \times d_2$ so that $\boldsymbol{T}(\boldsymbol{x})^\intercal = \boldsymbol{A} \cdot \boldsymbol{x}^\intercal$.

$$\begin{cases} y_1 &= a_{11}x_1 + \cdots + a_{1d_1}x_{d_1}\\ y_2 &= a_{21}x_1 + \cdots + a_{2d_1}x_{d_1}\\ &\vdots\\ y_{d_2} &= a_{d_21}x_1 + \cdots + a_{d_1d_2}x_{d_1}\end{cases}$$

DEFINITION. A function $\boldsymbol{f}\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ is said to be *differentiable* at $\boldsymbol{x}^\star$ if there exists a *linear map* $\boldsymbol{J}\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ so that

$$\lim_{\boldsymbol{h}\to\boldsymbol{0}} \frac{\|\boldsymbol{f}(\boldsymbol{x}^\star + \boldsymbol{h}) - \boldsymbol{f}(\boldsymbol{x}^\star) - \boldsymbol{J}(\boldsymbol{h})\|_{d_2}}{\|\boldsymbol{h}\|_{d_1}} = 0$$

EXAMPLE 2.2. Consider a real-valued function $f\colon \mathbb{R} \to \mathbb{R}$ of a real variable. To prove differentiability at a point $x^\star$, we need a linear map: $J(h) = ah$ for some $a \in \mathbb{R}$. Notice how in that case,

$$\frac{|f(x^\star + h) - f(x^\star) - J(h)|}{|h|} = \left|\frac{f(x^\star + h) - f(x^\star)}{h} - a\right|;$$

therefore, we could pick $a = \lim_{h\to 0} h^{-1}\big(f(x^\star + h) - f(x^\star)\big)$—this is the definition of derivative we learned in Calculus: $a = f'(x^\star)$.

REMARK 2.3. If a function $\boldsymbol{f}\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ is differentiable at $\boldsymbol{x}^\star$, then all of the partial derivatives exist at $\boldsymbol{x}^\star$, in which case the linear map $\boldsymbol{J}$ is given by the *Jacobian matrix*

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_{d_1}}\\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_{d_1}}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial f_{d_2}}{\partial x_1} & \frac{\partial f_{d_2}}{\partial x_2} & \cdots & \frac{\partial f_{d_2}}{\partial x_{d_1}}\end{bmatrix} \cdot \begin{bmatrix} x_1\\ x_2\\ \vdots\\ x_{d_1}\end{bmatrix}$$

The converse is not true in general: the existence of partial derivatives (or even all of the directional derivatives) is not guarantee that a function is differentiable at a point. For instance, the function $f\colon \mathbb{R}^2 \to \mathbb{R}$ given by

$$f(x, y) = \begin{cases} y^3/(x^2 + y^2) & \text{if } (x, y) \neq (0, 0)\\ 0 & \text{if } (x, y) = (0, 0)\end{cases}$$

is not differentiable at $(0,0)$, although all partial derivatives and all directional derivatives exist at that point.

A *friendly* version of the differentiability of real-valued functions comes with the next result (see, e.g. [**6**, p.818])

THEOREM 2.1. *If the partial derivatives* $\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d}$ *of a real-valued function* $f\colon \mathbb{R}^d \to \mathbb{R}$ *are continuous on an open region* $G \subseteq \mathbb{R}^d$*, then* f *is differentiable at every point of* G*.*

EXAMPLE 2.3. Let $f\colon \mathbb{R}^d \to \mathbb{R}$. To prove that f is differentiable at a point $\boldsymbol{x}^\star \in \mathbb{R}^d$ we need a linear map $J(h) = \langle \boldsymbol{a}, h\rangle$ for some $\boldsymbol{a} \in \mathbb{R}^d$. Under the conditions of Theorem 2.1 we may use

$$\boldsymbol{a} = \nabla f(\boldsymbol{x}^\star) = \left[\frac{\partial f(\boldsymbol{x}^\star)}{\partial x_1}, \ldots, \frac{\partial f(\boldsymbol{x}^\star)}{\partial x_d}\right],$$

if we are able to prove that all partial derivatives are continuous in an open set containing $\boldsymbol{x}^\star$.

It is a simple task to prove that all differentiable functions are continuous. Is it true that all continuous functions are differentiable?

EXAMPLE 2.4 (Weierstrass Function). For any positive real numbers a, b satisfying $0 < a < 1 < b$ and $ab \geq 1$, consider the Weierstrass function $\mathcal{W}_{a,b}\colon \mathbb{R} \to \mathbb{R}$ given by

$$\mathcal{W}_{a,b}(x) = \sum_{n=0}^{\infty} a^n \cos(b^n \pi x)$$

This function is continuous everywhere, yet *nowhere* differentiable! (see Figure 2.1). For a proof, see e.g. [**11**]

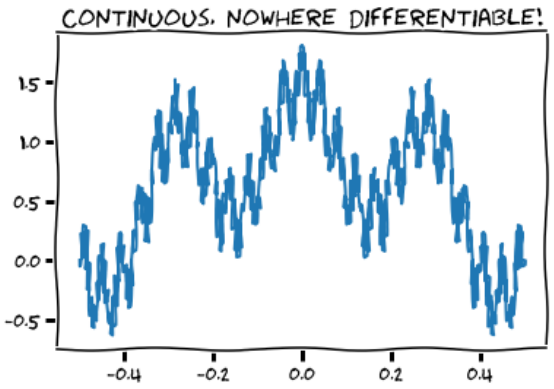


FIGURE 2.1. Detail of the graph of $\mathcal{W}_{0.5,7}$

A few more useful results about higher order derivatives follow:

THEOREM 2.2 (Clairaut). *If $f\colon \mathbb{R}^d \to \mathbb{R}$ and its partial derivatives of orders 1 and 2, $\frac{\partial f}{\partial x_k}$, $\frac{\partial^2 f}{\partial x_k \partial x_j}$, $(1 \le k, j \le d)$ are defined throughout an open region containing the point $\boldsymbol{x}^\star$, and are all continuous at $\boldsymbol{x}^\star$, then*

$$\frac{\partial^2 f(\boldsymbol{x}^\star)}{\partial x_k \partial x_j} = \frac{\partial^2 f(\boldsymbol{x}^\star)}{\partial x_j \partial x_k}, \quad (1 \le k, j \le d).$$

DEFINITION (Hessian). Given a twice-differentiable function $f\colon \mathbb{R}^d \to \mathbb{R}$, we define the *Hessian* of f at $\boldsymbol{x}$ to be the following matrix of second partial derivatives:

$$\operatorname{Hess} f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1^2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_d \partial x_1} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_d^2} \end{bmatrix}$$

Functions that satisfy the conditions of Theorem 2.2 have symmetric Hessians. We shall need some properties in regard to symmetric matrices.

DEFINITION. Given a symmetric matrix $\boldsymbol{A}$, we define its associated *quadratic form* as the function $\mathcal{Q}_{\boldsymbol{A}}\colon \mathbb{R}^d \to \mathbb{R}$ given by

$$\mathcal{Q}_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{A}\boldsymbol{x}^\intercal = \begin{bmatrix} x_1 \cdots x_d \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{1d} & \cdots & a_{dd} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

We say that a symmetric matrix is:
positive definite: if $\mathcal{Q}_{\boldsymbol{A}}(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \mathbb{R}^d \setminus \{\boldsymbol{0}\}$.
positive semidefinite: if $\mathcal{Q}_{\boldsymbol{A}}(\boldsymbol{x}) \ge 0$ for all $\boldsymbol{x} \in \mathbb{R}^d$.
negative definite: if $\mathcal{Q}_{\boldsymbol{A}}(\boldsymbol{x}) < 0$ for all $\boldsymbol{x} \in \mathbb{R}^d \setminus \{\boldsymbol{0}\}$.
negative semidefinite: if $\mathcal{Q}_{\boldsymbol{A}}(\boldsymbol{x}) \le 0$ for all $\boldsymbol{x} \in \mathbb{R}^d$.
indefinite: if there exist $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ so that $\mathcal{Q}_{\boldsymbol{A}}(\boldsymbol{x})\mathcal{Q}_{\boldsymbol{A}}(\boldsymbol{y}) < 0$.

EXAMPLE 2.5. Let $\boldsymbol{A}$ be the 3×3–symmetric matrix

$$\boldsymbol{A} = \begin{bmatrix} 2 & -1 & 2 \\ -1 & 3 & 0 \\ 2 & 0 & 5 \end{bmatrix}$$

The associated quadratic form is given by

$$\begin{aligned}\mathcal{Q}_{\boldsymbol{A}}(x,y,z) &= \begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} 2 & -1 & 2 \\ -1 & 3 & 0 \\ 2 & 0 & 5 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ &= \begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} 2x - y + 2z \\ -x + 3y \\ 2x + 5z \end{bmatrix} \\ &= x(2x - y + 2z) + y(-x + 3y) + z(2x + 5z) \\ &= 2x^2 + 3y^2 + 5z^2 - 2xy + 4xz\end{aligned}$$

To easily classify symmetric matrices, we usually employ any of the following three criteria:

DEFINITION. Given a general square matrix $\boldsymbol{A}$, we define for each $1 \leq \ell \leq d$, a *principal minor of order* ℓ as the determinant of the submatrix obtained by deleting $d-\ell$ rows and the $d-\ell$ columns with the same indices. There are always $\binom{d}{\ell}$ principal minors of order ℓ. We write $\boldsymbol{A}_\ell^d$ for any of them.

Among those principal minors, we define the *leading principal minors* as those that correspond to the upper left-hand corner $\ell \times \ell$–submatrix of $\boldsymbol{A}$. We denote them by Δ_ℓ.

$$\begin{array}{c} \Delta_1 \quad \Delta_2 \quad \Delta_3 \\ \left[\begin{array}{c:c:c:ccc} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ \hdashline a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \hdashline a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \hdashline \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{array}\right] \end{array}$$

THEOREM 2.3 (Principal Minor Criteria). *A symmetric matrix $\boldsymbol{A}$ is:*

- *Positive definite if and only if $\Delta_\ell > 0$ for all $1 \leq \ell \leq d$.*
- *Negative definite if and only if $(-1)^\ell \Delta_\ell > 0$ for all $1 \leq \ell \leq d$.*
- *Positive semidefinite if and only if $\boldsymbol{A}_\ell^d \geq 0$ for all principal minors, for all $1 \leq \ell \leq d$.*
- *Negative semidefinite if and only if $(-1)^\ell \boldsymbol{A}_\ell^d \geq 0$ for all principal minors, for all $1 \leq \ell \leq d$.*

EXAMPLE 2.6. The matrix $\boldsymbol{A}$ in Example 2.5 is positive definite:

$$\begin{aligned}&\Delta_1 = 2 > 0, \\ &\Delta_2 = \det \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix} = 5 > 0, \\ &\Delta_3 = \det \boldsymbol{A} = 2 \det \begin{bmatrix} -1 & 3 \\ 2 & 0 \end{bmatrix} + 5 \det \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix} = 13 > 0.\end{aligned}$$

EXAMPLE 2.7. Consider the symmetric matrix $\boldsymbol{A}$ below:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 4 & 6 \\ 4 & 2 & 1 \\ 6 & 1 & 6 \end{bmatrix}$$

Computing the leading principal minors, we obtain

$$\begin{aligned} \Delta_1 &= 1 > 0, \\ \Delta_2 &= \det \begin{bmatrix} 1 & 4 \\ 4 & 2 \end{bmatrix} = -14 < 0, \\ \Delta_3 &= \det \boldsymbol{A} = -109 < 0. \end{aligned}$$

The principal leading minors we have computed do not fit with any of the criteria in Theorem 2.3. We can conclude that $\boldsymbol{A}$ is indefinite.

THEOREM 2.4 (Eigenvalue Criteria). *Given a general square $d \times d$ matrix $\boldsymbol{A}$, consider the function $p_{\boldsymbol{A}} \colon \mathbb{C} \to \mathbb{C}$ given by $p_{\boldsymbol{A}}(\lambda) = \det\big(\boldsymbol{A} - \lambda \boldsymbol{I}_d\big)$. This is a polynomial of (at most) degree d in λ. We call it the* characteristic polynomial *of $\boldsymbol{A}$. The roots (in $\mathbb{C}$) of the characteristic polynomial are called the* eigenvalues *of $\boldsymbol{A}$. Symmetric matrices enjoy the following properties:*

(a) The eigenvalues of a symmetric matrix are all real.
(b) If $\lambda \in \mathbb{R}$ is a root of multiplicity n of the characteristic polynomial of a (non-trivial) symmetric matrix, then there exist n linearly independent vectors $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ satisfying $\boldsymbol{A}\boldsymbol{x}_k = \lambda \boldsymbol{x}_k$ $(1 \leq k \leq n)$.
(c) If $\lambda_1 \neq \lambda_2$ are different roots of the characteristic polynomial of a symmetric matrix, and $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d$ satisfy $\boldsymbol{A}\boldsymbol{x}_k = \lambda_k \boldsymbol{x}_k$ $(k = 1, 2)$, then $\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle = 0$.
(d) A symmetric matrix is positive definite (resp. negative definite) if and only if all its eigenvalues are positive (resp. negative).
(e) A symmetric matrix is positive semidefinite (resp. negative semidefinite) if and only if all its eigenvalues are non-negative (resp. non-positive)
(f) A symmetric matrix is indefinite if there exist two eigenvalues $\lambda_1 \neq \lambda_2$ with different sign.

EXAMPLE 2.8. Let's compute the eigenvalues of matrix $\boldsymbol{A}$ in Example 2.5:

$$\begin{aligned} \det(\boldsymbol{A} - \lambda \boldsymbol{I}) &= \det \begin{bmatrix} 2-\lambda & -1 & 2 \\ -1 & 3-\lambda & 0 \\ 2 & 0 & 5-\lambda \end{bmatrix} \\ &= 2 \det \begin{bmatrix} -1 & 2 \\ 3-\lambda & 0 \end{bmatrix} + (5-\lambda) \det \begin{bmatrix} 2-\lambda & -1 \\ -1 & 3-\lambda \end{bmatrix} \\ &= -4(3-\lambda) + (5-\lambda)\big((2-\lambda)(3-\lambda) - 1\big) \\ &= -\lambda^3 + 10\lambda^2 - 26\lambda + 13 \end{aligned}$$

Notice that this polynomial has three real roots: $\lambda_1 \approx 0.653537810432577$, $\lambda_2 \approx 3.27775436623747$, and $\lambda_3 \approx 6.06870782332996$, all of them positive (as we expected).

1.2. Coercive Functions. Other set of functions that play an important role in optimization are the kind of functions we explored in Example 1.3.

DEFINITION (Coercive functions). A continuous real-valued function f is said to be *coercive* if for all $M > 0$ there exists $R = R(M) > 0$ so that $f(\boldsymbol{x}) \geq M$ if $\|\boldsymbol{x}\| \geq R$.

REMARK 2.4. This is equivalent to the limit condition

$$\lim_{\|\boldsymbol{x}\| \to \infty} f(\boldsymbol{x}) = +\infty.$$

EXAMPLE 2.9. We saw in Example 1.3 how even-degree polynomials with positive leading coefficients are coercive, and how this helped guarantee the existence of a minimum.

We must be careful assessing coerciveness of polynomials in higher dimension. Consider for example $p_2(x, y) = x^2 - 2xy + y^2$. Note how $p_2(x, x) = 0$ for any $x \in \mathbb{R}$, which proves p_2 is not coercive.

To see that the polynomial $p_4(x, y) = x^4 + y^4 - 4xy$ is coercive, we start by factoring the leading terms:

$$x^4 + y^4 - 4xy = \left(x^4 + y^4\right)\left(1 - \frac{4xy}{x^4 + y^4}\right)$$

Assume $r > 1$ is large, and that $x^2 + y^2 = r^2$. We have then

$$x^4 + y^4 \geq \frac{r^4}{2} \qquad \text{(Why?)}$$

$$|xy| \leq \frac{r^2}{2} \qquad \text{(Why?)}$$

therefore,

$$\frac{4xy}{x^4 + y^4} \leq \frac{4}{r^2}$$

$$1 - \frac{4xy}{x^4 + y^4} \geq 1 - \frac{4}{r^2}$$

$$\left(x^4 + y^4\right)\left(1 - \frac{4xy}{x^4 + y^4}\right) \geq \frac{r^2(r^2 - 4)}{2}$$

We can then conclude that given $M > 0$, if $x^2 + y^2 \geq 2 + \sqrt{4 + 2M}$, then $p_4(x, y) \geq M$. This proves p_4 is coercive.

1.3. Convex Functions. There is one more kind of functions we should explore.

DEFINITION (Convex Sets). A subset $C \subseteq \mathbb{R}^d$ is said to be *convex* if for every $\boldsymbol{x}, \boldsymbol{y} \in C$, and every $\lambda \in [0, 1]$, the point $\lambda\boldsymbol{y} + (1 - \lambda)\boldsymbol{x}$ is also in C.

The following result is an interesting characterization of convex sets that allows us to actually construct any convex set from a family of points.

THEOREM 2.5. *Let $C \subseteq \mathbb{R}^d$ be a convex set and let $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset C$ be a family of points in C. The convex combinations $\lambda_1\boldsymbol{x}_1+\lambda_2\boldsymbol{x}_2+\cdots+\lambda_n\boldsymbol{x}_n$ are also in C, provided $\lambda_k \geq 0$ for all $1 \leq k \leq n$ and $\lambda_1 + \lambda_2 + \cdots + \lambda_n = 1$.*

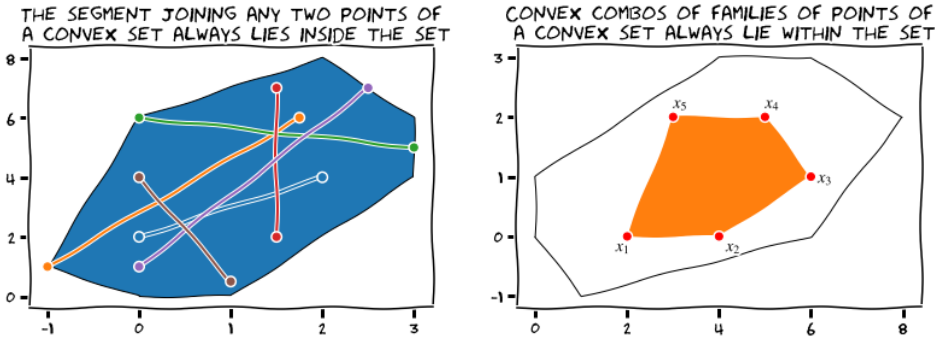


FIGURE 2.2. Convex sets.

DEFINITION (Convex Functions). Given a convex set $C \subseteq \mathbb{R}^d$, we say that a real-valued function $f\colon C \to \mathbb{R}$ is *convex* if

$$f\big(\lambda\boldsymbol{y} + (1-\lambda)\boldsymbol{x}\big) \leq \lambda f(\boldsymbol{y}) + (1-\lambda)f(\boldsymbol{x})$$

If instead we have $f\big(\lambda\boldsymbol{x}+(1-\lambda)f(\boldsymbol{y})\big) < \lambda f(\boldsymbol{x})+(1-\lambda)f(\boldsymbol{y})$ for $0 < \lambda < 1$, we say that the function is *strictly convex*. A function f is said to be *concave* (resp. *strictly concave*) if $-f$ is convex (resp. strictly convex).

REMARK 2.5. There is an alternative definition of convex functions using the concept of *epigraph* of a function. Given a convex function $f\colon C \to \mathbb{R}$ on a convex set C, the epigraph of f is a set $\operatorname{epi}(f) \subset \mathbb{R}^{d+1}$ defined by

$$\operatorname{epi}(f) = \{(\boldsymbol{x}, y) \in \mathbb{R}^{d+1} : \boldsymbol{x} \in C, y \in \mathbb{R}, f(\boldsymbol{x}) \leq y\}.$$

The function f is convex if and only if its epigraph is a convex set.

Convex functions have many pleasant properties:

THEOREM 2.6. *Convex functions are continuous.*

THEOREM 2.7. *Let $f\colon C \to \mathbb{R}$ be a real-valued convex function defined on a convex set $C \subseteq \mathbb{R}^d$. If $\lambda_1, \ldots, \lambda_n$ are nonnegative numbers satisfying $\lambda_1 + \cdots + \lambda_n = 1$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are n different points in C, then*

$$f\big(\lambda_1\boldsymbol{x}_1 + \cdots + \lambda_n\boldsymbol{x}_n\big) \leq \lambda_1 f(\boldsymbol{x}_1) + \cdots + \lambda_n f(\boldsymbol{x}_n).$$

THEOREM 2.8. *If $f\colon C \to \mathbb{R}$ is a function on a convex set $C \subseteq \mathbb{R}^d$ with continuous first partial derivatives on C, then*

(a) f is convex if and only if for all $\boldsymbol{x}, \boldsymbol{y} \in C$,

$$f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle \leq f(\boldsymbol{y}).$$

(b) f is strictly convex if for all $\boldsymbol{x} \neq \boldsymbol{y} \in C$,

$$f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle < f(\boldsymbol{y}).$$

REMARK 2.6. Theorem 2.8 implies that the graph of any (strictly) convex function always lies over the tangent hyperplane at any point of the graph.

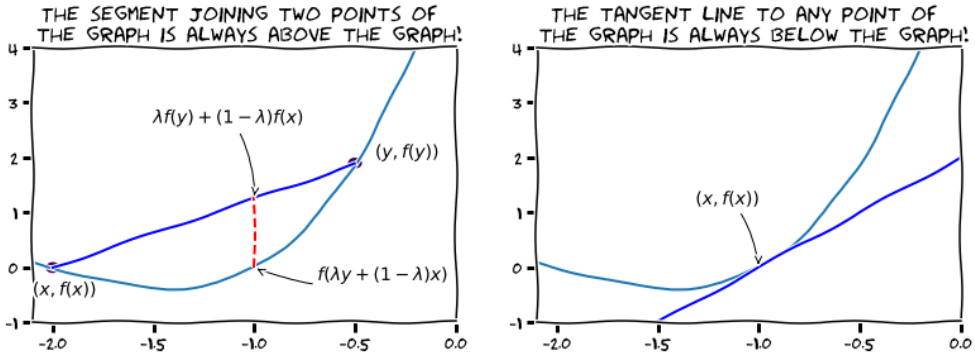


FIGURE 2.3. Convex Functions.

Two more useful characterization of convex functions.

THEOREM 2.9. *Suppose that $f\colon C \to \mathbb{R}$ is a function with second partial derivatives on an open convex set $C \subseteq \mathbb{R}^d$. If the Hessian is positive semidefinite (resp. positive definite) on C, then f is convex (resp. strictly convex).*

THEOREM 2.10. *Let $C \subseteq \mathbb{R}^d$ be a convex set.*

(a) If $f_k\colon C \to \mathbb{R}$ are convex functions for $1 \leq k \leq n$, then so is the sum $f\colon C \to \mathbb{R}$:

$$f(\boldsymbol{x}) = \sum_{k=1}^{n} f_k(\boldsymbol{x}).$$

If at least one of them is strictly convex, then so is f.

(b) If $f\colon C \to \mathbb{R}$ is convex (resp. strictly convex) on C, then so is λf for any $\lambda > 0$.

(c) If $f\colon C \to \mathbb{R}$ is convex (resp. strictly convex) on C, and $g\colon f(C) \to \mathbb{R}$ is an increasing convex function (resp. strictly increasing convex), then so is $g \circ f$.

(d) If $f, g\colon C \to \mathbb{R}$ are convex functions on C, then so is $\max\{f, g\}$.

EXAMPLE 2.10. Consider the function $f(x,y,z)$ defined on $\mathbb{R}^3$ by

$$f(x,y,z) = 2x^2 + y^2 + z^2 + 2yz.$$

Notice that for all $(x,y,z) \in \mathbb{R}^3$,

$$\text{Hess}f(x,y,z) = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix}, \quad \Delta_1 = 4 > 0, \quad \Delta_2 = 8 > 0, \quad \Delta_3 = 0.$$

By virtue of Theorem 2.9, we infer that the function f is convex, but not strictly convex.

EXAMPLE 2.11. To prove that $f(x,y,z) = e^{x^2+y^2+z^2}$ is convex, rather than computing the Hessian and address if it is positive (semi)definite, it is easier to realize that we can write $f = g \circ h$ with

$$\begin{aligned} &g\colon \mathbb{R} \to \mathbb{R} && h\colon \mathbb{R}^3 \to \mathbb{R} \\ &g(x) = e^x && h(x,y,z) = x^2 + y^2 + z^2 \end{aligned}$$

The function g is trivially strictly increasing and convex (since $g'(x) = g''(x) = e^x > 0$ for all $x \in \mathbb{R}$). The function h is strictly convex, since (by Theorem 2.9)

$$\text{Hess}h(x,y,z) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \quad \Delta_1 = 2 > 0, \quad \Delta_2 = 4 > 0, \quad \Delta_3 = 8 > 0.$$

By virtue of (c) in Theorem 2.10, we infer that f is strictly convex.

EXAMPLE 2.12. Set $C = \{(x,y) \in \mathbb{R}^2 : x > 0, y > 0\}$. Consider the function $f\colon C \to \mathbb{R}$ given by

$$f(x,y) = x^2 - 4xy + 5y^2 - \log(xy)$$

Notice we may write $f = g + h$ with $g, h\colon C \to \mathbb{R}$ given respectively by $g(x,y) = x^2 - 4xy + 5y^2$ and $h(x,y) = -\log(xy)$. Note also that both functions are strictly convex, since for all $(x,y) \in C$:

$$\begin{aligned} &\text{Hess}g(x,y) = \begin{bmatrix} 2 & -4 \\ -4 & 10 \end{bmatrix}, && \Delta_1 = 2 > 0, && \Delta_2 = 4 > 0, \\ &\text{Hess}h(x,y) = \begin{bmatrix} x^{-2} & 0 \\ 0 & y^{-2} \end{bmatrix}, && \Delta_1 = x^{-2} > 0, && \Delta_2 = (xy)^{-2} > 0. \end{aligned}$$

By virtue of part (a) in Theorem 2.10, we infer that f is strictly convex.

We find useful to study generalizations of convex functions as well. In these notes we are going to focus on two such generalizations: *quasi-convexity* and *pseudo-convexity*.

DEFINITION (Quasi-convex functions). Given a convex set $C \subseteq \mathbb{R}^d$, we say that a real-valued function $f\colon C \to \mathbb{R}$ is *quasi-convex* if for all $\boldsymbol{x}, \boldsymbol{y} \in C$ and all $0 \leq \lambda \leq 1$,

$$f\big(\lambda \boldsymbol{y} + (1-\lambda)\boldsymbol{x}\big) \leq \max\{f(\boldsymbol{x}), f(\boldsymbol{y})\}.$$

We say that f is *quasi-concave* if

$$f\big(\lambda \boldsymbol{y} + (1-\lambda)\boldsymbol{x}\big) \geq \min\{f(\boldsymbol{x}), f(\boldsymbol{y})\}.$$

DEFINITION (Pseudo-convex functions). Given a convex set $C \subseteq \mathbb{R}^d$, we say that a real-valued differentiable function $f\colon C \to \mathbb{R}$ is *pseudo-convex* if for all $\boldsymbol{x}, \boldsymbol{y} \in C$ satisfying $\langle \nabla f(\boldsymbol{x}), \boldsymbol{x} - \boldsymbol{y}\rangle \geq 0$, it must be $f(\boldsymbol{y}) \geq f(\boldsymbol{x})$

REMARK 2.7. The three functions are related as follows:

- Differentiable convex functions are pseudo-convex.
- Convex functions are quasi-convex.
- Pseudo-convex functions are quasi-convex.

Quasi-convex functions have a very nice characterization by means of their *level sets*:

THEOREM 2.11. *Given a convex set $C \subseteq \mathbb{R}^d$, a real-valued function $f\colon C \to \mathbb{R}$ is quasi-convex if and only if its level sets $\Lambda_t(f) = \{\boldsymbol{x} \in C : f(x) \leq t\}$ are convex for all $t \in \mathbb{R}$.*

We are now ready to explore existence and characterization of extrema in a wide variety of situations.

2. Existence results

2.1. Continuous functions on compact domains. The existence of global extrema is guaranteed for continuous functions over compact sets thanks to the following two basic results:

THEOREM 2.12 (Bounded Value Theorem). *The image $f(K)$ of a continuous real-valued function $f\colon \mathbb{R}^d \to \mathbb{R}$ on a compact set K is bounded: there exists $M > 0$ so that $|f(\boldsymbol{x})| \leq M$ for all $\boldsymbol{x} \in K$.*

THEOREM 2.13 (Extreme Value Theorem). *A continuous real-valued function $f\colon K \to \mathbb{R}$ on a compact set $K \subset \mathbb{R}^d$ takes on minimum and maximum values on K.*

2.2. Continuous functions on unbounded domains. Extra restrictions must be applied to the behavior of f in this case, if we want to guarantee the existence of extrema.

THEOREM 2.14. *Coercive functions always have a global minimum.*

PROOF. Since f is coercive, there exists $r > 0$ so that $f(\boldsymbol{x}) > f(\boldsymbol{0})$ for all $\boldsymbol{x}$ satisfying $\|\boldsymbol{x}\| > r$. On the other hand, consider the closed ball $K_r = \{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{x}\| \leq r\}$. The continuity of f guarantees a global minimum $\boldsymbol{x}^\star \in K_r$ with $f(\boldsymbol{x}^\star) \leq f(\boldsymbol{0})$. It is then $f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^d$ trivially. □

3. Characterization results

Differentiability is key to guarantee characterization of extrema. Critical points lead the way:

THEOREM 2.15 (First order necessary optimality condition for minimization). *Suppose $f\colon \mathbb{R}^d \to \mathbb{R}$ is differentiable at $\boldsymbol{x}^\star$. If $\boldsymbol{x}^\star$ is a local minimum, then $\nabla f(\boldsymbol{x}^\star) = 0$.*

To be able to classify extrema of a properly differentiable function, we take into account the behavior of the function around $f(\boldsymbol{x})$ with respect to the tangent hyperplane at the point $\big(\boldsymbol{x}, f(\boldsymbol{x})\big)$. Second derivatives make this process very easy.

THEOREM 2.16. *Suppose $f\colon \mathbb{R}^d \to \mathbb{R}$ is coercive and continuously differentiable at a point $\boldsymbol{x}^\star$. If $\boldsymbol{x}^\star$ is a global minimum, then $\nabla f(\boldsymbol{x}^\star) = \mathbf{0}$.*

THEOREM 2.17 (Second order necessary optimality condition for minimization). *Suppose that $f\colon \mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable at $\boldsymbol{x}^\star$.*

- *If $\boldsymbol{x}^\star$ is a local minimum, then $\nabla f(\boldsymbol{x}^\star) = 0$ and $\operatorname{Hess} f(\boldsymbol{x}^\star)$ is* positive semidefinite.
- *If $\boldsymbol{x}^\star$ is a strict local minimum, then $\nabla f(\boldsymbol{x}^\star) = 0$ and $\operatorname{Hess} f(\boldsymbol{x}^\star)$ is* positive definite.

THEOREM 2.18 (Second order sufficient optimality conditions for minimization). *Suppose $f\colon D \subseteq \mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable at a point $\boldsymbol{x}^\star$ in the interior of D and $\nabla f(\boldsymbol{x}^\star) = 0$. Then $\boldsymbol{x}^\star$ is a:*

Local Minimum: *if* $\operatorname{Hess} f(\boldsymbol{x}^\star)$ *is* positive semidefinite.
Strict Local Minimum: *if* $\operatorname{Hess} f(\boldsymbol{x}^\star)$ *is* positive definite.
If $D = \mathbb{R}^d$ and $\boldsymbol{x}^\star \in \mathbb{R}^d$ satisfies $\nabla f(\boldsymbol{x}^\star) = 0$, then $\boldsymbol{x}^\star$ is a:
Global Minimum: *if* $\operatorname{Hess} f(\boldsymbol{x})$ *is* positive semidefinite *for all* $\boldsymbol{x} \in \mathbb{R}^d$.
Strict Global Minimum: *if* $\operatorname{Hess} f(\boldsymbol{x})$ *is* positive definite *for all* $\boldsymbol{x} \in \mathbb{R}^d$.

THEOREM 2.19. *Any local minimum of a convex function $f\colon C \to \mathbb{R}$ on a convex set $C \subseteq \mathbb{R}^d$ is also a global minimum. If f is a strictly convex function, then any local minimum is the unique strict global minimum.*

THEOREM 2.20. *Suppose $f\colon C \to \mathbb{R}$ is a convex function with continuous first partial derivatives on a convex set $C \subseteq \mathbb{R}^d$. Then, any critical point of f in C is a global minimum of f.*

Examples

EXAMPLE 2.13. Find a global minimum in $\mathbb{R}^3$ (if it exists) for the function

$$f(x, y, z) = e^{x-y} + e^{y-x} + e^{x^2} + z^2.$$

This function has continuous partial derivatives of any order in $\mathbb{R}^3$. Its continuity does not guarantee existence of a global minimum initially since

the domain is not compact, but we may try our luck with its critical points. Note $\nabla f(x,y,z) = \left[e^{x-y} - e^{y-x} + 2xe^{x^2}, -e^{x-y} + e^{y-x}, 2x\right]$. The only critical point is then $(0,0,0)$ (Why?). The Hessian at that point is positive definite:

$$\mathrm{Hess} f(0,0,0) = \begin{bmatrix} 4 & -2 & 0 \\ -2 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \quad \Delta_1 = 4 > 0, \quad \Delta_2 = 4 > 0, \quad \Delta_3 = 8 > 0.$$

By Theorem 2.18, $f(0,0,0) = 3$ is a priori a strict local global minimum value. To prove that this point is actually a strict global minimum, notice that

$$\mathrm{Hess} f(x,y,z) = \begin{bmatrix} e^{x-y} + e^{y-x} + 4x^2e^{x^2} + 2e^{x^2} & -e^{x-y} - e^{y-x} & 0 \\ -e^{x-y} - e^{y-x} & e^{x-y} + e^{y-x} & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

The first principal minor is trivially positive: $\Delta_1 = e^{x-y} + e^{y-x} + 4x^2e^{x^2} + 2e^{x^2}$, since it is a sum of three positive terms and on non-negative term. The second principal minor is also positive:

$$\begin{aligned} \Delta_2 &= \det \begin{bmatrix} e^{x-y} + e^{y-x} + 4x^2e^{x^2} + 2e^{x^2} & -e^{x-y} - e^{y-x} \\ -e^{x-y} - e^{y-x} & e^{x-y} + e^{y-x} \end{bmatrix} \\ &= (e^{x-y} + e^{y-x})^2 + (e^{x-y} + e^{y-x})(4x^2e^{x^2} + 2e^{x^2}) - (e^{x-y} + e^{y-x})^2 \\ &= (e^{x-y} + e^{y-x})(4x^2e^{x^2} + 2e^{x^2}) > 0 \end{aligned}$$

The third principal minor is positive too: $\Delta_3 = 2\Delta_2 > 0$. We have just proved that $\mathrm{Hess} f(x,y,z)$ is positive definite for all $(x,y,z) \in \mathbb{R}^3$, and thus $(0,0,0)$ is a strict global minimum.

EXAMPLE 2.14. Find global minima in $\mathbb{R}^2$ (if they exist) for the function

$$f(x,y) = e^{x-y} + e^{y-x}.$$

This function also has continuous partial derivatives of any order, but no extrema is guaranteed a priori. Notice that all points (x,y) satisfying $y = x$ are critical. For such points, the corresponding Hessians and eigenvalues are

$$\mathrm{Hess} f(x,x) = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}, \qquad \lambda_1 = 2 > 0, \qquad \lambda_2 = 0;$$

therefore, $\mathrm{Hess} f(x,x)$ is positive semidefinite for each critical point. By Theorem 2.18, $f(x,x) = 2$ is a local minimum for all $x \in \mathbb{R}$. To prove they are global minima, notice that for each $(x,y) \in \mathbb{R}^2$:

$$\mathrm{Hess} f(x,y) = \begin{bmatrix} e^{x-y} + e^{y-x} & -e^{x-y} - e^{y-x} \\ -e^{x-y} - e^{y-x} & e^{x-y} + e^{y-x} \end{bmatrix} = \left(e^{x-y} + e^{y-x}\right) \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$
$$\lambda_1 = e^{x-y} + e^{y-x} > 0, \quad \lambda_2 = 0.$$

The Hessian is positive semidefinite for all points, hence proving that any point in the line $y = x$ is a global minimum of f.

EXAMPLE 2.15. Find local and global minima in $\mathbb{R}^2$ (if they exist) for the function

$$f(x,y) = x^3 - 12xy + 8y^3.$$

This is a polynomial of degree 3, so we have continuous partial derivatives of any order. It is easy to see that this function has no global minima:

$$\lim_{x\to-\infty} f(x,0) = \lim_{x\to-\infty} x^3 = -\infty.$$

Let's search instead for local minima. From the equation $\nabla f(x,y) = \mathbf{0}$ we obtain two critical points: $(0,0)$ and $(2,1)$. The corresponding Hessians and their eigenvalues are:

$$\operatorname{Hess} f(0,0) = \begin{bmatrix} 0 & -12 \\ -12 & 0 \end{bmatrix}, \quad \lambda_1 = -12 < 0, \quad \lambda_2 = 12 > 0,$$

$$\operatorname{Hess} f(2,1) = \begin{bmatrix} 12 & -12 \\ -12 & 48 \end{bmatrix}, \quad \lambda_1 = 30 - 6\sqrt{13} > 0, \quad \lambda_2 = 30 + 6\sqrt{30} > 0.$$

By Theorem 2.18, we have that $f(2,1) = -8$ is a local minimum, but $f(0,0) = 0$ is not.

EXAMPLE 2.16. Find local and global minima in $\mathbb{R}^2$ (if they exist) for the function

$$f(x,y) = x^4 - 4xy + y^4.$$

This is a polynomial of degree 4, so we do have continuous partial derivatives of any order. There are three critical points: $(0,0)$, $(-1,-1)$ and $(1,1)$. The latter two are both strict local minima (by virtue of Theorem 2.18).

$$\operatorname{Hess} f(-1,-1) = \operatorname{Hess} f(1,1) = \begin{bmatrix} 12 & -4 \\ -4 & 12 \end{bmatrix}, \quad \Delta_1 = 12 > 0, \quad \Delta_2 = 128 > 0.$$

We proved in Example 2.9 that f is coercive. By Theorems 2.14 and 2.16 we have that $f(-1,-1) = f(1,1) = -2$ must be strict global minimum values.

Exercises

PROBLEM 2.1 (Basic). Consider the function

$$f(x,y) = \frac{x+y}{2+\cos x}$$

At what points $(x,y) \in \mathbb{R}^2$ is this function continuous?

PROBLEM 2.2 (Advanced). Prove the following facts about linear maps $\boldsymbol{T}\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$:

(a) For the real-valued case $T\colon \mathbb{R}^d \to \mathbb{R}$, there exist a unique vector $\boldsymbol{a} \in \mathbb{R}^d$ so that $T(\boldsymbol{x}) = \langle \boldsymbol{a}, \boldsymbol{x} \rangle$ for all $\boldsymbol{x} \in \mathbb{R}^d$.
(b) For the general case $T\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$, there exists a unique matrix $\boldsymbol{A}$ of size $d_1 \times d_2$ so that $T(\boldsymbol{x})^\intercal = \boldsymbol{A} \cdot \boldsymbol{x}^\intercal$.
(c) Linear maps are continuous functions.

PROBLEM 2.3 (Intermediate). Give an example of a 2×2 symmetric matrix of each kind below:

(a) positive definite,
(b) positive semidefinite,
(c) negative definite,
(d) negative semidefinite,
(e) indefinite.

PROBLEM 2.4 (Basic). [**13**, p.31, #2] Classify the following matrices according to whether they are positive or negative definite or semidefinite or indefinite:

$$\text{(a)}\begin{bmatrix}1&0&0\\0&3&0\\0&0&5\end{bmatrix} \qquad \text{(b)}\begin{bmatrix}-1&0&0\\0&-3&0\\0&0&-2\end{bmatrix} \qquad \text{(c)}\begin{bmatrix}7&0&0\\0&-8&0\\0&0&5\end{bmatrix}$$

$$\text{(d)}\begin{bmatrix}3&1&2\\1&5&3\\2&3&7\end{bmatrix} \qquad \text{(e)}\begin{bmatrix}-4&0&1\\0&-3&2\\1&2&-5\end{bmatrix} \qquad \text{(f)}\begin{bmatrix}2&-4&0\\-4&8&0\\0&0&-3\end{bmatrix}$$

PROBLEM 2.5 (Basic). [**13**, p.31, #3] Write the quadratic form $\mathcal{Q}_{\boldsymbol{A}}(\boldsymbol{x})$ associated with each of the following matrices $\boldsymbol{A}$:

$$\text{(a)}\begin{bmatrix}-1&2\\2&3\end{bmatrix} \qquad \text{(b)}\begin{bmatrix}1&-1&0\\-1&-2&2\\0&2&3\end{bmatrix}$$

$$\text{(c)}\begin{bmatrix}2&-3\\-3&0\end{bmatrix} \qquad \text{(d)}\begin{bmatrix}-3&1&2\\1&2&-1\\2&-1&4\end{bmatrix}$$

PROBLEM 2.6 (Basic). [**13**, p.32, #4] Write each of the quadratic forms below in the form $\boldsymbol{x}\boldsymbol{A}\boldsymbol{x}^{\intercal}$ for an appropriate symmetric matrix $\boldsymbol{A}$:

(a) $3x^2 - xy + 2y^2$.
(b) $x^2 + 2y^2 - 3z^2 + 2xy - 4xz + 6yz$.
(c) $2x^2 - 4z^2 + xy - yz$.

PROBLEM 2.7 (Intermediate). Identify which of the following real-valued functions are coercive. Explain the reason.

(a) $f(x, y) = \sqrt{x^2 + y^2}$.
(b) $f(x, y) = x^2 + 9y^2 - 6xy$.
(c) $f(x, y) = x^4 - 3xy + y^4$.
(d) Rosenbrock functions $\mathcal{R}_{a,b}$.

PROBLEM 2.8 (Advanced). [**13**, p.36, #32] Find an example of a continuous, real-valued, non-coercive function $f\colon \mathbb{R}^2 \to \mathbb{R}$ that satisfies, for all $t \in \mathbb{R}$,

$$\lim_{x\to\infty} f(x, tx) = \lim_{y\to\infty} f(ty, y) = \infty.$$

PROBLEM 2.9 (Basic). Let $C \subseteq \mathbb{R}^2$ be a convex figure. Given a point $P \in C$, let $n(P)$ be the number of chords for which P is a midpoint. For instance, if C is a disk, any point $P \in C$ satisfies $n(P) = 0$ (if the point P is on the circle), $n(P) = \infty$ (if the point P is the center of the disk), or $n(P) = 1$ for any other point in the interior of C.

(a) Are there convex sets that contain points P with $n(P) = 2$? If so, sketch an example.
(b) Are there convex sets that contain points P with $n(P) = m$ for any $m \geq 3$? If so, sketch and example for $m = 3$.

PROBLEM 2.10 (Basic). [**13**, p.77, #1,2,7] Determine whether the given functions are convex, concave, strictly convex or strictly concave on the specified domains:

(a) $f(x) = \log(x)$ on $(0, \infty)$.
(b) $f(x) = e^{-x}$ on $\mathbb{R}$.
(c) $f(x) = |x|$ on $[-1, 1]$.
(d) $f(x) = |x^3|$ on $\mathbb{R}$.
(e) $f(x, y) = 5x^2 + 2xy + y^2 - x + 2x + 3$ on $\mathbb{R}^2$.
(f) $f(x, y) = x^2/2 + 3y^2/2 + \sqrt{3}xy$ on $\mathbb{R}^2$.
(g) $f(x, y) = 4e^{3x-y} + 5e^{x^2+y^2}$ on $\mathbb{R}^2$.
(h) $f(x, y, z) = x^{1/2} + y^{1/3} + z^{1/5}$ on $C = \{(x, y, z) : x > 0, y > 0, z > 0\}$.

PROBLEM 2.11 (Intermediate). [**13**, p.79 #11] Sketch the epigraph of the following functions

(a) $f(x) = e^x$.
(b) $f(x, y) = x^2 + y^2$.

PROBLEM 2.12 (Advanced). Prove the Bounded Value and Extreme Value Theorems (Theorems 2.12 and 2.13).

PROBLEM 2.13 (Intermediate). For the following optimization problems, state whether existence of a solution is guaranteed:

(a) $f(x) = (1 + x)/x$ over $[1, \infty)$
(b) $f(x) = 1/x$ over $[1, 2)$
(c) The following piecewise function over $[1, 2]$

$$f(x) = \begin{cases} 1/x, & 1 \leq x < 2 \\ 1, & x = 2 \end{cases}$$

PROBLEM 2.14 (Advanced). State and prove equivalent results to Theorems 2.15, 2.17 and 2.18 to describe necessary and sufficient conditions for the characterization of *maxima*.

PROBLEM 2.15 (Basic). [**13**, p.32, #7] Use the *Principal Minor Criteria* (Theorem 2.3) to determine—if possible—the nature of the critical points of the following functions:

(a) $f(x, y) = x^3 + y^3 - 3x - 12y + 20$.

(b) $f(x,y,z) = 3x^2 + 2y^2 + 2z^2 + 2xy + 2yz + 2xz$.
(c) $f(x,y,z) = x^2 + y^2 + z^2 - 4xy$.
(d) $f(x,y) = x^4 + y^4 - x^2 - y^2 + 1$.
(e) $f(x,y) = 12x^3 + 36xy - 2y^3 + 9y^2 - 72x + 60y + 5$.

PROBLEM 2.16 (Intermediate). [**13**, p.35 #26] Show that the function

$$f(x,y,z) = e^{x^2+y^2+z^2} - x^4 - y^6 - z^6$$

has a global minimum on $\mathbb{R}^3$.

PROBLEM 2.17 (Intermediate). [**13**, p.36 #33] Consider the function

$$f(x,y) = x^3 + e^{3y} - 3xe^y.$$

Show that f has exactly one critical point, and that this point is a local minimum but not a global minimum.

PROBLEM 2.18 (Basic). Let $f(x,y) = -\log(1-x-y) - \log x - \log y$.
(a) Find the domain D of f.
(b) Prove that D is a convex set.
(c) Prove that f is strictly convex on D.
(d) Find the strict global minimum.

PROBLEM 2.19 (Basic). [**13**, p.81 #27] Find local and global minima in $\mathbb{R}^3$ (if they exist) for the function

$$f(x,y) = e^{x+z-y} + e^{y-x-z}$$

CHAPTER 3

Numerical Approximation for Unconstrained Optimization

Although technically any characterization result finds the exact value of the extrema of a function, computationally this is hardly feasible (specially for functions of very high dimension). See the following session based on problem 2.16 for an example, where we try to find the critical points of the function $f(x,y,z) = e^{x^2+y^2+z^2} - x^4 - y^y - z^6$ symbolically in Python with the `sympy` libraries:

```
# Importing necessary symbols/libraries/functions
from sympy.abc import x,y,z
from sympy import Matrix, solve, exp
from sympy.tensor.array import derive_by_array

# Description of f, computation of its gradient and Hessian
f = exp(x**2 + y**2 + z**2) - x**4 -y**6 - z**6
gradient = derive_by_array(f, [x,y,z])
hessian  = Matrix([derive_by_array(gradient, a) for a in [x,y,z]])
```

While the correct expressions for ∇f and $\mathrm{Hess} f$ are quickly computed, trying to find critical points results in an error:

```
>>> solve(gradient) # Search of critical points by solving ∇f = 0
NotImplementedError: could not solve
4*x**2*sqrt(-log(exp(x**2)/(2*x**2))) - 6*(-log(exp(x**2)/(2*x**2)))**(5/2)
```

Too complex a task to be performed symbolically, although the obvious answer is $(0,0,0)$. A better way to approach this is by trying to approximate this minimum using the structure of the graph of f. In these notes we are going to explore several strategies to accomplish this task, based on the concept of *iterative methods for finding zeros of real-valued functions*.

1. Newton-Raphson's Method

1.1. Newton-Raphson Method to search for roots of univariate functions. In order to find a good estimation of $\sqrt{2}$ with many decimal places, we allow a computer to find better and better approximations of the root of the polynomial $p(x) = x^2 - 2$. We start with an initial guess, say $x_0 = 3$. We construct a sequence $\{x_n\}_{n\in\mathbb{N}}$ that converges to $\sqrt{2}$ as follows:

(a) Find the tangent line to the graph of p at x_0,

$$y = p(x_0) + p'(x_0)(x - x_0)$$

Intuitively, we *substitute* the original function f by the approximation given by the derivative. It is so much easier to seek roots of this simpler function, and that is what we do to *approximate* the roots of p.

(b) Provided the corresponding line is not horizontal ($p'(x_0) \neq 0$), report the intersection of this line with the x–axis. Call this intersection x_1

$$p'(x_0)(x_1 - x_0) = -p(x_0)$$

or equivalently,

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

(c) Repeat this process, to get a sequence

$$x_{n+1} = x_n - \frac{p(x_n)}{p'(x_n)} = x_n - \frac{x_n^2 - 2}{2x_n} = \frac{x_n}{2} - \frac{1}{x_n}.$$

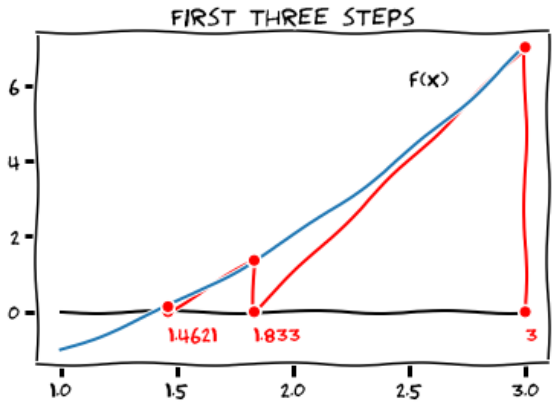


FIGURE 3.1. Newton-Raphson iterative method

Note the result of applying this process a few times:

n	x_n	$p(x_n)$
0	3.000000000000000	$7.0000E+00$
1	1.833333333333333	$1.3611E+00$
2	1.462121212121212	$1.3780E-01$
3	1.414998429894803	$2.2206E-03$
4	1.414213780047198	$6.1568E-07$
5	1.414213562373112	$4.7518E-14$
6	1.414213562373095	$-4.4409E-16$
7	1.414213562373095	$4.4409E-16$

DEFINITION. Given a differentiable real-valued function $f\colon \mathbb{R} \to \mathbb{R}$ and an initial guess $x_0 \in \mathbb{R}$, we define the *Newton-Raphson iteration* to be the sequence $\{x_n\}_{n\in\mathbb{N}}$ given by one of the two following *recursive formulas*

$$\begin{aligned} f'(x_n)(x_{n+1} - x_n) &= -f(x_n) \\ x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \end{aligned} \tag{3}$$

The *Newton-Raphson method* (or *Newton method*) refers to employing this sequence to search and approximate roots of the equation $f(x) = 0$.

EXAMPLE 3.1. Consider now the function $f(x) = 1 - \frac{1}{x}$ over $(0, \infty)$, which has the obvious root $x = 1$. The Newton-Raphson method gives the following iterates for any $x_0 \in (0, \infty)$:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n\big(2 - x_n\big).$$

Notice the two factors in the right-hand side of that expression: x_n, and $2 - x_n$. If the initial guess does not satisfy $0 < x_0 < 2$, then the next iteration gives a non-positive value (see Figure 3.2). The method will not work on those instances: convergence to a solution is not guaranteed.

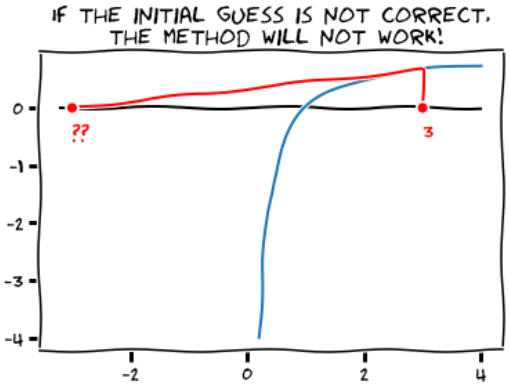


FIGURE 3.2. Initial guess must be carefully chosen in Newton-Raphson

EXAMPLE 3.2. Consider now $f(x) = \operatorname{sign}(x)\sqrt{|x|}$ over $\mathbb{R}$, with root at $x = 0$. The Newton-Raphson method fails miserably with this function: for any $x_0 \neq 0$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = x_0 - \frac{\operatorname{sign}(x_0)|x_0|^{1/2}}{\frac{1}{2}|x_0|^{-1/2}} = -x_0.$$

This sequence turns into a loop: $x_{2n} = x_0$, $x_{2n+1} = -x_0$ for all $n \in \mathbb{N}$ (see Figure 3.3).

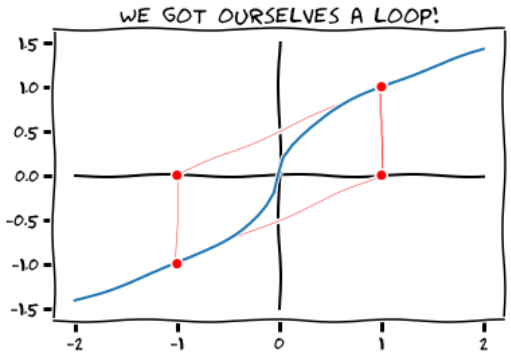


FIGURE 3.3. Newton-Raphson fails for some functions

1.2. Efficiency of Newton-Raphson's Method. To study the error in a Newton-Raphson iteration $\{x_n\}_{n\in\mathbb{N}}$ that converges to a root $x^\star$ of the function f, we observe that

$$\begin{aligned} x_{n+1} - x^\star &= x_n - x^\star - \frac{f(x_n)}{f'(x_n)} \\ &= (x_n - x^\star)\left(1 - \frac{f(x_n) - \overbrace{f(x^\star)}^{0}}{(x_n - x^\star)f'(x_n)}\right) \\ &= (x_n - x^\star)\left(1 - \frac{1}{f'(x_n)}\frac{f(x_n) - f(x^\star)}{x_n - x^\star}\right) \end{aligned} \tag{4}$$

Recall at this point the definition of *divided differences* for any n–times differentiable function $g\colon \mathbb{R} \to \mathbb{R}$, and a family of values $t_0 \le t_1 \le \cdots \le t_n$:

$$\begin{aligned} \Delta g[t_0, t_1] &= \frac{g(t_1) - g(t_0)}{t_1 - t_0} \qquad (\text{if } t_0 \neq t_1) \\ \Delta g[t_0, t_0] &= g'(t_0) \\ \Delta g[t_0, t_1, \ldots, t_n] &= \frac{\Delta g[t_1, t_2, \ldots, t_n] - \Delta g[t_0, t_1, \ldots, t_{n-1}]}{t_n - t_0} \quad (\text{if } t_0 \neq t_n) \\ \Delta g[\underbrace{t_0, \ldots, t_0}_{n+1 \text{ times}}] &= \frac{1}{n!} g^{(n)}(t_0) \qquad (\text{Why? } \textbf{Hint:} \text{ Taylor's Polynomial}) \end{aligned}$$

We can then rewrite (4) in terms of divided differences as follows:

$$\begin{aligned} x_{n+1} - x^\star &= (x_n - x^\star)\left(1 - \frac{\Delta f[x_n, x^\star]}{\Delta f[x_n, x_n]}\right) \\ &= (x_n - x^\star)\frac{\Delta f[x_n, x_n] - \Delta f[x_n, x^\star]}{\Delta f[x_n, x_n]} \\ &= (x_n - x^\star)\frac{\Delta f[x_n, x_n] - \Delta f[x_n, x^\star]}{x_n - x^\star}\frac{x_n - x^\star}{\Delta f[x_n, x_n]} \end{aligned}$$

$$= (x_n - x^\star)^2 \frac{\Delta f[x_n, x_n, x^\star]}{\Delta f[x_n, x_n]}$$

Therefore,

$$\lim_n \frac{x_{n+1} - x^\star}{(x_n - x^\star)^2} = \lim_n \frac{\Delta f[x_n, x_n, x^\star]}{\Delta f[x_n, x_n]} = \frac{f''(x^\star)}{2f'(x^\star)}.$$

If $f''(x^\star) \neq 0$, the Newton-Raphson's iteration exhibits quadratic convergence.[1]

REMARK 3.1. We have just proven that, if a Newton-Raphson iteration for a function f gives a convergent sequence, the convergence is quadratic. But, how can we guarantee convergence to a root of f? The key is in *how far can we start the sequence* given the structure of the graph of f.

THEOREM 3.1 (Local Convergence for the Newton-Raphson Method). *Let $x^\star$ be a simple root of the equation $f(x) = 0$, and assume that there exists $\varepsilon > 0$ so that*

- *f is twice continuously differentiable in the interval $(x^\star - \varepsilon, x^\star + \varepsilon)$, and*
- *there are no critical points of f on that interval.*

Set

$$M(\varepsilon) = \max\left\{ \left| \frac{f''(s)}{2f'(t)} \right| : x^\star - \varepsilon < s, t < x^\star + \varepsilon \right\}.$$

If ε is small enough so that $\varepsilon M(\varepsilon) < 1$, then

(a) There are no other roots of f in $(x^\star - \varepsilon, x^\star + \varepsilon)$.
(b) Any Newton-Raphson iteration starting at an initial guess $x_0 \neq x^\star$ in that interval will converge (quadratically) to $x^\star$

I have seen this Theorem in [**9**] *with the condition $2\varepsilon M(\varepsilon) < 1$ instead, but I could not see why that 2 was necessary. What am I missing?*

PROOF. Start with Taylor's Theorem for f around $x^\star$. Given $x \neq x^\star$ satisfying $|x - x^\star| < \varepsilon$, there exists ξ between x and $x^\star$ so that

$$\begin{aligned} f(x) &= f(x^\star) + (x - x^\star) f'(x^\star) + \tfrac{1}{2}(x - x^\star)^2 f''(\xi) \\ &= (x - x^\star) f'(x^\star) \left(1 + (x - x^\star) \frac{f''(\xi)}{2f'(x^\star)} \right) \end{aligned}$$

Note that the three factors on the last expression are never zero:

$$\begin{aligned} x - x^\star &\neq 0 && \text{(since } x \neq x^\star \text{ by hypothesis)} \\ f'(x^\star) &\neq 0 && \text{(no critical points by hypothesis)} \\ \left| (x - x^\star) \frac{f''(\xi)}{2f'(x^\star)} \right| &\leq \varepsilon M(\varepsilon) < 1 && \text{(by hypothesis on } M(\varepsilon)) \end{aligned}$$

This proves (a).

We want to prove now that all terms of a Newton-Raphson iteration stay in the interval $(x^\star - \varepsilon, x^\star + \varepsilon)$. We do that by induction:

- $|x_0 - x^\star| < \varepsilon$ by hypothesis.

[1] See Appendix A

- Assume $|x_n - x^\star| < \varepsilon$. In that case,

$$\big|x_{n+1} - x^\star\big| = \big|x_n - x^\star\big|^2 \left|\frac{\Delta f[x_n, x_n, x^\star]}{\Delta f[x_n, x_n]}\right|$$

but $\Delta f[x_n, x_n] = f'(x_n)$, and there exist ξ_n between x_n and $x^\star$ so that $\Delta f[x_n, x_n, x^\star] = \frac{1}{2} f''(\xi_n)$; therefore,

$$\big|x_{n+1} - x^\star\big| = \big|x_n - x^\star\big|^2 \left|\frac{f''(\xi)}{2f'(x_n)}\right| \leq \varepsilon^2 M(\varepsilon) = \varepsilon \cdot \varepsilon M(\varepsilon) < \varepsilon.$$

The next step is to prove that there is convergence. A similar computation to the previous gives

$$\big|x_n - x^\star\big| \leq \varepsilon M(\varepsilon)\big|x_{n-1} - x^\star\big| \leq \big(\varepsilon M(\varepsilon)\big)^n \big|x_0 - x^\star\big|.$$

Since $\varepsilon M(\varepsilon) < 1$, $\lim_n \big(\varepsilon M(\varepsilon)\big)^n = 0$, and $\{x_n\}_{n\in\mathbb{N}}$ converges to $x^\star$. □

Example 3.3. Let's use this Theorem to prove convergence of Newton-Raphson iterations for $f(x) = x^2 - 2$ for initial guesses x_0 in a meaningful interval—something like the obvious choice $x_0 \in [1, 2]$. Pick for example $\varepsilon = \sqrt{2}/2$, so we can analyze convergence with initial guesses in the interval $[\sqrt{2}/2, 3\sqrt{2}/2] \supset [1, 2]$. Note that

$$M\big(\tfrac{\sqrt{2}}{2}\big) = \max\left\{\frac{1}{2t} : \frac{\sqrt{2}}{2} < t < \frac{3\sqrt{2}}{2}\right\} = \frac{\sqrt{2}}{2}$$

and thus,

$$\varepsilon M(\varepsilon) = \frac{\sqrt{2}}{2} M\big(\tfrac{\sqrt{2}}{2}\big) = \frac{1}{2} < 1.$$

By virtue of Theorem 3.1 we have quadratic convergence to $\sqrt{2}$ if we start with any initial guess in this interval.

1.3. Extension to higher dimensions. Let's proceed to extend this process to functions $\boldsymbol{g}\colon \mathbb{R}^d \to \mathbb{R}^d$ as follows.

- Any function $\boldsymbol{g}\colon \mathbb{R}^d \to \mathbb{R}^d$ can be described in the form $\boldsymbol{g}(\boldsymbol{x}) = \big[g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_d(\boldsymbol{x})\big]$ for d real-valued functions $g_k\colon \mathbb{R}^d \to \mathbb{R}$ $(1 \leq k \leq d)$.
- For such a function $\boldsymbol{g}$, the *derivative* is the Jacobian

$$\boldsymbol{J}\boldsymbol{g} = \nabla \boldsymbol{g} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_d} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_d}{\partial x_1} & \frac{\partial g_d}{\partial x_2} & \cdots & \frac{\partial g_d}{\partial x_d} \end{bmatrix}$$

Start with a guess for the solution, $\boldsymbol{x}_0$. As we did in the one-dimensional case, we *substitute* the function $\boldsymbol{g}$ by the linear approximation given by the Jacobian:

$$\boldsymbol{y}^\intercal = \boldsymbol{g}(\boldsymbol{x}_0)^\intercal + \nabla \boldsymbol{g}\,(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0)^\intercal.$$

Provided the determinant of the Jacobian $\nabla \boldsymbol{g}$ is not zero, there is only one solution for the equation $\boldsymbol{y} = \boldsymbol{0}$. The computation of $\boldsymbol{x}_1$ is therefore the solution of the following system of linear equations:

$$\nabla \boldsymbol{g}\,(\boldsymbol{x}_0)(\boldsymbol{x}_1 - \boldsymbol{x}_0)^\intercal = -\boldsymbol{g}(\boldsymbol{x}_0)^\intercal,$$

or equivalently,

$$\boldsymbol{x}_1{}^\intercal = \boldsymbol{x}_0{}^\intercal - \big[\,\nabla \boldsymbol{g}(\boldsymbol{x}_0)\big]^{-1} \cdot \boldsymbol{g}(\boldsymbol{x}_0)^\intercal.$$

We iterate this process to obtain a sequence of approximations $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ to a root, via any of the following two recursive formulas:

$$\begin{aligned} \nabla \boldsymbol{g}(\boldsymbol{x}_n)(\boldsymbol{x}_{n+1} - \boldsymbol{x}_n)^\intercal &= -\boldsymbol{g}(\boldsymbol{x}_n)^\intercal, \\ \boldsymbol{x}_{n+1}{}^\intercal &= \boldsymbol{x}_n{}^\intercal - \big[\,\nabla \boldsymbol{g}(\boldsymbol{x}_n)\big]^{-1} \cdot \boldsymbol{g}(\boldsymbol{x}_n)^\intercal, \end{aligned} \tag{5}$$

EXAMPLE 3.4. Consider the function $\boldsymbol{g}\colon \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$\boldsymbol{g}(x, y, z) = \big[x^3 - y, y^3 - x\big]$$

Its Jacobian at each (x, y) is given by

$$\nabla \boldsymbol{g}(x, y) = \begin{bmatrix} 3x^2 & -1 \\ -1 & 3y^2 \end{bmatrix}$$

Note the determinant of this matrix is $\det \nabla \boldsymbol{g}(x, y) = 9x^2y^2 - 1 = (3xy - 1)(3xy + 1)$. For any point (x, y) that does not make this expression zero, this is an invertible matrix with

$$\big[\,\nabla \boldsymbol{g}(x, y)\big]^{-1} = \frac{1}{9x^2y^2 - 1} \begin{bmatrix} 3y^2 & 1 \\ 1 & 3x^2 \end{bmatrix}$$

For an initial guess (x_0, y_0), the sequence computed by this method is then given by

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - \frac{1}{9x_n^2y_n^2 - 1} \begin{bmatrix} 3y_n^2 & 1 \\ 1 & 3x_n^2 \end{bmatrix} \begin{bmatrix} x_n^3 - y_n \\ y_n^3 - x_n \end{bmatrix}$$

Let's run this process with three different initial guesses:

(a) Starting at $(x_0, y_0) = (-1.0, 1.0)$, the sequence converges to $(0, 0)$.

n	x_n	y_n
0	-1.00000000	1.00000000
1	-0.50000000	0.50000000
2	-0.14285714	0.14285714
3	-0.00549451	0.00549451
4	-0.00000033	0.00000033
5	-0.00000000	0.00000000
6	-0.00000000	0.00000000

(b) Starting at $(x_0, y_0) = (3.5, 2.1)$, the sequence converges to $(1, 1)$.

n	x_n	y_n
0	3.50000000	2.10000000
1	2.37631607	1.57961573
2	1.65945969	1.27476534
3	1.23996276	1.10419072
4	1.04837462	1.02274752
5	1.00260153	1.00133122
6	1.00000824	1.00000451
7	1.00000000	1.00000000
8	1.00000000	1.00000000

(c) Starting at $(x_0, y_0) = (-13.5, -7.3)$, the sequence converges to $(-1, -1)$.

n	x_n	y_n
0	-13.50000000	-7.30000000
1	-9.00900415	-4.92301873
2	-6.01982204	-3.36480659
3	-4.03494126	-2.36199873
4	-2.72553474	-1.73750959
5	-1.87830623	-1.36573112
6	-1.36121191	-1.15374930
7	-1.09518303	-1.04341362
8	-1.00932090	-1.00463507
9	-1.00010404	-1.00005571
10	-1.00000001	-1.00000001
11	-1.00000000	-1.00000000
12	-1.00000000	-1.00000000
13	-1.00000000	-1.00000000

REMARK 3.2. The Newton-Raphson's method to solve $\boldsymbol{g} = \mathbf{0}$, as given by the recursive formula with the inverse of the gradient is very convenient to provide explicit descriptions of the different iterations. However, it is hardly suitable for practical purposes, due to the computational issues involving matrix inversion.

To avoid dealing with matrix inversion, we exclusively use the equivalent formula $\nabla \boldsymbol{g}(\boldsymbol{x}_n)\big(\boldsymbol{x}_{n+1} - \boldsymbol{x}_n\big)^\intercal = -\boldsymbol{g}(\boldsymbol{x}_n)^\intercal$, since it offers a simple system of linear equations, which is a much more reliable process, prone to less numerical error.

1.4. Optimization via Newton's Method. We can readily see how this process aids in the computation of critical points of twice continuously differentiable real-valued function $f\colon \mathbb{R}^d \to \mathbb{R}$:

(a) Set $\boldsymbol{g}(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) = \left[\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d}\right]$
(b) It is then $\nabla \boldsymbol{g}(\boldsymbol{x}) = \operatorname{Hess} f(\boldsymbol{x})$
(c) Perform a Newton method (with initial guess $\boldsymbol{x}_0$) on $\boldsymbol{g} = \nabla f$ to obtain the recursive formula

$$\boldsymbol{x}_{n+1}{}^\intercal = \boldsymbol{x}_n{}^\intercal - \big[\operatorname{Hess} f(\boldsymbol{x}_n)\big]^{-1} \cdot \nabla f(\boldsymbol{x}_n)^\intercal \tag{6}$$

EXAMPLE 3.5. Consider the polynomial $p_4(x, y) = x^4 - 4xy + y^4$. Notice $\nabla p_4(x, y) = \big[x^3 - y, y^3 - x\big]$—this is function $\boldsymbol{g}$ in Example 3.4. The critical points we found were $(0, 0)$, $(-1, -1)$ and $(1, 1)$. See Figure 3.4.

EXAMPLE 3.6. A similar process for the Rosenbrock function

$$\mathcal{R}_{1,1}(x, y) = (1 - x)^2 + (y - x^2)^2$$

gives the following recursive formula:

$$\begin{aligned}\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} &= \begin{bmatrix} x_n \\ y_n \end{bmatrix} - \big[\operatorname{Hess} \mathcal{R}_{1,1}(x_n, y_n)\big]^{-1} \cdot \nabla \mathcal{R}_{1,1}(x_n, y_n)^\intercal \\ &= \frac{1}{2x_n^2 - 2y_n + 1} \begin{bmatrix} 2x_n^3 - 2x_n y_n + 1 \\ x_n(2x_n^3 - 2x_n y_n - x_n + 2) \end{bmatrix}\end{aligned}$$

For instance, starting with the initial guess $(x_0, y_0) = (-2, 2)$, the sequence converges to the critical point $(1, 1)$ in a few steps. See Figure 3.4.

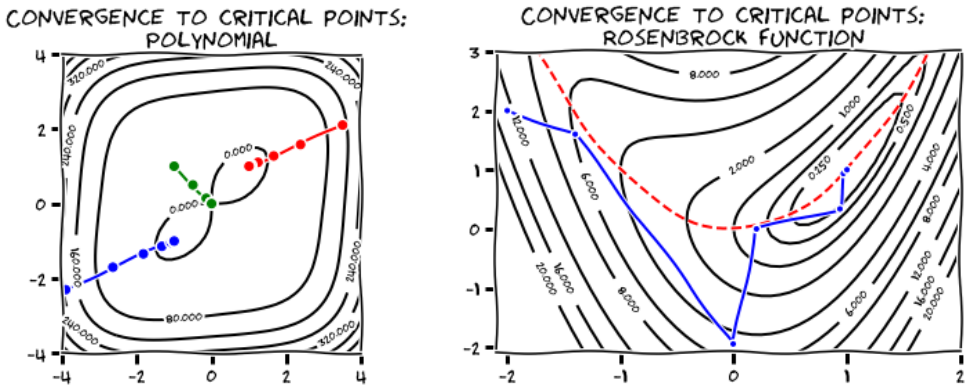


FIGURE 3.4. Newton-Raphson method

REMARK 3.3. The equivalent recursive formula to (6) to search for critical points of a twice continuously differentiable real-valued function $f\colon \mathbb{R}^d \to \mathbb{R}$ is given by

$$\mathrm{Hess} f(\boldsymbol{x}_n)(\boldsymbol{x}_{n+1} - \boldsymbol{x}_n)^\intercal = -\nabla f(\boldsymbol{x}_n)^\intercal. \tag{7}$$

We refer to this formula as the *Newton-Raphson recursive formula*, and the sequence $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ is coined the *Newton-Raphson iteration*. The sequence of directions associated to $\{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n\}_{n\in\mathbb{N}}$ are called *Newton-Raphson directions*.

EXAMPLE 3.7. The equivalent recursive formula to the one we obtained in Example 3.4 is as follows:

$$\begin{bmatrix} 3x_n^2 & -1 \\ -1 & 3y_n^2 \end{bmatrix} \begin{bmatrix} x_{n+1} - x_n \\ y_{n+1} - y_n \end{bmatrix} = \begin{bmatrix} x_n^3 - y_n \\ y_n^3 - x_n \end{bmatrix}$$

All we need to do is, at each step n, solve for X and Y the system of linear equations

$$\begin{cases} 3x_n^2(X - x_n) - (Y - y_n) = x_n^3 - y_n \\ -(X - x_n) + 3y_n^2(Y - y_n) = y_n^2 - x_n \end{cases}$$

or equivalently,

$$\begin{cases} 3x_n^2 X - Y = 4x_n^3 - 2y_n \\ -X + 3y_n^2 Y = 4y_n^2 - 2x_n \end{cases}$$

There are some theoretical results that aid in the search for a *good* initial guess in case of multivariate functions. The following states a simple set of conditions on f and $\boldsymbol{x}_0$ to guarantee *quadratic convergence* of the corresponding sequences $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ to a critical point $\boldsymbol{x}^\star$.

THEOREM 3.2 (Quadratic Convergence Theorem). *Suppose $f\colon \mathbb{R}^d \to \mathbb{R}$ is a twice continuously differentiable real-valued function, and $\boldsymbol{x}^\star$ is a critical point of f. Let $\mathcal{N}(\boldsymbol{x})^\intercal = \boldsymbol{x}^\intercal - \big[\mathrm{Hess} f(\boldsymbol{x})\big]^{-1} \cdot \nabla f(\boldsymbol{x})^\intercal$. If there exists*

(a) $h > 0$ so that[2] $\big\|\big[\mathrm{Hess} f(\boldsymbol{x}^\star)\big]^{-1}\big\| \leq \frac{1}{h}$,
(b) $\beta > 0$, $L > 0$ for which $\|\mathrm{Hess} f(\boldsymbol{x}) - \mathrm{Hess} f(\boldsymbol{x}^\star)\| \leq L\|\boldsymbol{x} - \boldsymbol{x}^\star\|$ provided $\|\boldsymbol{x} - \boldsymbol{x}^\star\| \leq \beta$.

In that case, for all $\boldsymbol{x} \in \mathbb{R}^d$ satisfying $\|\boldsymbol{x} - \boldsymbol{x}^\star\| \leq \min\{\beta, \frac{2h}{3L}\}$,

$$\frac{\|\mathcal{N}(\boldsymbol{x}) - \boldsymbol{x}^\star\|}{\|\boldsymbol{x} - \boldsymbol{x}^\star\|^2} \leq \frac{3L}{2h}$$

EXAMPLE 3.8. There is an implementation of Newton-Raphson method in the `scipy.optimize` libraries in Python: the routine `fmin_ncg()` (the `cg` indicates that the inversion of the Hessian is performed using the technique of conjugate gradient). The following session illustrates how to use this

[2]Recall the *norm* of a matrix M, defined by $\|M\| = \max\{\|M \cdot \boldsymbol{x}\| : \|\boldsymbol{x}\| = 1\}$.

method to approximate the minimum of the Rosenbrock function $\mathcal{R}_{1,100}$ with an initial guess $(x_0, y_0) = (2, 2)$

```
import numpy as np, matplotlib.pyplot as plt
from scipy.optimize import fmin_ncg

# Rosenbrock R_{1,100} function and its Jacobian ∇R_{1,100}
from scipy.optimize import rosen, rosen_der
```

We call `fmin_ncf` with the function and its gradient (with the option `fprime=`), and with the initial guess $(2, 2)$.

```
>>> fmin_ncg(rosen, [2.,2.], fprime=rosen_der, retall=False)
Optimization terminated successfully.
         Current function value: 0.000000
         Iterations: 277
         Function evaluations: 300
         Gradient evaluations: 1540
         Hessian evaluations: 0
array([ 0.99999993,  0.99999987])
```

2. Secant Methods

The Newton-Raphson method to compute local extrema of real-valued functions $f\colon \mathbb{R}^d \to \mathbb{R}$, has several shortcomings:

- Convergence is not guaranteed.
- Even when convergence is guaranteed, the limit of a Newton-Raphson iteration is not necessarily a local minimum. Any critical point of f is a target.
- For a successful implementation of Newton-Raphson we do need expressions for the function itself, its gradient and Hessian matrix.

The *secant method* takes care of the the latter issue, while retaining the same convergence rates.

2.1. A secant method to search for roots of univariate functions. To explain how it works, let's once again try to find an accurate value of $\sqrt{2}$ as the root of the polynomial $p(x) = x^2 - 2$.

(a) Consider two initial guesses $x_0 = 3$, $x_1 = 2.8$. Notice $f(3) = 7 \neq 5.84 = f(2.8)$.

(b) The line that joins the points $(3, 7)$ and $(2, 8, 5.84)$ has equation

$$y - 7 = \frac{5.84 - 7}{2.8 - 3}(x - 3),$$
$$y = 5.8x - 10.4.$$

The latter can be seen as a linear approximation to the original function—one that did not use the derivative of f. This linear function intersects the x–axis at

$$x_2 = \frac{10.4}{5.8} \approx 1.7931034483$$

We take this value as an approximation to the root of $f(x) = 0$.

(c) Repeat this process to get a sequence of approximations x_n.

EXAMPLE 3.9. Observe the result of applying this recursive process, and compare with the similar experiment we conducted using the Newton-Raphson method in page 30.

n	x_n	$f(x_n)$
0	3.000000000000000	$7.0000E+00$
1	2.800000000000000	$5.8400E+00$
2	1.793103448275862	$1.2152E+00$
3	1.528528528528528	$3.3640E-01$
4	1.427253172054743	$3.7052E-02$
5	1.414717869757887	$1.4267E-03$
6	1.414215876250105	$6.5446E-06$
7	1.414213562785585	$1.1667E-09$
8	1.414213562373095	$8.8818E-16$

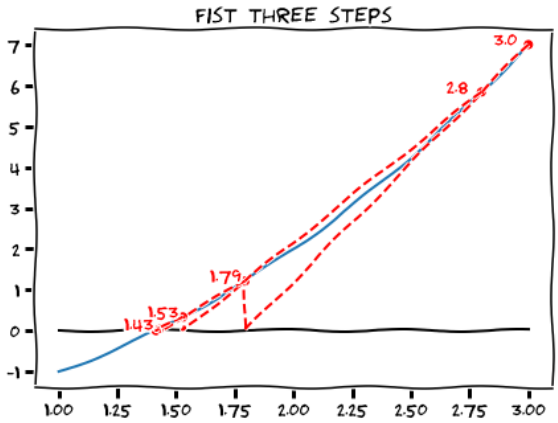


FIGURE 3.5. Secant iterative method

DEFINITION. Given a function $f : \mathbb{R} \to \mathbb{R}$ and two initial guesses $x_0 \neq x_1$ satisfying $f(x_0) \neq f(x_1)$, we define the *Secant method iteration* to be the sequence $\{x_n\}_{n\in\mathbb{N}}$ obtained by one of the following recursive formulas:

$$\begin{aligned} \left[\frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}\right](x_{n+1} - x_n) &= -f(x_n) \\ x_{n+1} = x_n - \left[\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\right] f(x_n) & \end{aligned} \tag{8}$$

The *Secant method* refers to employing this sequence to search and approximate roots of the equation $f(x) = 0$.

2.2. Efficiency of the Secant Method. Let $f\colon \mathbb{R} \to \mathbb{R}$ be a real-valued function with a root $x^\star$, and let $\{x_n\}_{n\in\mathbb{N}}$ be a secant method iteration.

$$\begin{aligned}
x_{n+1} - x^\star &= x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) - x^\star \\
&= (x_n - x^\star)\left(1 - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \frac{f(x_n) - \overbrace{f(x^\star)}^{0}}{x_n - x^\star}\right) \\
&= (x_n - x^\star)\left(1 - \frac{\Delta f[x_n, x^\star]}{\Delta f[x_{n-1}, x_n]}\right) \\
&= (x_n - x^\star)\frac{\Delta f[x_{n-1}, x_n] - \Delta f[x_n, x^\star]}{\Delta f[x_{n-1}, x_n]} \\
&= (x_n - x^\star)\frac{\Delta f[x_{n-1}, x_n] - \Delta f[x_n, x^\star]}{x_{n-1} - x^\star} \frac{x_{n-1} - x^\star}{\Delta f[x_{n-1}, x_n]} \\
&= (x_n - x^\star)(x_{n-1} - x^\star)\frac{\Delta f[x_{n-1}, x_n, x^\star]}{\Delta f[x_{n-1}, x_n]}
\end{aligned}$$

We use this identity to prove the corresponding *local convergence result*, as we did for the Newton-Raphson method:

THEOREM 3.3 (Local Convergence for the Secant Method). *Let $x^\star$ be a simple root of the equation $f(x) = 0$, and there exists $\varepsilon > 0$ so that*

- *f is twice continuously differentiable in the interval $(x^\star - \varepsilon, x^\star + \varepsilon)$, and*
- *there are no critical points of f on that interval.*

Set

$$M(\varepsilon) = \max\left\{\left|\frac{f''(s)}{2f'(t)}\right| : x^\star - \varepsilon < s, t < x^\star + \varepsilon\right\}.$$

If ε is small enough so that $\varepsilon M(\varepsilon) < 1$, then

(a) There are no other roots of f in $(x^\star - \varepsilon, x^\star + \varepsilon)$.
(b) Any Secant method iteration starting with initial guesses $x_0 \neq x_1$ (none of them equal to $x^\star$, and with $f(x_0) \neq f(x_1)$) in that interval will converge (quadratically) to $x^\star$

The proof is similar to the one for Theorem 3.1, and is left as exercise.

2.3. Generalization of the Secant Method to higher dimensions: Broyden's Method. The objective now is the search of a root for a function $\boldsymbol{g}\colon \mathbb{R}^d \to \mathbb{R}^d$, which we assume of the form

$$\boldsymbol{g}(x_1, \ldots, x_d) = \big[g_1(x_1, \ldots, x_d), \ldots, g_d(x_1, \ldots, x_d)\big]$$

for appropriate functions $g_k\colon \mathbb{R}^d \to \mathbb{R}$ $(1 \le k \le d)$. We start at an initial guess $\boldsymbol{x}_0 \in \mathbb{R}^d$ and, if possible, we choose as $\boldsymbol{x}_1$ the first step of Newton-Raphson:

$$\boldsymbol{x}_1{}^\intercal = \boldsymbol{x}_0{}^\intercal - \big[\,\nabla \boldsymbol{g}(\boldsymbol{x}_0)\big]^{-1} \cdot \boldsymbol{g}(\boldsymbol{x}_0)^\intercal.$$

At that point we craft a linear approximation $\boldsymbol{L}_1 \colon \mathbb{R}^d \to \mathbb{R}^d$ to the graph of $\boldsymbol{g}$ that interpolates $\big(\boldsymbol{x}_0, \boldsymbol{g}(\boldsymbol{x}_0)\big)$ and $\big(\boldsymbol{x}_1, \boldsymbol{g}(\boldsymbol{x}_1)\big)$. We may impose this *secant property* by requesting $\boldsymbol{L}_1$ to be of the form

$$\boldsymbol{L}_1(\boldsymbol{x})^\intercal = \boldsymbol{g}(\boldsymbol{x}_1)^\intercal + \boldsymbol{A}_1 \cdot (\boldsymbol{x} - \boldsymbol{x}_1)^\intercal,$$

where $\boldsymbol{A}_1$ is a non-singular square matrix of size $d \times d$ that satisfies

$$\boldsymbol{A}_1 \cdot (\boldsymbol{x}_1 - \boldsymbol{x}_0)^\intercal = \boldsymbol{g}(\boldsymbol{x}_1)^\intercal - \boldsymbol{g}(\boldsymbol{x}_0)^\intercal.$$

At this point, we approximate the root of $\boldsymbol{g}$ by the root $\boldsymbol{x}_2$ of $\boldsymbol{L}_1$:

$$\boldsymbol{A}_1 \cdot (\boldsymbol{x}_2 - \boldsymbol{x}_1)^\intercal = -\boldsymbol{g}(\boldsymbol{x}_1)^\intercal,$$

or equivalently,

$$\boldsymbol{x}_2{}^\intercal = \boldsymbol{x}_1{}^\intercal - \boldsymbol{A}_1^{-1} \cdot \boldsymbol{g}(\boldsymbol{x}_1)^\intercal.$$

We repeat this process to obtain a sequence of approximations $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ via a sequence of non-singular square matrices $\boldsymbol{A}_n$ of size $d \times d$ that satisfy the secant property

$$\boldsymbol{A}_n \cdot (\boldsymbol{x}_{n+1} - \boldsymbol{x}_n)^\intercal = \boldsymbol{g}(\boldsymbol{x}_{n+1})^\intercal - \boldsymbol{g}(\boldsymbol{x}_n)^\intercal. \tag{9}$$

The corresponding linear functions are given by

$$\boldsymbol{L}_n(\boldsymbol{x})^\intercal = \boldsymbol{g}(\boldsymbol{x}_n)^\intercal + \boldsymbol{A}_n \cdot (\boldsymbol{x} - \boldsymbol{x}_n)^\intercal.$$

The recurrence formula can then be expressed in any of the following two ways:

$$\begin{aligned} \boldsymbol{A}_n \cdot (\boldsymbol{x}_{n+1} - \boldsymbol{x}_n)^\intercal &= -\boldsymbol{g}(\boldsymbol{x}_n)^\intercal, \\ \boldsymbol{x}_{n+1}{}^\intercal &= \boldsymbol{x}_n{}^\intercal - \boldsymbol{A}_n^{-1} \cdot \boldsymbol{g}(\boldsymbol{x}_n)^\intercal \end{aligned} \tag{10}$$

We refer to this process as the *Broyden Method* for finding roots.

REMARK 3.4. There is not a unique way to choose the matrices $\boldsymbol{A}_n$ in Broyden's method (and I encourage you to find one of your own devise). A straightforward construction is given by the following recursive formulas:

$$\boldsymbol{A}_0 = \nabla \boldsymbol{g}(\boldsymbol{x}_0)$$
$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n + \frac{\big[\big(\boldsymbol{g}(\boldsymbol{x}_{n+1}) - \boldsymbol{g}(\boldsymbol{x}_n)\big)^\intercal - \boldsymbol{A}_n(\boldsymbol{x}_{n+1} - \boldsymbol{x}_n)^\intercal\big] \otimes (\boldsymbol{x}_{n+1} - \boldsymbol{x}_n)}{\|\boldsymbol{x}_{n+1} - \boldsymbol{x}_n\|^2} \tag{11}$$

REMARK 3.5. There are other possibilities for the selection of the point $\boldsymbol{x}_1$ that do not involve using the Jacobian $\nabla \boldsymbol{g}(\boldsymbol{x}_0)$ as the matrix $\boldsymbol{A}_0$. For the purposes of root searching this choice is not too crucial but, as we shall see in the context of optimization, having the initial matrix $\boldsymbol{A}_0$ satisfy extra properties could be very advantageous. To learn about different strategies in this regard, see e.g. [**5**, chapter 8].

EXAMPLE 3.10. Let's use this method to find any of the roots $(-1,-1)$, $(0,0)$, $(1,1)$ of the function $\boldsymbol{g}(x,y) = \big[x^3 - y, y^3 - x\big]$ from Example 3.4. Starting with initial guess $(x_0, y_0) = (-1, 1)$, we take the first step from Newton-Raphson:

$$\nabla \boldsymbol{g}(x_0,y_0) = \begin{bmatrix} 3x_0^2 & -1 \\ -1 & 3y_0^2 \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$$

$$\boldsymbol{L}_0(x,y)^\intercal = \begin{bmatrix} -2 \\ 2 \end{bmatrix} + \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3x - y - 2 \\ -x + 3y + 2 \end{bmatrix}$$

This gives $(x_1, y_1) = (-1/2, 1/2)$ as root of $\boldsymbol{L}_0$. For the second step, we compute the matrix $\boldsymbol{A}_1$ satisfying the secant property:

$$\begin{aligned}
\boldsymbol{A}_1 &= \boldsymbol{A}_0 + \frac{\big[\big(\boldsymbol{g}(-1/2,1/2) - \boldsymbol{g}(-1,1)\big)^\intercal - \boldsymbol{A}_0[\tfrac{1}{2},-\tfrac{1}{2}]^\intercal\big] \otimes [\tfrac{1}{2},-\tfrac{1}{2}]}{\|[\tfrac{1}{2},-\tfrac{1}{2}]\|^2} \\
&= \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} + \frac{\left(\begin{bmatrix} -\frac{5}{8} - (-2) \\ \frac{5}{8} - 2 \end{bmatrix} - \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} \right) \otimes \big[\tfrac{1}{2},-\tfrac{1}{2}\big]}{1/2} \\
&= \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} + 2\big(\big[-\tfrac{5}{8},\tfrac{5}{8}\big] \otimes \big[\tfrac{1}{2},-\tfrac{1}{2}\big]\big) \\
&= \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} + 2 \begin{bmatrix} -\frac{5}{8}\cdot\frac{1}{2} & -\frac{5}{8}\cdot\big(-\frac{1}{2}\big) \\ \frac{5}{8}\cdot\frac{1}{2} & \frac{5}{8}\cdot\big(-\frac{1}{2}\big) \end{bmatrix} \\
&= \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} + \begin{bmatrix} -5/8 & 5/8 \\ 5/8 & -5/8 \end{bmatrix} = \begin{bmatrix} 19/8 & -3/8 \\ -3/8 & 19/8 \end{bmatrix}
\end{aligned}$$

This gives a linear approximation

$$\boldsymbol{L}_1(x,y)^\intercal = \begin{bmatrix} -5/8 \\ 5/8 \end{bmatrix} + \begin{bmatrix} 19/8 & -3/8 \\ -3/8 & 19/8 \end{bmatrix} \cdot \begin{bmatrix} x + 1/2 \\ y - 1/2 \end{bmatrix} = \begin{bmatrix} \frac{19}{8}x - \frac{3}{8}x + \frac{3}{4} \\ \\ -\frac{3}{8}x + \frac{19}{8}y - \frac{3}{4} \end{bmatrix}$$

which has root $(x_2, y_2) = (-3/11, 3/11)$.

If we continue the computations, we arrive to the root $(0,0)$ to an accuracy of 6 decimal places in just 5 steps!

n	x_n	y_n	$f(x_n,\, y_n)$
1	-1.000000	1.000000	$\big(-2.000000,\, 2.000000\big)$
2	-0.500000	0.500000	$\big(-0.625000,\, 0.625000\big)$
2	-0.272727	0.272727	$\big(-0.293013,\, 0.293013\big)$
3	-0.072136	0.072136	$\big(-0.072511,\, 0.072511\big)$
4	-0.006172	0.006172	$\big(-0.006172,\, 0.006172\big)$
5	-0.000035	0.000035	$\big(-0.000035,\, 0.000035\big)$
6	-0.000000	0.000000	$\big(-0.000000,\, 0.000000\big)$

2.4. Secant Methods for Optimization. Given a real-valued function $f\colon \mathbb{R}^d \to \mathbb{R}$, we may apply Broyden's method to search for roots of the gradient $\nabla f\colon \mathbb{R}^d \to \mathbb{R}^d$. As it happened with Newton's method, we are not guaranteed convergence to a local minimum.

EXAMPLE 3.11. As we did in Example 3.5, we may explore the convergence to different critical points of the polynomial $p_4(x, y) = x^4 - 4xy + y^4$ using different initial guesses. Note once again that the gradient of p_4 is the function $\nabla p_4 = \boldsymbol{g}$ from Example 3.10:

For instance, starting with $(x_0, y_0) = (-1, 1)$ we converge to the critical point $(0, 0)$, which is not a local minimum.

3. The Method of Steepest Descent

The method of *Steepest Descent* (also known as the method of *Gradient Descent*) is based upon the following property of gradients that we learned in Vector Calculus:

THEOREM 3.4. *If $f\colon \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable, then at any point $\boldsymbol{x} \in \mathbb{R}^d$, the vector $-\nabla f(\boldsymbol{x})$ points in the direction of most rapid decrease for f at $\boldsymbol{x}$. The rate of decrease of f at $\boldsymbol{x}$ in this direction is precisely $-\|\nabla f(\boldsymbol{x})\|$.*

REMARK 3.6. For this reason, the vector $-\nabla f(\boldsymbol{x})/\|\nabla f(\boldsymbol{x})\|$ is called the *direction of steepest descent* of f at $\boldsymbol{x}$.

In order to search for a local minimum for a twice continuously differentiable function $f\colon \mathbb{R}^d \to \mathbb{R}$, we start by choosing an initial guess $\boldsymbol{x}_0$.

(a) Restrict the function f over the line through $\boldsymbol{x}_0$ in the direction of $-\nabla f(\boldsymbol{x}_0)$:

$$\varphi_0(t) = f\big(\boldsymbol{x}_0 - t\,\nabla f(\boldsymbol{x}_0)\big), \quad t \geq 0$$

(b) *Line-search*: Search for the value of $t_0 \geq 0$ that minimizes φ_0, and set

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - t_0\,\nabla f(\boldsymbol{x}_0)$$

(c) Repeat this process to get the sequence

$$\begin{aligned} &\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - t_n\,\nabla f(\boldsymbol{x}_n), \\ &t_n = \operatorname*{argmin}_{t\geq 0} \varphi_n(t) = \operatorname*{argmin}_{t\geq 0} f\big(\boldsymbol{x}_n - t\,\nabla f(\boldsymbol{x}_n)\big) \end{aligned} \tag{12}$$

REMARK 3.7. Sequences constructed following the formula in (12) are said to be *sequences of Steepest Descent* for f.

Unlike Newton-Raphson or the secant methods, this algorithm guarantees that these sequences are non-increasing: $f(\boldsymbol{x}_{n+1}) \leq f(\boldsymbol{x}_n)$ for all $n \in \mathbb{N}$. And even better: if there is convergence, their limit must be a critical point of f. These results are formalized in Theorems 3.5 and 3.6 below.

Steepest descent sequences have another interesting property: on each step n, the direction of steepest descent is perpendicular to the direction of steepest descent of step $n+1$ (!!) We state and prove this result in Theorem 3.7.

THEOREM 3.5. *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a continuously differentiable real-valued function, and let $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ be a sequence of steepest descent for f. If $\nabla f(\boldsymbol{x}_N) \neq 0$, then $f(\boldsymbol{x}_{N+1}) < f(\boldsymbol{x}_N)$.*

THEOREM 3.6. *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a real-valued function, let $\boldsymbol{x}_0 \in \mathbb{R}^d$ be an initial guess. Assume $S = \{\boldsymbol{x} \in \mathbb{R}^d : f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$ is a compact set and f is continuously differentiable in S. Under these conditions, the limit of any convergent subsequence of the associated sequence of steepest descent $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ is a critical point of f.*

THEOREM 3.7. *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a continuously differentiable real-valued function, and $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ a sequence of steepest descent for f. For any $n \in \mathbb{N}$, $\langle \boldsymbol{x}_{n+2} - \boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1} - \boldsymbol{x}_n\rangle = 0$.*

PROOF. Consider for each $n \in \mathbb{N}$ the function $\varphi_n(t) = f\big(\boldsymbol{x}_n - t\,\nabla f(\boldsymbol{x}_x)\big)$, with a global minimum at $t_n \geq 0$. It must then be

$$0 = \varphi_n'(t_n) = \langle \nabla f(\boldsymbol{x}_n), -\nabla f\left(\boldsymbol{x}_n - t_n\,\nabla f(\boldsymbol{x}_n)\right)\rangle = -\langle \nabla f(\boldsymbol{x}_{n+1}), \nabla f(\boldsymbol{x}_n)\rangle,$$

which proves that the gradient of consecutive terms of the sequence of steepest descent for f are perpendicular. Now, by virtue of the recurrence formula (12),

$$\begin{aligned}\langle \boldsymbol{x}_{n+2} - \boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1} - \boldsymbol{x}_n\rangle &= \langle t_{n+1}\,\nabla f(\boldsymbol{x}_{n+1}), t_n\,\nabla f(\boldsymbol{x}_n)\rangle \\ &= t_{n+1}t_n\langle \nabla f(\boldsymbol{x}_{n+1}), \nabla f(\boldsymbol{x}_n)\rangle = 0,\end{aligned}$$

which proves the statement. □

EXAMPLE 3.12. For the polynomial function $p_4(x, y) = x^4 - 4xy + y^4$ from Example 3.5, using the same initial guesses as in Example 3.4, we find the following behavior:

- Starting at $(x_0, y_0) = (-1.0, 1.0)$, the sequence jumps to $(0, 0)$ in one step. At that point, since the gradient of the function is zero, the method of Steepest descent ceases to work.

n	x_n	y_n	$f(x_n, y_n)$
0	−1.000000	1.000000	6.000000
1	0.000000	0.000000	0.000000
2	`nan`	`nan`	`nan`

- Starting at $(x_0, y_0) = (3.5, 2.1)$, the sequence converges to $(1, 1)$.

n	x_n	y_n	$f(x_n, y_n)$
0	3.500000	2.100000	140.110600
1	1.044472	1.753064	3.310777
2	1.141931	1.063276	−1.878163
3	1.008581	1.044435	−1.988879
4	1.013966	1.006319	−1.998931
5	1.000898	1.004472	−1.999891
6	1.001437	1.000651	−1.999989
7	1.000093	1.000461	−1.999999
8	1.000149	1.000067	−2.000000
9	1.000010	1.000048	−2.000000
10	1.000015	1.000007	−2.000000
11	1.000001	1.000005	−2.000000
12	1.000002	1.000001	−2.000000
13	1.000000	1.000001	−2.000000
14	1.000000	1.000000	−2.000000
15	1.000000	1.000000	−2.000000

- Starting at $(x_0, y_0) = (-13.5, -7.3)$, the sequence converges to $(1, 1)$ as well.

n	x_n	y_n	$f(x_n, y_n)$
0	−13.500000	−7.300000	35660.686600
1	2.362722	−4.871733	640.498302
2	1.434154	1.194162	−0.586492
3	1.021502	1.130993	−1.896212
4	1.038817	1.017881	−1.991558
5	1.002305	1.012291	−1.999167
6	1.003909	1.001808	−1.999917
7	1.000236	1.001246	−1.999992
8	1.000399	1.000185	−1.999999
9	1.000024	1.000127	−2.000000
10	1.000041	1.000019	−2.000000
11	1.000002	1.000013	−2.000000
12	1.000004	1.000002	−2.000000
13	1.000000	1.000001	−2.000000
14	1.000000	1.000000	−2.000000
15	1.000000	1.000000	−2.000000

EXAMPLE 3.13. Notice what happens when we try to implement the same process on the Rosenbrock function $\mathcal{R}_{1,1}(x, y) = (1 - x)^2 + (y - x^2)^2$, with the initial guess $(x_0, y_0) = (-2, 2)$. The sequence does converge to the minimum $(1, 1)$, albeit very slowly.

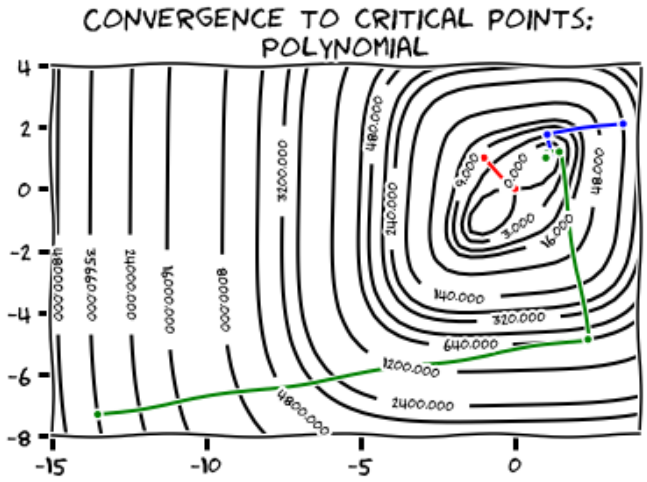


FIGURE 3.6. The Method of Steepest Descent: Polynomial function

n	x_n	y_n	$f(x_n, y_n)$	n	x_n	y_n	$f(x_n, y_n)$
0	−2.000000	2.000000	13.000000	17	0.916394	0.789239	0.009544
1	−0.166290	2.309522	6.567163	18	0.911201	0.818326	0.008028
2	0.256054	−0.056128	0.568264	19	0.929317	0.821560	0.006766
3	0.613477	0.007683	0.285318	20	0.925024	0.845608	0.005723
4	0.568566	0.259241	0.190235	21	0.939976	0.848277	0.004847
5	0.715784	0.285524	0.132227	22	0.936397	0.868329	0.004118
6	0.689755	0.431319	0.098227	23	0.948845	0.870551	0.003502
7	0.779264	0.447299	0.074310	24	0.945840	0.887385	0.002986
8	0.761554	0.546496	0.057977	25	0.956276	0.889248	0.002548
9	0.823325	0.557524	0.045696	26	0.953739	0.903457	0.002178
10	0.810322	0.630358	0.036667	27	0.962537	0.905028	0.001864
11	0.855862	0.638488	0.029614	28	0.960386	0.917075	0.001597
12	0.845883	0.694385	0.024199	29	0.967837	0.918405	0.001369
13	0.880846	0.700627	0.019862	30	0.966007	0.928657	0.001176
14	0.872964	0.744776	0.016437	31	0.972342	0.929788	0.001010
15	0.900551	0.749702	0.013647	32	0.970780	0.938539	0.000869
16	0.894200	0.785276	0.011399	33	0.976182	0.939503	0.000748

3.1. Efficiency of Steepest Descent Method. The analysis of efficiency of the method of Steepest descent is quite involved, but it boils down to studying the efficiency of Steepest descent for quadratic functions—since any function can be approximated using a Taylor's polynomial of degree two. We will study that easier case in these notes.

THEOREM 3.8 (Taylor's Formula). *If $f\colon \mathbb{R}^d \to \mathbb{R}$ is a real-valued function of d variables with continuous first and second partial derivatives on $\mathbb{R}^d$, then*

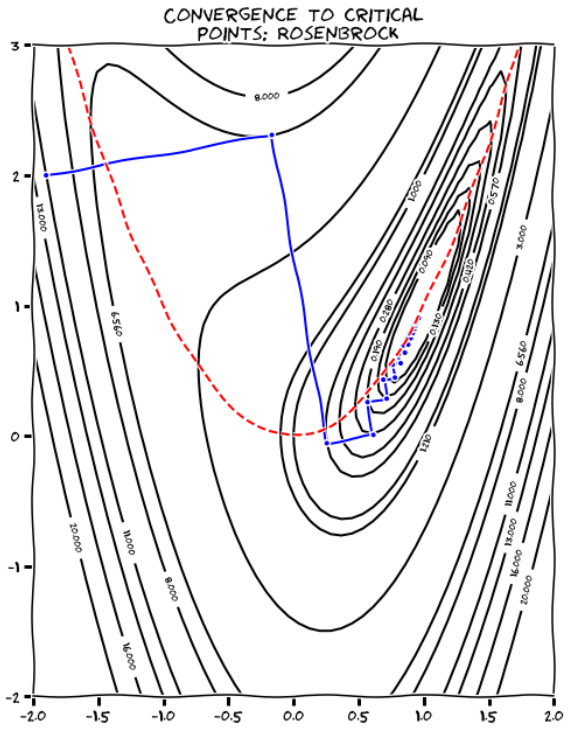


FIGURE 3.7. The Method of Steepest Descent: Rosenbrock Function

for any choice $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$*, there exists a point* $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x}, \boldsymbol{y})$ *in the segment joining* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *so that*

$$f(\boldsymbol{x}) = f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle + \tfrac{1}{2} \mathcal{Q}_{\mathbf{Hess} f(\boldsymbol{\xi})}(\boldsymbol{x} - \boldsymbol{y})$$

Assume we have a quadratic function $p\colon \mathbb{R}^d \to \mathbb{R}$ satisfying $p(\mathbf{0}) = 0$. There exist a d-dimensional vector $D = \big[q_1, \dots, q_d\big]$ and a symmetric matrix $Q = \big[q_{jk}\big]_{j,k=1}^{d}$ (with $q_{jk} = q_{kj}$ for all $1 \le j, k \le d$) so that

$$p(\boldsymbol{x}) = \langle D, \boldsymbol{x} \rangle + \tfrac{1}{2} \mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{x}) = \sum_{k=1}^{d} \big(\tfrac{1}{2} q_{kk} x_k^2 + q_k x_k \big) + \sum_{1 \le j < k \le d} q_{jk} x_j x_k$$

The gradient of this function is thus $\nabla p(\boldsymbol{x}) = x \cdot Q + D$. It has one unique critical point $\boldsymbol{x}^\star = -DQ^{-1}$ (Why?). At that point, it is

$$\begin{aligned} p(\boldsymbol{x}^\star) &= \tfrac{1}{2}\big(- DQ^{-1}\big) Q \big(- DQ^{-1}\big)^{\mathsf{T}} + D\big(- DQ^{-1}\big)^{\mathsf{T}} \\ &= \tfrac{1}{2} DQ^{-1} D^{\mathsf{T}} - DQ^{-1} D^{\mathsf{T}} \\ &= -\tfrac{1}{2} DQ^{-1} D^{\mathsf{T}} = -\tfrac{1}{2} \mathcal{Q}_{(\boldsymbol{Q^{-1}})}(D). \end{aligned} \tag{13}$$

If $\boldsymbol{x}_n$ is a term in a sequence of steepest descent, then to compute $\boldsymbol{x}_{n+1}$ we proceed as follows:

(a) The direction of steepest descent at $\boldsymbol{x}_n$ is

$$\boldsymbol{v}_n = -\nabla p(\boldsymbol{x}_n) = -(x_n Q + D).$$

(b) The restriction $\varphi\colon (0,\infty) \to \mathbb{R}$ of the quadratic function p over the half-line through $\boldsymbol{x}_n$ in the direction $\boldsymbol{v}_n$ is given by

$$\begin{aligned}
\varphi(t) &= p(\boldsymbol{x}_n + t\boldsymbol{v}_n) \\
&= \tfrac{1}{2}(\boldsymbol{x}_n + t\boldsymbol{v}_n)Q(\boldsymbol{x}_n + t\boldsymbol{v}_n)^\intercal + D(\boldsymbol{x}_n + t\boldsymbol{v}_n)^\intercal \\
&= \tfrac{1}{2}\boldsymbol{x}_n Q(\boldsymbol{x}_n + t\boldsymbol{v}_n)^\intercal + \tfrac{1}{2}t\boldsymbol{v}_n Q(\boldsymbol{x}_n + t\boldsymbol{v}_n)^\intercal \\
&\quad + D\boldsymbol{x}_n^\intercal + tD\boldsymbol{v}_n^\intercal \\
&= \tfrac{1}{2}\boldsymbol{x}_n Q\boldsymbol{x}_n^\intercal + \tfrac{1}{2}t\boldsymbol{x}_n Q\boldsymbol{v}_n^\intercal + \tfrac{1}{2}t\boldsymbol{v}_n Q\boldsymbol{x}_n^\intercal + \tfrac{1}{2}t^2\boldsymbol{v}_n Q\boldsymbol{v}_n^\intercal \\
&\quad + D\boldsymbol{x}_n^\intercal + tD\boldsymbol{v}_n^\intercal \\
&= \tfrac{1}{2}\underbrace{\boldsymbol{v}_n Q\boldsymbol{v}_n^\intercal}_{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)} t^2 + \underbrace{\tfrac{1}{2}\boldsymbol{x}_n Q\boldsymbol{x}_n^\intercal + D\boldsymbol{x}_n^\intercal}_{p(\boldsymbol{x}_n)} + tD\boldsymbol{v}_n^\intercal + t\boldsymbol{x}_n Q\boldsymbol{v}_n^\intercal \\
&= \tfrac{1}{2}\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)t^2 + p(\boldsymbol{x}_n) + t\underbrace{\big(x_n Q + D\big)}_{-\boldsymbol{v}_n}\boldsymbol{v}_n^\intercal \\
&= \tfrac{1}{2}t^2\boldsymbol{v}_n Q\boldsymbol{v}_n^\intercal - t\boldsymbol{v}_n\boldsymbol{v}_n^\intercal + p(\boldsymbol{x}_n) \\
&= \tfrac{1}{2}\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)t^2 - \|\boldsymbol{v}_n\|^2 t + p(\boldsymbol{x}_n)
\end{aligned}$$

(c) The restriction function has its global minimum at

$$t_n = \frac{\|\boldsymbol{v}_n\|^2}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)};$$

therefore, the next iteration occurs at

$$x_{n+1} = x_n + t_n\boldsymbol{v}_n = x_n + \frac{\|\boldsymbol{v}_n\|^2}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)}\boldsymbol{v}_n$$

We want to observe the convergence behavior of the sequence of evaluations $\{p(\boldsymbol{x}_n)\}_{n\in\mathbb{N}}$ to $p(\boldsymbol{x}^\star)$. We have

$$\begin{aligned}
p(\boldsymbol{x}_{n+1}) &= p\big(\boldsymbol{x}_n + \tfrac{\|\boldsymbol{v}_n\|^2}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)}\boldsymbol{v}_n\big) \\
&= \tfrac{1}{2}\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)\big(\tfrac{\|\boldsymbol{v}_n\|^2}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)}\big)^2 - \|\boldsymbol{v}_n\|^2\tfrac{\|\boldsymbol{v}_n\|^2}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)} + p(\boldsymbol{x}_n) \\
&= p(\boldsymbol{x}_n) - \frac{\|\boldsymbol{v}_n\|^4}{2\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)};
\end{aligned}$$

therefore,

$$\frac{p(\boldsymbol{x}_{n+1}) - p(\boldsymbol{x}^\star)}{p(\boldsymbol{x}_n) - p(\boldsymbol{x}^\star)} = \frac{p(\boldsymbol{x}_n) - p(\boldsymbol{x}^\star) - \frac{\|\boldsymbol{v}_n\|^4}{2\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)}}{p(\boldsymbol{x}_n) - p(\boldsymbol{x}^\star)}$$

$$
\begin{aligned}
&= 1 - \frac{\|\boldsymbol{v}_n\|^4}{2\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)\big(p(\boldsymbol{x}_n) - p(\boldsymbol{x}^\star)\big)} \\
&= 1 - \frac{\|\boldsymbol{v}_n\|^4}{2\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)\big(\frac{1}{2}\boldsymbol{x}_n Q \boldsymbol{x}_n^\intercal + D\boldsymbol{x}_n^\intercal + \frac{1}{2}DQ^{-1}D^\intercal\big)} \\
&= 1 - \frac{\|\boldsymbol{v}_n\|^4}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)\big(\boldsymbol{x}_n Q \boldsymbol{x}_n^\intercal + 2D\boldsymbol{x}_n^\intercal + DQ^{-1}D^\intercal\big)}.
\end{aligned}
$$

Note in the denominator we may rewrite some of the terms:

$$
\begin{aligned}
\boldsymbol{x}_n Q \boldsymbol{x}_n^\intercal &= \boldsymbol{x}_n Q (Q^{-1}Q) \boldsymbol{x}_n^\intercal = (\boldsymbol{x}_n Q) Q^{-1} (\boldsymbol{x}_n Q)^\intercal, \\
2D\boldsymbol{x}_n^\intercal &= D\boldsymbol{x}_n^\intercal + D\boldsymbol{x}_n^\intercal = \boldsymbol{x}_n D^\intercal + D(Q^{-1}Q)\boldsymbol{x}_n^\intercal \\
&= \boldsymbol{x}_n (QQ^{-1}) D^\intercal + DQ^{-1}(\boldsymbol{x}_n Q)^\intercal \\
&= (\boldsymbol{x}_n Q) Q^{-1} D^\intercal + DQ^{-1}(\boldsymbol{x}_n Q)^\intercal.
\end{aligned}
$$

This allows us to rewrite in the following convenient form

$$
\begin{aligned}
\frac{p(\boldsymbol{x}_{n+1}) - p(\boldsymbol{x}^\star)}{p(\boldsymbol{x}_n) - p(\boldsymbol{x}^\star)} &= 1 - \frac{\|\boldsymbol{v}_n\|^4}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)(\boldsymbol{x}_n Q + D)Q^{-1}(\boldsymbol{x}_n Q + D)^\intercal} \\
&= 1 - \frac{\|\boldsymbol{v}_n\|^4}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)\mathcal{Q}_{(\boldsymbol{Q^{-1}})}(\boldsymbol{v}_n)}.
\end{aligned}
$$

We are ready to state the main result of this subsection:

THEOREM 3.9. *Given a d-dimensional vector D, and a positive definite symmetric matrix Q of size $d \times d$, consider the quadratic function $p(\boldsymbol{x}) = \frac{1}{2}\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{x}) + \langle D, \boldsymbol{x}\rangle$. Any sequence $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ of steepest descent converges to the global minimum $\boldsymbol{x}^\star = -DQ^{-1}$. The sequence of evaluations $\{p(\boldsymbol{x}_n)\}_{n\in\mathbb{N}}$ converges linearly to $p(\boldsymbol{x}^\star) = -\frac{1}{2}\mathcal{Q}_{(\boldsymbol{Q^{-1}})}(D)$. In particular, if $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_d$ are the eigenvalues of Q, then*

$$
\frac{p(\boldsymbol{x}_{n+1}) - p(\boldsymbol{x}^\star)}{p(\boldsymbol{x}_n) - p(\boldsymbol{x}^\star)} \leq \left(\frac{\lambda_d - \lambda_1}{\lambda_d + \lambda_1}\right)^2
$$

PROOF. We start by offering the following lower bound estimate[3] involving the associated directions of steepest descent $\boldsymbol{v}_n$ in terms of the largest and smallest eigenvalues of Q. For all $n \in \mathbb{N}$,

$$
\frac{\|\boldsymbol{v}_n\|^4}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)\mathcal{Q}_{(\boldsymbol{Q^{-1}})}(\boldsymbol{v}_n)} \geq \frac{4\lambda_0\lambda_d}{(\lambda_0 + \lambda_d)^2} \tag{14}
$$

We have then

$$
\frac{p(\boldsymbol{x}_{n+1}) - p(\boldsymbol{x}^\star)}{p(\boldsymbol{x}_n) - p(\boldsymbol{x}^\star)} = 1 - \frac{\|\boldsymbol{v}_n\|^4}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)\mathcal{Q}_{(\boldsymbol{Q^{-1}})}(\boldsymbol{v}_n)}
$$

[3]This is left as and advanced exercise. It is not too tricky; if you are stuck, see e.g. [**1**, section 1.3.1] for a proof.

$$\leq 1 - \frac{4\lambda_1\lambda_d}{(\lambda_1+\lambda_d)^2} = \left(\frac{\lambda_d-\lambda_1}{\lambda_d+\lambda_1}\right)^2 \qquad \square$$

EXAMPLE 3.14. The global minimum value of the quadratic function $p(x,y) = 5x^2 + 5y^2 - xy - 11x + 11y + 11$ is zero, and found at $(1,-1)$. Notice that we may write this function in the form $p(x,y) = \frac{1}{2}\mathcal{Q}_{\boldsymbol{Q}}(x,y) + \langle D, [x,y]\rangle + 11$, where

$$D = [-11, 11], \qquad Q = \begin{bmatrix} 10 & -1 \\ 10 & -1 \end{bmatrix}.$$

The symmetric matrix Q has eigenvalues $\lambda_1 = 9 > 0$, $\lambda_2 = 11 > 0$ and is therefore positive definite. Theorem 3.9 states that sequences of steepest descent exhibit linear convergence with a rate of convergence not larger than $\delta = \left(\frac{11-9}{11+9}\right)^2 = 0.01$.

Observe the computations of the first six iterations for values of the ratios $\frac{p(\boldsymbol{x}_n)}{p(\boldsymbol{x}_{n-1})}$ when we use $(1.5, 3.5)$ as our initial guess.

n	x_n	y_n	$p(x_n, y_n)$	$\frac{\boldsymbol{p(x_n,y_n)}}{\boldsymbol{p(x_{n-1},y_{n-1})}}$
0	1.5000000000	3.5000000000	100.2500000000	
1	1.4498874016	−0.9600212545	1.0019989373	**0.0099950019**
2	1.0049975009	−0.9550224916	0.0100149812	**0.0099950019**
3	1.0044966254	−0.9996004124	0.0001000998	**0.0099950019**
4	1.0000499500	−0.9995504497	0.0000010005	**0.0099950019**
5	1.0000449438	−0.9999960061	0.0000000100	**0.0099950042**
6	1.0000004993	−0.9999955067	0.0000000001	**0.0099950528**

4. Effective Algorithms for Unconstrained Optimization

All of the methods we have explored so far (Newton-Raphson, Secant methods, Steepest descent) offer sound algorithms to compute local extrema of real-valued functions $f\colon \mathbb{R}^d \to \mathbb{R}$. They do have some pros and cons.

- The method of Steepest descent produces non-increasing iterations.
- In order to obtain new approximations on each Steepest descent iteration, we have to solve many different one-dimensional optimizations, each of them offering their own computational issues.
- Both Newton-Raphson and Secant methods offers faster sequences, but we cannot always guarantee convergence.
- Another drawback of Newton-Raphson is the fact that we do need to evaluate the function itself, its gradient and Hessian matrix.
- The recurrence formulas of Broyden's method are simple, and require only evaluations of the function itself.

The goal of this section is precisely gathering the best properties of the previous methods, so we may craft new methods with all the advantages, but none of the shortcomings.

Given a function $f\colon \mathbb{R}^d \to \mathbb{R}$ with continuous first partial derivatives, and a given initial guess $\boldsymbol{x}_0 \in \mathbb{R}^d$, we search for a recursive formula to approximate a minimum of f. We request that this formula has the form

$$\boldsymbol{x}_{n-1} = \boldsymbol{x}_n + t_n \boldsymbol{w}_n,$$

with positive parameters $t_n > 0$, and vectors $\boldsymbol{w}_n$ satisfying the following criteria:

Non-increasing sequences: $f(\boldsymbol{x}_{n+1}) < f(\boldsymbol{x}_n)$ whenever $\nabla f(\boldsymbol{x}_n) \neq 0$.
This is achieved by requiring the vectors ω_n to have an angle larger than $\pi/2$ with respect to $\nabla f(\boldsymbol{x}_n)$:

$$\langle \boldsymbol{w}_n, \nabla f(\boldsymbol{x}_n)\rangle < 0. \tag{15}$$

Why does this work? Consider at each step $n \in \mathbb{N}$ the restriction $\varphi_n(t) = f\big(\boldsymbol{x}_n + t\boldsymbol{w}_n\big)$ of f over the line $\boldsymbol{x}_n + t\boldsymbol{w}_n$ with $t > 0$. We have then $\varphi_n'(0) = \langle \nabla f(\boldsymbol{x}_n), \boldsymbol{w}_n\rangle < 0$ (a decreasing function near $t = 0$). We have a guaranteed value $t_n > 0$ (that could be very small) that give us a point $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + t_n\boldsymbol{w}_n$ with $f(\boldsymbol{x}_{n+1}) < f(\boldsymbol{x}_n)$.

Control over length of steps: The steps t_n are not *too short, nor too long.*
This is achieved by picking first $0 < \mu < \lambda < 1$ and forcing vectors ω_n that satisfy

$$\langle \boldsymbol{w}_n, \nabla f(\boldsymbol{x}_{n+1}) - \lambda\, \nabla f(\boldsymbol{x}_n)\rangle > 0 \tag{16}$$

$$f(\boldsymbol{x}_{n+1}) \leq f(\boldsymbol{x}_n) + \mu t_n \langle \boldsymbol{w}_n\, \nabla f(\boldsymbol{x}_n)\rangle \tag{17}$$

if this is at all possible.

Easy to compute: Duh!

Is it possible to create a iteration satisfying these criteria? The following result gives us a condition that helps in this regard:

THEOREM 3.10 (Wolfe). *Suppose that $f\colon \mathbb{R}^d \to \mathbb{R}$ is a real-valued function with continuous partial derivatives. Assume there exists $M \in \mathbb{R}$ so that $f(x) \geq M$. Let λ, μ be fixed numbers satisfying $0 < \lambda < \mu < 1$. If $\boldsymbol{w}_n$, $\boldsymbol{x}_n \in \mathbb{R}^d$ satisfy Criterion* (15)*, then there exist real numbers a_n, b_n such that $0 \leq a_n < b_n$ and*

(a) Criterion (16) *is satisfied for any choice of $t_n \in (0, b_n)$, and*
(b) Criterion (17) *is satisfied for any choice of $t_n \in (a_n, b_n)$.*

REMARK 3.8. For a proof, see [**13**, Theorem 3.3.1].

Using these principles, we are going to see two constructions based upon secant methods: the DFP and BFGS methods.

4.1. The DFP Method. The Davidon-Fletcher-Powell method is is one of the earliest and most effective secant methods. Its effectiveness stems from the fact that the method simultaneously generates conjugate directions and constructs an approximation of the inverse of the Hessian matrix. In each step, the inverse of the Hessian is updated by the sum of two symmetric rank 1 matrices. For this reason, it is referred to as a rank 2 correction procedure. It is also called a variable metric method.

To minimize $f\colon \mathbb{R}^d \to \mathbb{R}$, select an initial guess $\boldsymbol{x}_0$ and an initial *positive definite* matrix $\boldsymbol{A}_0$. If $\boldsymbol{x}_n$ and $\boldsymbol{A}_n$ have been computed, then

(a) Find $t_n > 0$ so that $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + t_n \overbrace{\big(-\boldsymbol{A}_n \cdot \nabla f(\boldsymbol{x}_n)^\intercal\big)}^{\boldsymbol{w}_n}$ satisfies criteria (16) and (17).

(b) Update

$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n + \tfrac{1}{\langle \boldsymbol{y}_n, \boldsymbol{d}_n\rangle}\big(\boldsymbol{d}_n \otimes \boldsymbol{d}_n\big) - \tfrac{1}{\mathcal{Q}_{\boldsymbol{A_n}}(\boldsymbol{y}_n)}\big(\boldsymbol{A}_n\boldsymbol{y}_n^\intercal \otimes \boldsymbol{A}_n\boldsymbol{y}_n^\intercal\big) \tag{18}$$

with $\boldsymbol{d}_n = \boldsymbol{x}_{n+1} - \boldsymbol{x}_n = t_n\boldsymbol{w}_n$, and $\boldsymbol{y}_n = \nabla f(\boldsymbol{x}_{n+1}) - \nabla f(\boldsymbol{x}_n)$.

REMARK 3.9. Unlike in the general Broyden's method, all the matrices $\boldsymbol{A}_n$ constructed in (18) are positive definite.

4.2. The BFGS method. The DFP method was soon superseded by the BFGS method, which is its dual (interchanging the roles of $\boldsymbol{d}_n$ and $\boldsymbol{y}_n$).

Different parts of this method were devised by Broyden, Fletcher, Goldfarb and Shanno independently in 1969. Their communications were sent as manuscripts between March and June of 1969, and reviewed between October 1969 and January 1970. They were all published in 1970. Each of these mathematicians derived similar formulas using different techniques (see [**3**], [**7**], [**10**], [**15**] and [**16**])

To minimize $f\colon \mathbb{R}^d \to \mathbb{R}$, select an initial guess $\boldsymbol{x}_0$ and an initial *positive definite* matrix $\boldsymbol{A}_0$. If $\boldsymbol{x}_n$ and $\boldsymbol{A}_n$ have been computed, then

(a) Find $t_n > 0$ so that $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + t_n \overbrace{\big(-\boldsymbol{A}_n^{-1} \cdot \nabla f(\boldsymbol{x}_n)^\intercal\big)}^{\boldsymbol{w}_n}$ satisfies criteria (16) and (17).

(b) Update

$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n + \tfrac{1}{\langle \boldsymbol{d}_n, \boldsymbol{y}_n\rangle}\big(\boldsymbol{y}_n \otimes \boldsymbol{y}_n\big) - \tfrac{1}{\mathcal{Q}_{\boldsymbol{A_n}}(\boldsymbol{d}_n)}\big(\boldsymbol{A}_n\boldsymbol{d}_n^\intercal \otimes \boldsymbol{A}_n\boldsymbol{d}_n^\intercal\big) \tag{19}$$

with $\boldsymbol{d}_n = \boldsymbol{x}_{n+1} - \boldsymbol{x}_n = t_n\boldsymbol{w}_n$, and $\boldsymbol{y}_n = \nabla f(\boldsymbol{x}_{n+1}) - \nabla f(\boldsymbol{x}_n)$.

REMARK 3.10. Like in the DFP method, all the matrices $\boldsymbol{A}_n$ constructed in (19) are positive definite. For a proof, see e.g. [**13**, Theorem 3.5.2].

EXAMPLE 3.15. In Python there is an implementation of BFGS in the libraries `scipy.optimize`: the routine `fmin_bfgs()`. The following session illustrates how to use this method to approximate the minimum of the Rosenbrock function $\mathcal{R}_{1,1}$ with an initial guess $(x_0, y_0) = (-2, 2)$

```
import numpy as np, matplotlib.pyplot as plt
from scipy.optimize import fmin_bfgs

# Rosenbrock R_{1,1} function
def R(x):
        return (1.0-x[0])**2 + (x[1] - x[0]**2)**2

# its Jacobian/gradient ∇R_{1,1}
def jacR(x):
        return np.array([-2.*(1.-x[0])-4.*x[0]*(x[1]-x[0]**2),
                         2.*(x[1]-x[0]**2)])

# Setup for diagrams.
x = np.linspace(-3,3)
y = np.linspace(-3,3)
X,Y = np.meshgrid(x,y)
```

We call `fmin_bfgs` with the function and its gradient (with the option `fprime=`), the initial guess $(-2, 2)$, and activate the option `retall=True` that offers the complete list of iterations obtained by the algorithm.

```
>>> result = fmin_bfgs(R, [-2.,2.], fprime=jacR, retall=True)
Optimization terminated successfully.
         Current function value: 0.000000
         Iterations: 12
         Function evaluations: 15
         Gradient evaluations: 15

>>> plt.figure(figsize=(8,8));
... plt.axes(aspect='equal');
... plt.contour(X, Y, (1-X)**2+(Y-X**2)**2, \
...             colors='k', \
...             levels=[0.2,0.8,1,1.4,1.78,2.23,4,5.4,8,13,32,64]);
... plt.plot(x, x**2, 'r--');
... plt.xlim(-3, 3);
... plt.ylim(-1, 3);
... plt.plot([p[0] for p in result[1]], [p[1] for p in result[1]], 'b.-');
... plt.title("Convergence to critical point:\nBFGS on Rosenbrock");
... plt.show()
```

Exercises

PROBLEM 3.1 (Basic). [**9**, p.249 #1] The following sequences all converge to zero.

$$v_n = n^{-10} \quad w_n = 10^{-n} \quad x_n = 10^{-n^2} \quad y_n = n^{10}3^{-n} \quad z_n = 10^{-3\cdot 2^n}$$

Indicate the type of convergence (See Appendix A).

PROBLEM 3.2 (Advanced). [**9**, p.249 #4] Give an example of a positive sequence $\{\varepsilon_n\}_{n\in\mathbb{N}}$ converging to zero in such a way that $\lim_n \frac{\varepsilon_{n+1}}{\varepsilon_n^p} = 0$ for some $p > 1$, but not converging to zero with any order $q > p$.

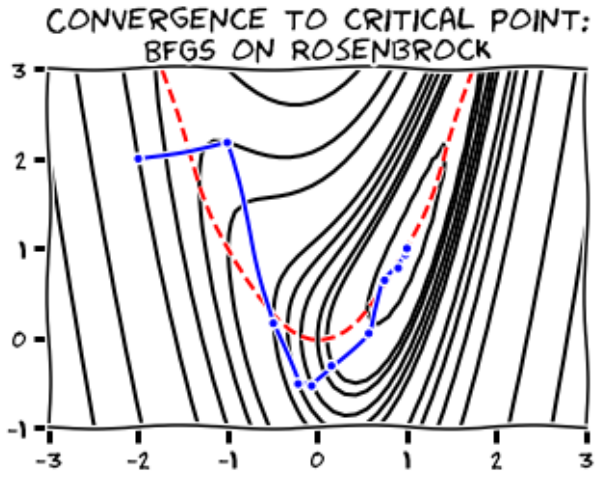


FIGURE 3.8. The BFGS method: Rosenbrock function

PROBLEM 3.3 (Basic). Find an example of a function $f\colon \mathbb{R} \to \mathbb{R}$ (different from the function in Example 3.2) with a unique root at $x = 0$ for which the Newton-Raphson sequence is a loop no matter the initial guess $x_0 \neq 0$: $x_{2n} = x_0$, $x_{2n+1} = -x_0$ for all $n \in \mathbb{N}$. Bonus points is your function is trigonometric.

PROBLEM 3.4 (Intermediate). [**9**, p.251 #14] Consider the equation

$$x = \cos x.$$

(a) Show graphically that there exists a unique positive root $x^\star$. Indicate approximately where it is located.
(b) Show that Newton's method applied to $f(x) = x - \cos x$ converges for any initial guess $x_0 \in \left[0, \frac{\pi}{2}\right]$.

PROBLEM 3.5 (Intermediate). [**9**, p.251 #16] Consider the equation

$$\tan x + \lambda x = 0, \quad (0 < \lambda < 1).$$

(a) Show graphically, as simply as possible, that there is exactly one root $x^\star$ in the interval $\left[\frac{1}{2}\pi, \pi\right]$.
(b) Does Newton's method converge to the root $x^\star \in \left[\frac{1}{2}\pi, \pi\right]$ if the initial approximation is taken to be $x_0 = \pi$? Justify your answer.

PROBLEM 3.6 (Intermediate). [**9**, p.252 #17] Consider the equation

$$\log^2 x - x - 1 = 0, (x > 0).$$

(a) Graphical considerations suggest that there is exactly one positive solution $x^\star$, and that $0 < x^\star < 1$. Prove this.
(b) What is the largest positive $0 < x_0 \leq 1$ such that Newton's method with $f(x) = \log^2 x - x - 1$ started at x_0 converges to $x^\star$?

PROBLEM 3.7 (Advanced). [**9**, p.252 #18] Consider Kepler's equation

$$x - a\sin x - b = 0, \quad 0 < |a| < 1, \quad b \in \mathbb{R}$$

where a, b are parameters.

(a) Show that for each a, b there is exactly one real solution $x^\star = x^\star(a,b)$ that satisfies

$$b - |a| \leq x^\star(a,b) \leq b + |a|$$

(b) Let $m \in \mathbb{N}$ satisfy $m\pi < b < (m+1)\pi$. Show that Newton's method for $f(x) = x - a\sin x - b$ with starting value

$$x_0 = \begin{cases} (m+1)\pi & \text{if } (-1)^m a > 0 \\ m\pi & \text{otherwise} \end{cases}$$

is guaranteed to converge (monotonically) to $x^\star(a,b)$.

PROBLEM 3.8 (Basic). Consider the two equivalent equations

$$x\log x - 1 = 0, \tag{20}$$

$$\log x - \frac{1}{x} = 0. \tag{21}$$

(a) Show that there is exactly one positive root and find a rough interval containing it.

(b) For both (20) and (21), determine the largest interval on which Newton's method converges.
Hint: Investigate the convexity of the functions involved.

PROBLEM 3.9 (CAS). Design a process in `desmos.com` to test the search for critical points given by the recursion formulas produced by Newton's method.

PROBLEM 3.10 (CAS). In a computer language or CAS of your choice, design a routine that gathers the following as input:

- the definition of a generic real-valued function $f\colon \mathbb{R} \to \mathbb{R}$,
- the derivative f' of that function,
- an initial guess $x_0 \in \mathbb{R}$,
- a number N of steps,

and produces the first $N+1$ terms of the Newton-Raphson sequence to approximate a root of f.

Modify the previous routine to receive as input, instead of a number of steps, a *tolerance* `tol` indicating the accuracy of the solution. For example, if we require a root of the equation $f(x) = 0$ accurate to the first 16 correct decimal places, we use `tol = 1e-16`.

PROBLEM 3.11 (Basic). The purpose of this problem is the design of *Horner's method* to evaluate polynomials effectively. Given a polynomial

$$p(x) = \sum_{k=0}^{n} a_k x^k = a_0 + a_1 x + \cdots + a_n x^n,$$

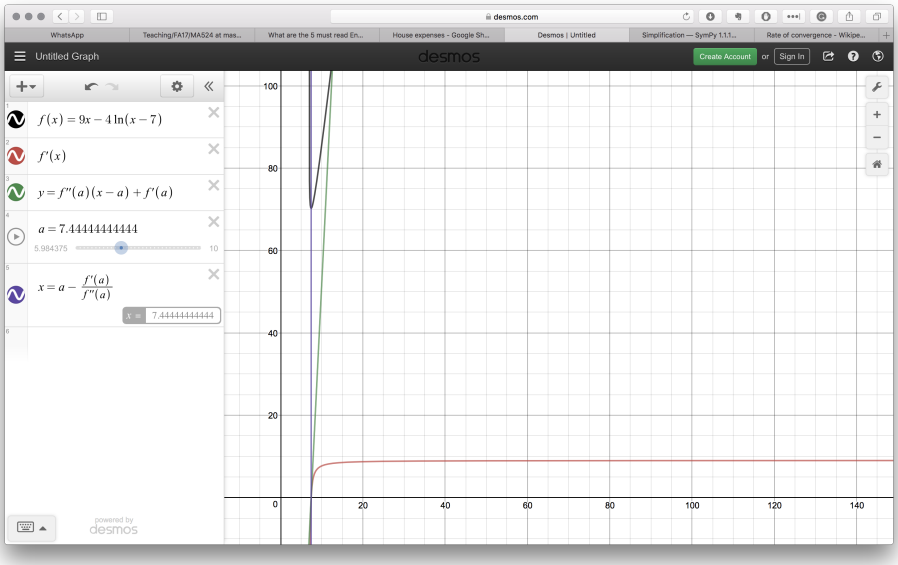


FIGURE 3.9. Newton method in `desmos.com`

where $a_0, a_1, \ldots, a_n$ are real numbers, and given $x_0 \in \mathbb{R}$, we define the Horner's scheme $\{b_0, b_1, \ldots, b_n\}$ to evaluate $p(x_0)$ as follows:

$$\begin{aligned} b_n &= a_n \\ b_{n-1} &= a_{n-1} + b_n x_0 \\ b_{n-2} &= a_{n-2} + b_{n-1} x_0 \\ &\vdots \\ b_0 &= a_0 + b_1 x_0 \end{aligned}$$

(a) Prove that $b_0 = p(x_0)$
(b) Use Horner's method to evaluate $p(x) = 2x^3 - 6x^2 + 2x - 1$ at $x = 3$. Illustrate all steps, and count the number of basic operations (addition, subtraction, multiplication, division) used.
(c) Employ the usual method of evaluation of polynomials to evaluate $p(x) = 2x^3 - 6x^2 + 2x - 1$ at $x = 3$. Count the number of basic operations (note that a^3 counts as two multiplications: $a \times a \times a$, for instance)

PROBLEM 3.12 (CAS). In a computer language or CAS of your choice, write a routine to apply Horner's scheme to evaluate polynomials. Your routine should gather the following inputs:

- A list of coefficients `[a0, a1, ..., an]` representing the polynomial $p(x) = a_0 + a_1 x + \cdots + a_n x^n$.
- A value x_0

The output of your routine should be $p(x_0)$.

PROBLEM 3.13 (CAS). Use any of the routines that you wrote in Problem 3.10 to produce a table and a visual representation for the numerical solution of the following equations, with the given initial guesses and steps.

(a) $f(x) = \sin x$, with $x_0 = 0.5$, 5 steps.
(b) $f(x) = \sin x$, with $x = 3$, enough steps to obtain accurately the first 16 correct decimal places of π.
(c) $f(x) = -1 + \log x$, with $x = 2$, enough steps to obtain accurately the first 16 correct decimal places of e.

PROBLEM 3.14 (CAS). The objective of this problem is to use Newton's method to find an approximation to the golden ratio $\phi = \frac{1}{2}(1+\sqrt{5})$ accurate to the first 16 decimal places. Find first an appropriate polynomial $p(x)$ with integer coefficients for which ϕ is a root. Employ any of the routines that you wrote in Problem 3.10 with a good initial guess to guarantee the required result.

PROBLEM 3.15 (Intermediate—CAS). [**8**, lec3_newton_mthd, 3.1] Consider the function

$$f(x) = 9x - 4\log(x-7).$$

We wish to study the behavior of Newton-Raphson to find approximations to the critical points of this function.

(a) Find the domain D of f.
(b) Find the global minimum of f analytically.
(c) Compute an exact formula for the Newton-Raphson iterate x_{n+1} for an initial guess $x_0 \in D$.
(d) Compute five iterations of the Newton-Raphson method starting at each of the following initial guesses:
    (a) $x_0 = 7.4$.
    (b) $x_0 = 7.2$.
    (c) $x_0 = 7.01$.
    (d) $x_0 = 7.8$.
    (e) $x_0 = 7.88$.
(e) Prove that the Newton-Raphson method converges to the optimal solution for any initial guess $x_0 \in (7, 7.8888)$.
(f) What is the behavior of the Newton-Raphson method if the initial guess is not in the interval $(7, 7.8888)$?

PROBLEM 3.16 (Intermediate—CAS). [**8**, lec3_newton_mthd, 3.2] Consider the function

$$f(x) = 6x - 4\log(x-2) - 3\log(25-x).$$

We wish to study the behavior of Newton-Raphson to find approximations to the critical points of this function.

(a) Find the domain D of f.

(b) Find the global minimum of f analytically.
(c) Compute an exact formula for the Newton-Raphson iterate x_{n+1} for an initial guess $x_0 \in D$.
(d) Compute five iterations of the Newton-Raphson method starting at each of the following initial guesses:
  (a) $x_0 = 2.6$.
  (b) $x_0 = 2.7$.
  (c) $x_0 = 2.4$.
  (d) $x_0 = 2.8$.
  (e) $x_0 = 3$.
(e) Prove that the Newton-Raphson method converges to the optimal solution for any initial guess $x_0 \in (2, 3.05)$.
(f) What is the behavior of the Newton-Raphson method if the initial guess is not in the interval $(2, 3.05)$?

PROBLEM 3.17 (Basic). Approximate the solution of the following system by computing two steps of Newton-Raphson's method for an appropriate function $\boldsymbol{g}\colon \mathbb{R}^3 \to \mathbb{R}^3$, and initial guess $\boldsymbol{x}_0 = (1, 0, 1)$.

$$\begin{cases} 3 &= x^2 + y^2 + z^2 \\ 1 &= x^2 + y^2 - z \\ 1 &= x + y + z \end{cases}$$

PROBLEM 3.18 (Intermediate). [**1**, p.91 #1.4.1] The purpose of this exercise is to show that Newton's method is unaffected by linear scaling of the variables. Consider a linear invertible transformation of variables $\boldsymbol{x}^\intercal = \boldsymbol{A} \cdot \boldsymbol{y}^\intercal$. Write Newton's method in the space of the variables $\boldsymbol{y}$ and show that it generates the sequence ${\boldsymbol{y}_n}^\intercal = \boldsymbol{A}^{-1} \cdot {\boldsymbol{x}_n}^\intercal$, where $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ is the sequence generated by Newton's method in the space of variables $\boldsymbol{x}$.

PROBLEM 3.19 (Basic). Let $\boldsymbol{A}$ be a square matrix. An *LU-decomposition* is a factorization of $\boldsymbol{A} = \boldsymbol{L} \cdot \boldsymbol{U}$ into a *lower triangular* matrix $\boldsymbol{L}$ and an *upper triangular* matrix $\boldsymbol{U}$, both of which have non-zero entries in their diagonals. For example, the general case for 3×3 square matrices:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \underbrace{\begin{bmatrix} \ell_{11} & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} \end{bmatrix}}_{\boldsymbol{L}} \cdot \underbrace{\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}}_{\boldsymbol{U}}$$

(a) Find an LU-decomposition of the following matrix

$$\begin{bmatrix} 4 & 3 \\ 6 & 3 \end{bmatrix}$$

that satisfies that all diagonal entries of $\boldsymbol{L}$ are ones.
(b) Find an example of a 2×2 square matrix for which there is not any possible LU-decomposition.

PROBLEM 3.20 (Advanced). Prove the following statements:

(a) A square matrix $\boldsymbol{A}$ of size $d \times d$ admits an LU-decomposition if and only if the leading principal minors are non-zero: $\Delta_k \neq 0$ for $1 \leq k \leq d$.
(b) If $\boldsymbol{A}$ is a symmetric positive definite matrix, then it is possible to find an LU-decomposition where $\boldsymbol{U} = \boldsymbol{L}^\intercal$: $\boldsymbol{A} = \boldsymbol{L} \cdot \boldsymbol{L}^\intercal$. In this case, this factorization is also called a *Cholesky decomposition*.

PROBLEM 3.21 (CAS). In a computer language or CAS of your choice, design a routine that solves a linear system

$$\overbrace{\begin{bmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{d1} & \cdots & a_{dd} \end{bmatrix}}^{\boldsymbol{A}} \cdot \overbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}}^{\boldsymbol{x}^\intercal} = \overbrace{\begin{bmatrix} c_1 \\ \vdots \\ c_d \end{bmatrix}}^{\boldsymbol{c}^\intercal}$$

by performing first a LU-decomposition $\boldsymbol{A} = \boldsymbol{L} \cdot \boldsymbol{U}$ (provided this is possible!) and manipulating the resulting equation to solve instead the two (faster) systems:

(a) Find $\boldsymbol{y}$ that solves the system $\boldsymbol{L} \cdot \boldsymbol{y}^\intercal = \boldsymbol{c}^\intercal$ by Gaussian elimination.
(b) Find $\boldsymbol{x}$ that solves the system $\boldsymbol{U} \cdot \boldsymbol{x}^\intercal = \boldsymbol{y}^\intercal$ by Gaussian elimination.

You may design a routine that computes LU-decompositions, or you may use a built-in routine for that purpose.

PROBLEM 3.22 (CAS). In a computer language or CAS of your choice, design a routine that gathers the following as input:

- the definition of a generic real-valued function $f \colon \mathbb{R}^d \to \mathbb{R}$,
- the gradient ∇f of that function,
- an initial guess $\boldsymbol{x}_0 \in \mathbb{R}^d$,
- a number N of steps,

and produces the first $N + 1$ terms of the Newton-Raphson sequence to approximate a root of f.

PROBLEM 3.23 (Advanced). Prove the *Local convergence for the Secant method*, Theorem 3.3.

PROBLEM 3.24 (Basic). Approximate the solution for the system in Problem 3.17 by computing the first two iterations of a Broyden method with $\boldsymbol{A}_0 = \nabla \boldsymbol{g}(\boldsymbol{x}_0)$ for an appropriate function $\boldsymbol{g} \colon \mathbb{R}^d \to \mathbb{R}$ and initial guess $\boldsymbol{x}_0 = (1, 0, 1)$.

PROBLEM 3.25 (Basic). Compute the first two iterations of Broyden method with initial guess $(1, 4)$ to search for the critical points of the function $f(x, y) = 2x^2 + y^2 - xy$

(a) Using $\boldsymbol{A}_0 = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$.
(b) using $\boldsymbol{A}_0 = \mathrm{Hess} f(1, 4)$.

PROBLEM 3.26 (Advanced). Prove Theorems 3.5, and 3.6.

PROBLEM 3.27 (Advanced). We want to prove estimate (14) in the proof of Theorem 3.9 in page 50. This follows directly from the equivalent statement below, which is easier to handle. Prove the following result:

Kantorovich Estimate. Given a positive definite symmetric matrix Q of size $d \times d$, consider the quadratic function $p(\boldsymbol{x}) = \frac{1}{2}\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{x})$. Assume $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_d$ are the eigenvalues of Q. For any sequence $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ of steepest descent for f, we have the following estimate involving the directions of steepest descent $\{\boldsymbol{v}_n\}_{n\in\mathbb{N}}$:

$$\frac{\|\boldsymbol{v}_n\|^4}{\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{v}_n)\mathcal{Q}_{(\boldsymbol{Q}^{-1})}(\boldsymbol{v}_n)} \geq \frac{4\lambda_1\lambda_d}{(\lambda_1+\lambda_d)^2}$$

PROBLEM 3.28 (Basic). Approximate the solution for the system in Problem 3.17 by computing the first two iterations of a Steepest descent method for an appropriate function $\boldsymbol{g}\colon \mathbb{R}^d \to \mathbb{R}$ and initial guess $\boldsymbol{x}_0 = (1, 0, 1)$.

PROBLEM 3.29 (Basic). Compute the first two iterations of the method of Steepest descent with initial guess $(1, 4)$ to search for the critical points of the function $f(x, y) = 2x^2 + y^2 - xy$.

PROBLEM 3.30. [**8**, lec5_steep_desce, 8.3] Consider the quadratic polynomial

$$p(x, y) = \tfrac{1}{2}\mathcal{Q}_{\boldsymbol{Q}}(x, y) + \langle D, [x, y]\rangle + 13,$$

with

$$D = [4, -15], \qquad Q = \begin{bmatrix} 10 & -9 \\ -9 & 10 \end{bmatrix}$$

(a) Find the global minimum value of p, and its location.
(b) Compute the eigenvalues of Q. Is Q positive definite?
(c) What is the worst-case scenario rate of convergence of sequences of steepest descent for this function?
(d) Compute sequences of steepest descent for this function with the initial guesses below. Make sure to report a table similar to the one in Example 3.14.
   - $(0, 0)$
   - $(-0.4, 0)$
   - $(10, 0)$
   - $(11, 0)$

PROBLEM 3.31. [**8**, lec5_steep_desce, 8.4] Consider the quadratic polynomial

$$p(x, y, z) = \tfrac{1}{2}\mathcal{Q}_{\boldsymbol{Q}}(x, y, z) + \langle D, [x, y, z]\rangle,$$

with

$$D = [12, -47, -8], \qquad Q = \begin{bmatrix} 10 & -18 & 2 \\ -18 & 40 & -1 \\ 2 & -1 & 3 \end{bmatrix}$$

(a) Find the global minimum value of p, and its location.
(b) Compute the eigenvalues of Q. Is Q positive definite?
(c) What is the worst-case scenario rate of convergence of sequences of steepest descent for this function?
(d) Compute sequences of steepest descent for this function with the initial guesses below. Make sure to report a table similar to the one in Example 3.14.
- $(0, 0, 0)$
- $(15.09, 7.66, -6.56)$
- $(11.77, 6.42, -4.28)$
- $(4.46, 2.25, 1.85)$

CHAPTER 4

Existence and Characterization of Extrema for Constrained Optimization

Let us begin by reviewing the concept of *constrained optimization*, and some associated notation and terminology. The objective is the search for extrema of a function $f\colon D \to \mathbb{R}$ where the input values $\boldsymbol{x}$ belong in an open subset $D \subseteq \mathbb{R}^d$ and satisfy a finite set of *constraints* of the form

$$g_1(\boldsymbol{x}) \leq 0, g_2(\boldsymbol{x}) \leq 0, \ldots, g_m(\boldsymbol{x}) \leq 0,$$
$$h_1(\boldsymbol{x}) = 0, h_2(\boldsymbol{x}) = 0, \ldots, h_\ell(\boldsymbol{x}) = 0,$$

for real-valued functions $g_k\colon \mathbb{R}^d \to \mathbb{R}, (1 \leq k \leq m)$, $h_k\colon \mathbb{R}^d \to \mathbb{R}, (1 \leq k \leq \ell)$.

For simplicity, we write instead

$$(P) \begin{cases} \min_{x \in D} f(\boldsymbol{x}) \\ g_1(\boldsymbol{x}) \leq 0, \ldots, g_m(\boldsymbol{x}) \leq 0 \\ h_1(\boldsymbol{x}) = 0, \ldots, h_\ell(\boldsymbol{x}) = 0 \end{cases} \tag{22}$$

or better, if we set

$$S = \{\boldsymbol{x} \in D : g_1(\boldsymbol{x}) \leq 0, \ldots, g_m(\boldsymbol{x}) \leq 0, h_1(\boldsymbol{x}) = 0, \ldots, h_\ell(\boldsymbol{x}) = 0\},$$

we may simply write

$$(P) = \min_{x \in S} f.$$

We refer to it as the *program* (P). The function f is called the *objective function of* (P). We refer to the functions g_k as the *inequality constraints*. The functions h_k are called the *equality constraints*.

A point $\boldsymbol{x} \in D$ that satisfies all the constraints of the program (P) is said to be *feasible*. The set of all feasible points is called the *feasibility region* of (P). If the feasibility region is non-empty, we say that the program (P) is *consistent*. If a feasible point satisfies $g_k(\boldsymbol{x}) < 0$ for all inequality constraints, we call it a *Slater point*. Consistent programs (P) that have Slater points are said to be *super-consistent*.

If the objective function f and all constraints g_k, h_k of a program (P) are linear functions, we denote it by (LP) and refer to it as a *linear program*. If the objective function, all constraints g_k, h_k and the set D are convex, we call (P) a *convex program*.

EXAMPLE 4.1. Let $f(x,y) = x^4 + y^4$, and consider the program $(P) = \min_{x \in S} f(x,y)$, where $S = \{(x,y) \in \mathbb{R}^2 : x^2 \leq 1, y^2 \leq 1, e^{x+y} \leq 1\}$.

The inequality constraints are $g_1(x,y) = x^2 - 1$, $g_2(x,y) = y^2 - 1$ and $g_3(x,y) = e^{x+y} - 1 \leq 0$ (although you may choose simpler equivalent expressions). The feasibility region is thus a triangular region (see Figure 4.1):

$$S = \{(x,y) \in \mathbb{R}^2 : |x| \leq 1, |y| \leq 1, x + y \leq 0\} \neq \emptyset.$$

This is a super-consistent convex program, since any interior point of the

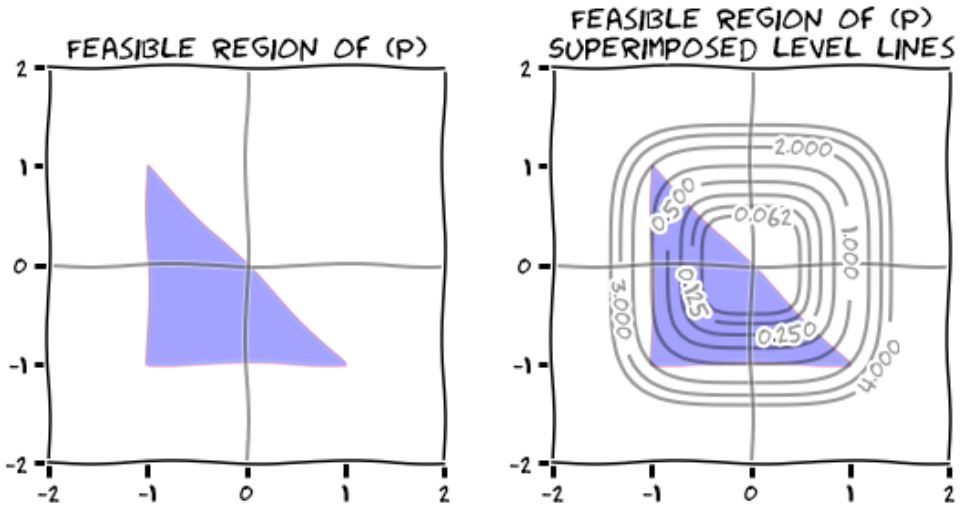


FIGURE 4.1. Can you tell what are the global maximum and minimum values of f in S?

triangle S is a Slater point for (P), and all relevant functions are convex.

DEFINITION. Given a consistent program (P) as defined in (22), for each feasible point $\boldsymbol{x} \in S$, we define:

(a) The *cone of improving directions* of f at $\boldsymbol{x}$, as

$$\mathcal{F}_0(\boldsymbol{x}) = \{\boldsymbol{v} \in \mathbb{R}^d : \|\boldsymbol{v}\| = 1, \langle \nabla f(\boldsymbol{x}), \boldsymbol{v} \rangle < 0\}$$

(b) The set of *indices of the binding inequality constraints* for $\boldsymbol{x}$, as

$$\mathcal{I}(\boldsymbol{x}) = \big\{k \in \{1, \ldots, m\} : g_k(\boldsymbol{x}) = 0\big\}.$$

(c) The *cone of inward pointing directions for the binding constraints* at $\boldsymbol{x}$, as

$$\mathcal{G}_0(\boldsymbol{x}) = \{\boldsymbol{v} \in \mathbb{R}^d : \|\boldsymbol{v}\| = 1, \langle \nabla g_k(\boldsymbol{x}), \boldsymbol{v} \rangle < 0 \text{ for all } k \in \mathcal{I}(\boldsymbol{x})\}$$

(d) The set of *tangent directions for the equality constraints* at $\boldsymbol{x}$, as

$$\mathcal{H}_0(\boldsymbol{x}) = \{\boldsymbol{v} \in \mathbb{R}^d : \langle \nabla h_k(\boldsymbol{x}), \boldsymbol{v} \rangle = 0, 1 \leq k \leq \ell\}$$

EXAMPLE 4.2. For the program (P) in Example 4.1, consider the feasible points $(0,0)$, $(-1,-1)$ and $(0,-1/2)$. Since $\nabla f(x,y) = [4x^3, 4y^3]$, we have the following cones on improving direction

$$\begin{aligned}\mathcal{F}_0(0,0) &= \emptyset,\\ \mathcal{F}_0(-1,-1) &= \{\boldsymbol{v} = (v_1,v_2) \in \mathbb{R}^2 : \|\boldsymbol{v}\| = 1, v_1 + v_2 > 0\},\\ \mathcal{F}_0(0,-1/2) &= \{\boldsymbol{v} = (v_1,v_2) \in \mathbb{R}^2 : \|\boldsymbol{v}\| = 1, v_2 > 0\}\end{aligned}$$

The indices of binding inequality constraints are

$$\mathcal{I}(0,0) = \{3\}, \quad \mathcal{I}(-1,-1) = \{1,2\}, \quad \mathcal{I}(0,-1/2) = \emptyset,$$

and therefore, the cones of inward pointing directions for the binding constraints are

$$\begin{aligned}\mathcal{G}_0(0,0) &= \{\boldsymbol{v} = (v_1,v_2) \in \mathbb{R}^2 : \|\boldsymbol{v}\| = 1, v_1 + v_2 < 0\},\\ \mathcal{G}_0(-1,-1) &= \{\boldsymbol{v} = (v_1,v_2) \in \mathbb{R}^2 : \|\boldsymbol{v}\| = 1, v_1 > 0, v_2 > 0\},\\ \mathcal{G}_0(0,-1/2) &= \emptyset.\end{aligned}$$

Since there are no equality constraints, we do not have any sets of tangent directions.

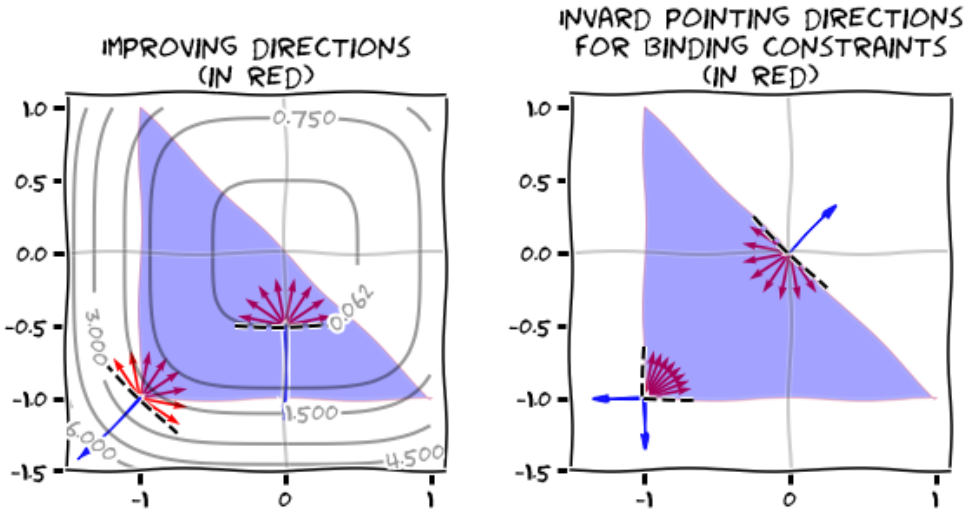


FIGURE 4.2. Cones for $(0,0)$, $(-1,-1)$ and $(0,-1/2)$.

In the following sections we are going to discuss necessary and sufficient conditions for the existence of (strict) global minima for any consistent program (P). We are going to focus exclusively on the main results without showing their proofs (if interested in those proofs, consult [**8**, lec6_constr_opt]).

1. Necessary Conditions

We begin with two results that focus on the structure of the equality constraints.

THEOREM 4.1 (Geometric Necessary Condition for Linear Equality Constraints). *If (P) is a consistent program with linear equality constraints $h_k(\boldsymbol{x}) = \langle \boldsymbol{a}_k, \boldsymbol{x}\rangle + b_k$ ($\boldsymbol{a}_k \in \mathbb{R}^d$, $b_k \in \mathbb{R}$) for all $1 \leq k \leq \ell$, then for all feasible local minima $\boldsymbol{x} \in S$,*

$$\mathcal{F}_0(\boldsymbol{x}) \cap \mathcal{G}_0(\boldsymbol{x}) \cap \mathcal{H}_0(\boldsymbol{x}) = \emptyset.$$

THEOREM 4.2 (Geometric Necessary Condition for Linearly Independent Equality Constraints). *If $\boldsymbol{x} \in S$ is a feasible local minimum for the consistent program (P), and the gradient vectors $\{\nabla h_k(\boldsymbol{x}) : 1 \leq k \leq \ell\}$ are linearly independent, then $\mathcal{F}_0(\boldsymbol{x}) \cap \mathcal{G}_0(\boldsymbol{x}) \cap \mathcal{H}_0(\boldsymbol{x}) = \emptyset$.*

As a consequence, an algebraic version of this geometric necessary condition gives the following result.

THEOREM 4.3 (Fritz John Necessary Conditions). *If $\boldsymbol{x} \in S$ is a feasible local minimum of the consistent program (P), then there exist $\lambda_k \geq 0$ for $0 \leq k \leq m$, and $\mu_1, \ldots, \mu_\ell \in \mathbb{R}$ so that*

(a) $[\lambda_0, \lambda_1, \ldots, \lambda_m, \mu_1 \ldots, \mu_\ell] \neq \mathbf{0}$,
(b) $\lambda_k g_k(\boldsymbol{x}) = 0$ *for all* $1 \leq k \leq m$.
(c) $\lambda_0 \nabla f(\boldsymbol{x}) + \sum_{k=1}^{m} \lambda_k \nabla g_k(\boldsymbol{x}) + \sum_{k=1}^{\ell} \mu_k \nabla h_k(\boldsymbol{x}) = 0$.

EXAMPLE 4.3. Continuing with example 4.1, let's check if the point $(0,0)$ is a candidate to optimal solution of this program. Let's use Theorem 4.3 to verify this claim:

$$\begin{aligned}
\nabla f(x,y) &= [4x^3, 4y^3] & \nabla f(0,0) &= [0,0],\\
\nabla g_1(x,y) &= [2x, 0] & \nabla g_1(0,0) &= [0,0],\\
\nabla g_2(x,y) &= [0, 2y] & \nabla g_2(0,0) &= [0,0],\\
\nabla g_3(x,y) &= \left[e^{x+y}, e^{x+y}\right] & \nabla g_3(0,0) &= [1,1].
\end{aligned}$$

Notice how the gradients *line up* nicely—can we find $\lambda_k \geq 0$ so that:

(a) $[\lambda_0, \lambda_1, \lambda_2, \lambda_3] \neq [0,0,0,0]$,
(b) $\lambda_1 = \lambda_2 = 0$ (since $\lambda_1 g_1(0,0) = -2\lambda_1$ and $\lambda_2 g_2(0,0) = -2\lambda_2$), and
(c) the following linear combination is equal to $[0,0]$

$$\lambda_0 \nabla f(0,0) + \lambda_1 \nabla g_1(0,0) + \lambda_2 \nabla g_2(0,0) + \lambda_3 \nabla g_3(0,0) = [0,0],$$
$$\lambda_0[0,0] + \lambda_1[0,0] + \lambda_2[0,0] + \lambda_3[1,1] = [0,0],$$

We may select, for instance $\lambda_0 = 1$, $\lambda_1 = \lambda_2 = \lambda_3 = 0$, which proves that the point $(0,0)$ is indeed a candidate for optimal solution of (P).

Further properties of the involved functions provide us with simpler sets of conditions

THEOREM 4.4 (Karush-Kuhn-Tucker Necessary Conditions). *If $\boldsymbol{x} \in S$ is a feasible local minimum of the consistent program (P) for which all the vectors $\{\nabla h_k(\boldsymbol{x}), \nabla g_j(\boldsymbol{x}) : 1 \leq k \leq \ell, j \in \mathcal{I}(\boldsymbol{x})\}$ are linearly independent, then there exist $\lambda_k \geq 0$ for $1 \leq k \leq m$, and $\mu_1, \ldots, \mu_\ell \in \mathbb{R}$ so that*

(a) $\lambda_k g_k(\boldsymbol{x}) = 0$ *for all* $1 \leq k \leq m$.
(b) $\nabla f(\boldsymbol{x}) + \sum_{k=1}^{m} \lambda_k \nabla g_k(\boldsymbol{x}) + \sum_{k=1}^{\ell} \mu_k \nabla h_k(\boldsymbol{x}) = 0$.

REMARK 4.1. The conditions (a) and (b) of Theorem 4.4 are called *the KKT conditions* of the program (P) in the literature. The values λ_k, μ_k are called *multipliers*.

EXAMPLE 4.4. Set $f(x, y) = (x-12)^2 + (y+6)^2$. Consider the program (P) designed to find the global minimum of this function on the set $S = \{(x, y) \in \mathbb{R}^2 : x^2+3x+y^2-4.5y \leq 6.5, (x-9)^2+y^2 \leq 64, 8x+4y = 20\}$. We want to prove that the point $(2, 1)$ is a good candidate for optimal solution of (P). The point $(2, 1)$ is feasible. To see this, set

$$\begin{aligned} g_1(x, y) &= x^2 + 3x + y^2 - 4.5y - 6.5, \\ g_2(x, y) &= (x-9)^2 + y^2 - 64, \\ h_1(x, y) &= 8x + 4y - 20, \end{aligned}$$

(or simpler equivalent constraints), and notice that

$$g_1(2, 1) = 0, \quad g_2(2, 1) = -14, \quad h_1(2, 1) = 0.$$

We have concluded that (P) is consistent. Notice that, $\mathcal{I}(2, 1) = \{1\}$, since $g_1(2, 1) = 0$, $g_2(2, 1) \neq 0$. Further,

$$\begin{aligned} \nabla f(x, y) &= [2(x-12), 2(y+6)] & \nabla f(2, 1) &= [-20, 14], \\ \nabla g_1(x, y) &= [2x+3, 2y-4.5] & \nabla g_1(2, 1) &= [7, -2.5], \\ \nabla g_2(x, y) &= [2(x-9), 2y] & \nabla g_2(2, 1) &= [-14, 2], \\ \nabla h_1(x, y) &= [8, 4] & \nabla h_1(2, 1) &= [8, 4], \end{aligned}$$

The vectors $\nabla g_1(2, 1) = [7, -2.5]$ and $\nabla h_1(2, 1) = [8, 4]$ are linearly independent. Therefore, to verify that $(2, 1)$ is candidate for optimal solution of (P), we may now use Theorem 4.4. The KKT conditions read as follows: we are looking for $\lambda_k \geq 0, \mu_1 \in \mathbb{R}$ so that $\lambda_k g_k(2, 1) = 0$ $(k = 1, 2)$ and

$$\nabla f(2, 1) + \lambda_1 \nabla g_1(2, 1) + \lambda_2 \nabla g_2(2, 1) + \mu_1 \nabla h_1(2, 1) = [0, 0],$$

Let's address the first condition: Since $g_1(2, 1) = 0$ and $g_2(2, 1) = -14 < 0$, it must be $\lambda_2 = 0$. The second condition turns then into the equation

$$[-20, 14] + \lambda_1[7, -2.5] + 0 \cdot [-14, 2] + \mu_1[8, 4] = [0, 0]$$

FIGURE 4.3. Feasibility region for (P) in example 4.4

or equivalently

$$\begin{bmatrix} 7 & 8 \\ -2.5 & 4 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \mu_1 \end{bmatrix} = \begin{bmatrix} 20 \\ -14 \end{bmatrix},$$

which gives $\lambda_1 = 4$, and $\mu_1 = -1$. This proves that the point $(2, 1)$ is indeed a good candidate for the optimal solution of (P).

There are other instances in which the KKT conditions can be used instead of those in the Fritz John Theorem.

THEOREM 4.5 (Slater Necessary Condition). *Suppose that the inequality constraints g_k of a super-consistent program (P) are pseudo-convex $(1 \leq k \leq m)$, the equality constraints h_k are linear $(1 \leq k \leq \ell)$, and the vectors $\nabla h_k(\boldsymbol{x})$ are linearly independent at a feasible point $\boldsymbol{x}$. Then the KKT conditions (a) and (b) of Theorem 4.4 are necessary to characterize $\boldsymbol{x}$ as an optimal solution of (P).*

THEOREM 4.6. *If all constraints of a consistent program (P) are linear, then the KKT conditions (a) and (b) of Theorem 4.4 are necessary to characterize optimal solutions of (P).*

2. Sufficient Conditions

It all boils down to a single result.

THEOREM 4.7 (KKT Sufficient Conditions). *Let $\boldsymbol{x} \in S$ be a feasible point of the consistent program (P) for which there are multipliers $\lambda_k \geq 0$ $(1 \leq k \leq m)$ and $\mu_k \in \mathbb{R}$ $(1 \leq k \leq \ell)$ satisfying the conditions (a) and (b) of Theorem 4.4. If f is pseudo-convex, g_k is quasi-convex for all $1 \leq k \leq m$, and h_k is linear for all $1 \leq k \leq \ell$, then $\boldsymbol{x}$ is a global optimal solution of (P).*

EXAMPLE 4.5. We saw that the point $(0,0)$ satisfies the KKT conditions for the super-consistent convex program (P) in Example 4.1. As a consequence of Theorems 4.3 and 4.7, this point must be the optimal global minimum of (P).

We also saw that the point $(2,1)$ satisfies the KKT conditions for the program (P) in Example 4.4. It is not hard to see that this program is super-consistent, f is pseudo-convex, g_1 and g_2 are quasi-convex, and h_1 is linear. By virtue of Theorems 4.4 and 4.7, the point $(2,1)$ must be the optimal solution of (P).

Key Examples

In the following section we are going to use the KKT conditions to address the characterization of optimal solutions of generic programs.

EXAMPLE 4.6. Let $\boldsymbol{Q}$ be a symmetric $d \times d$ square matrix. Consider the associated quadratic form $\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{x})$. We wish to find the global *maximum* over all points of this function in the unit ball $\mathbb{B}_d = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\| \leq 1\}$.

An equivalent program (P) is thus defined with $f(\boldsymbol{x}) = -\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{x})$ as its objective function, and a single inequality constraint $g_1(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 - 1$. This is trivially a super-consistent program with a convex inequality constraint. Checking the KKT conditions to look for the optimal solution is thus justified under the hypothesis of Theorem 4.5. Notice that

$$\nabla f(\boldsymbol{x})^\intercal = -2\boldsymbol{Q}\boldsymbol{x}^\intercal,$$
$$\nabla g_1(\boldsymbol{x}) = 2\boldsymbol{x};$$

therefore, the KKT conditions request the search for $\boldsymbol{x} \in \mathbb{B}_d$ and $\lambda \geq 0$ so that $\lambda\big(1 - \|\boldsymbol{x}\|^2\big) = 0$ and $-2\boldsymbol{Q}\boldsymbol{x}^\intercal + 2\lambda\boldsymbol{x}^\intercal = \boldsymbol{0}^\intercal$.

It must be $\|\boldsymbol{x}\| = 1$ by the first condition. The second condition states that $\boldsymbol{x}$ must be an eigenvector of $\boldsymbol{Q}$ with eigenvalue λ: $\boldsymbol{Q}\boldsymbol{x}^\intercal = \lambda\boldsymbol{x}^\intercal$. The value of the objective function in this case is $f(\boldsymbol{x}) = -\mathcal{Q}_{\boldsymbol{Q}}(\boldsymbol{x}) = -\boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^\intercal = -\lambda\|\boldsymbol{x}\|^2 = -\lambda$. In order to obtain the requested global minimum value (different than zero), λ has to be the largest non-negative eigenvalue of $\boldsymbol{Q}$, and $\boldsymbol{x}$ its corresponding normalized eigenvector.

EXAMPLE 4.7. A simple case of the previous example: Set $\boldsymbol{Q} = \left[\begin{smallmatrix} 1 & 3 \\ 3 & 1 \end{smallmatrix}\right]$. The eigenvalues of $\boldsymbol{Q}$ are -2 and 4, and therefore the maximum of the associated quadratic form $\mathcal{Q}_{\boldsymbol{Q}}(x,y) = x^2 + y^2 + 6xy$ over the ball $x^2 + y^2 \leq 1$ happens at the (normalized) solution of the system

$$\begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 4 \begin{bmatrix} x \\ y \end{bmatrix}.$$

This gives the points $\pm(\sqrt{2}/2, \sqrt{2}/2)$.

EXAMPLE 4.8. Let $\boldsymbol{x}_0 \in \mathbb{R}^d \setminus \{\boldsymbol{0}\}$ and $r > 0$. Find the point on the sphere of radius r, $\mathbb{S}_d = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\| = r\}$ that is closer to $\boldsymbol{x}_0$.

We may write a super-consistent convex program to solve this optimization problem by using $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_0\|^2$ as objective function, and one equality constraint $h_1(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 - r^2$. With this choice, we are well within the hypothesis of Theorems 4.4 and 4.7. The KKT conditions request $\mu \in \mathbb{R}$ and a point $\boldsymbol{x} \in \mathbb{R}^d$ with $\|\boldsymbol{x}\| = r$ so that

$$\nabla f(\boldsymbol{x}) + \mu \nabla h_1(\boldsymbol{x}) = \boldsymbol{0},$$

This gives $2(\boldsymbol{x} - \boldsymbol{x}_0) + 2\mu\boldsymbol{x} = \boldsymbol{0}$, or equivalently, $(1 + \mu)\boldsymbol{x} = \boldsymbol{x}_0$.

It must be $\mu = -1 + \|\boldsymbol{x}_0\|/r$. We have then two cases:

(a) If $\|\boldsymbol{x}_0\| = r$, then $\mu = 0$ and $\boldsymbol{x} = \boldsymbol{x}_0$ is the only solution.
(b) If $\|\boldsymbol{x}_0\| \neq r$ (the point $\boldsymbol{x}_0$ is not on the sphere), then $\boldsymbol{x} = r\boldsymbol{x}_0/\|\boldsymbol{x}_0\|$.

EXAMPLE 4.9. Find the minimum value of a (real-valued) linear map over the unit ball.

Given $\boldsymbol{a} \in \mathbb{R}^d \backslash \{\boldsymbol{0}\}$, consider the corresponding linear map $\boldsymbol{L}(\boldsymbol{x}) = \langle \boldsymbol{a}, \boldsymbol{x} \rangle$. We wish to find the minimum value of $\boldsymbol{L}$ over all points in the unit ball $\mathbb{B}_d = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\| \leq 1\}$.

An equivalent program (P) is defined with $\boldsymbol{L}$ as its objective function and $g_1(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 - 1$. This is a super-consistent convex program. Checking the KKT conditions is justified under the hypothesis of Theorem 4.4. Notice that

$$\nabla \boldsymbol{L}(\boldsymbol{x}) = \boldsymbol{a},$$
$$\nabla g_1(\boldsymbol{x}) = 2\boldsymbol{x};$$

therefore, the KKT conditions request the search for $\boldsymbol{x} \in \mathbb{B}_d$ and $\lambda \geq 0$ so that $\lambda\big(\|\boldsymbol{x}\|^2 - 1\big) = 0$ and $\boldsymbol{a} + 2\lambda\boldsymbol{x} = \boldsymbol{0}$.

This first condition imposes $\|\boldsymbol{x}\| = 1$. The second condition requires $\boldsymbol{x} = -\boldsymbol{a}/(2\lambda)$. These two put together imply that it must be $\lambda = -\|\boldsymbol{a}\|/2$, and hence $\boldsymbol{x} = -\boldsymbol{a}/\|\boldsymbol{a}\|$.

Exercises

PROBLEM 4.1 (Basic). Consider the following problem: Find the global minimum of the function $f(x, y) = 6(x - 10)^2 + 4(y - 12.5)^2$ on the set $S = \{(x, y) \in \mathbb{R}^2 : x^2 + (y - 5)^2 \leq 50, x^2 + 3y^2 \leq 200, (x - 6)^2 + y^2 \leq 37\}$.

(a) Write the statement of this problem as a program with the notation from equation 22. Label the objective function, as well as the inequality constraints accordingly.
(b) Is the objective function f pseudo-convex? Why or why not?
(c) Are the inequality constraints quasi-convex? Why or why not?
(d) Sketch the feasibility region. Label all relevant objects involved.
(e) Is the point $(7, 6)$ feasible? Why or why not?
(f) Employ Theorem 4.4 to write a necessary condition for optimality and verify that is satisfied by the point $(7, 6)$.
(g) Employ Theorem 4.7 to decide whether this point is an optimal solution of (P).

PROBLEM 4.2 (Basic). [**8**, lec6_constr_opt, 10] Let $f(x,y) = (x-4)^2 + (y-6)^2$. Consider the program (P) to find the global minimum of f on the set $S = \{(x,y) \in \mathbb{R}^2 : y - x^2 \geq 0, y \leq 4\}$.

(a) Write the statement of this problem as a program with the notation from equation 22. Label the objective function, as well as the inequality constraints accordingly.
(b) Is the objective function f pseudo-convex? Why or why not?
(c) Are the inequality constraints quasi-convex? Why or why not?
(d) Sketch the feasibility region. Label all relevant objects involved.
(e) Is the point $(2,4)$ feasible? Why or why not?
(f) Employ Theorem 4.4 to write a necessary condition for optimality and verify that is satisfied by the point $(2,4)$.
(g) Employ Theorem 4.7 to decide whether this point is an optimal solution of (P).

PROBLEM 4.3 (Basic). [**8**, lec6_constr_opt, 12] Let $f(x,y) = (x-9/4)^2 + (y-2)^2$. Consider the program (P) to find the global minimum of f on the set $S = \{(x,y) \in \mathbb{R}^2 : y - x^2 \geq 0, x + y \leq 6, x \geq 0, y \geq 0\}$.

(a) Write down the KKT optimality conditions and verify that these conditions are satisfied at the point $(3/2, 9/4)$.
(b) Present a graphical interpretation of the KKT conditions at $(3/2, 9/4)$.
(c) Show that this point is the optimal solution to the program.

PROBLEM 4.4 (Basic). Find examples of non-diagonal 3×3 symmetric square matrices with integer-valued eigenvalues of each type below:

- $\boldsymbol{A}_1$ positive definite,
- $\boldsymbol{A}_2$ positive semi-definite,
- $\boldsymbol{A}_3$ negative definite,
- $\boldsymbol{A}_4$ negative semi-definite, and
- $\boldsymbol{A}_5$ indefinite.

For each of these matrices, find the maximum of their corresponding quadratic form $\mathcal{Q}_{\boldsymbol{A_k}}(x,y,z)$ over the unit ball $\mathbb{B}_3 = \{(x,y,z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 \leq 1\}$.

PROBLEM 4.5 (Advanced). [**8**, lec6_constr_opt, 19]
Arrow-Hurwicz-Uzawa constraint qualification: Consider the problem to minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in X$ (X being an open set $X \subseteq \mathbb{R}^d$) and a set of continuous inequality constraints $g_k(\boldsymbol{x}) \leq 0$ $(1 \leq k \leq m)$. Let $\mathcal{J} = \big\{k \in \{1,\ldots,m\} : g_k \text{ is pseudo-concave}\big\}$. Prove that if the set

$$\big\{\boldsymbol{x} \in X : \langle \nabla g_k(\boldsymbol{x}^\star), \boldsymbol{x}\rangle \leq 0, (k \in \mathcal{J}); \langle \nabla g_k(\boldsymbol{x}^\star), \boldsymbol{x}\rangle < 0, \big(k \in \mathcal{I}(\boldsymbol{x}^\star) \setminus \mathcal{J}\big)\big\}$$

is nonempty, then $\boldsymbol{x}^\star$ satisfies the KKT conditions.

CHAPTER 5

Numerical Approximation for Constrained Optimization

1. Projection Methods for Linear Equality constrained programs

Consider the minimization of a function $f\colon \mathbb{R}^d \to \mathbb{R}$ subject to ℓ linear constraints $h_k(\boldsymbol{x}) = 0$, where $h_k(\boldsymbol{x}) = \langle \boldsymbol{a}_k, \boldsymbol{x}\rangle - b_k$ for vectors $\boldsymbol{a}_k = [a_{k1}, \ldots, a_{kd}] \in \mathbb{R}^d$, and real values $b_k \in \mathbb{R}$ $(1 \leq k \leq \ell)$. If we set

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{\ell 1} & \cdots & a_{\ell d} \end{bmatrix}$$

and $\boldsymbol{b} = [b_1, \ldots, b_\ell]$ we may write these linear constraints as $\boldsymbol{A}\boldsymbol{x}^\intercal = \boldsymbol{b}^\intercal$.

The corresponding program (P) has f as objective function and linear equality constraints $h_k\colon \mathbb{R}^d \to \mathbb{R}$ as defined above. If this program happens to be consistent (it depends on $\boldsymbol{A}$ and $\boldsymbol{b}$), then Theorem 4.6 allows us to use KKT to find the optimal solutions.

The KKT conditions read as follows: Find $\boldsymbol{x} \in \mathbb{R}^d$ satisfying $\boldsymbol{A}\boldsymbol{x}^\intercal = \boldsymbol{b}^\intercal$, $\mu_k \in \mathbb{R}$ for $1 \leq k \leq \ell$ so that

$$\nabla f(\boldsymbol{x}) + \sum_{k=1}^{\ell} \mu_k \boldsymbol{a}_k = \boldsymbol{0}. \tag{23}$$

Example 5.1. We would like to find the minimum value of the function $f(x, y, z) = x^2 + y^2 + z^2$ over the line at the intersection of the planes $x + y + z = 0$ and $x - y + 2z = 3$.

The method we used in MATH 241 would start by computing a parameterization of the line first. For instance, by forcing $z = 0$ and solving the system formed by the two planes (with this restriction), we find that the point $(3/2, -3/2, 0)$ belongs in this line. The cross product of the normal vectors to the planes is the direction of the line:

$$[1,1,1] \times [1,-1,2] = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ 1 & 1 & 1 \\ 1 & -1 & 2 \end{vmatrix} = [3,-1,-2].$$

We have then the line with equation $(3/2 + 3t, -3/2 - t, -2t)$, $t \in \mathbb{R}$. A restriction of f on this line gives

$$\varphi(t) = f\left(\tfrac{3}{2} + 3t, -\tfrac{3}{2} - t, -2t\right)$$

$$\begin{aligned}
&= \left(\tfrac{3}{2} + 3t\right)^2 + \left(-\tfrac{3}{2} - t\right)^2 + (-2t)^2 \\
&= \tfrac{9}{4} + 9t^2 + 9t + \tfrac{9}{4} + t^2 + 3t + 4t^2 \\
&= 14t^2 + 12t + \tfrac{9}{2}
\end{aligned}$$

The minimum of this function occurs at $t = -3/7$. This yields the point

$$\left(\tfrac{3}{2} - 3 \cdot \tfrac{3}{7}, -\tfrac{3}{2} + \tfrac{3}{7}, 2 \cdot \tfrac{3}{7}\right) = \left(\tfrac{3}{14}, -\tfrac{15}{14}, \tfrac{6}{7}\right).$$

The method we have explained in Chapter 4 starts by collecting the constraints first, and claiming the use of the KKT conditions. In this case, Theorem 4.6 guarantees we may use this technique.

$$\begin{aligned}
h_1(x, y, z) &= x + y + z = \langle [1, 1, 1], [x, y, z] \rangle, \\
h_2(x, y, z) &= x - y + 2z - 3 = \langle [1, -1, 2], [x, y, z] \rangle - 3.
\end{aligned}$$

The corresponding KKT condition request feasible points $(x, y, z) \in \mathbb{R}^3$ and two real values $\mu_1, \mu_2 \in \mathbb{R}$ that satisfy:

$$\begin{aligned}
0 &= x + y + z, \\
3 &= x - y + 2z, \\
[0, 0, 0] &= [2x, 2y, 2z] + \mu_1[1, 1, 1] + \mu_2[1, -1, 2].
\end{aligned}$$

This gives the following system:

$$\begin{aligned}
0 = x + y + z &= -\tfrac{1}{2}(\mu_1 + \mu_2) - \tfrac{1}{2}(\mu_1 - \mu_2) - \tfrac{1}{2}(\mu_1 + 2\mu_2), \\
3 = x - y + 2z &= -\tfrac{1}{2}(\mu_1 + \mu_2) + \tfrac{1}{2}(\mu_1 - \mu_2) - (\mu_1 + 2\mu_2),
\end{aligned}$$

which reduces to

$$\begin{bmatrix} 3/2 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \end{bmatrix}.$$

There is a unique solution $\mu_1 = 6/7$, $\mu_2 = -9/7$, with the point $\left(\tfrac{3}{14}, -\tfrac{15}{14}, \tfrac{6}{7}\right)$.

Notice how, in either case above, we ended solving the optimization problem *symbolically*. In this section we are going to adapt the techniques we learned in Chapter 3 to approximate the solution of this kind of programs numerically. As we have done in the past, it all starts by selecting a feasible initial guess, and solving a related program, associated to a simpler approximation instead. We explore the following options:

- Using linear approximations (this leads to the *steepest descent method*).
- Using quadratic approximations (this leads to the *Newton method*).

1.1. Steepest Descent. Given a feasible initial guess $\boldsymbol{x}_0 \in \mathbb{R}^d$ (satisfying $\boldsymbol{A}\boldsymbol{x}_0^\intercal = \boldsymbol{b}^\intercal$), we proceed to search for a next iteration $\bar{\boldsymbol{x}}_0$ *within the feasibility region* that minimizes the linear approximant $L_0(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0)^\intercal$. We usually further impose a maximum distance from $\boldsymbol{x}_0$

to $\bar{\boldsymbol{x}}_0$; that is, we force for example $\|\bar{\boldsymbol{x}}_0 - \boldsymbol{x}_0\| \leq R_0$ for some $R_0 > 0$. This gives the *Direction Finding Program* (DFP)

$$\min_{\boldsymbol{x}\in S_0} L_0(\boldsymbol{x}), \quad S_0 = \{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{A}\boldsymbol{x}^\intercal = \boldsymbol{b}^\intercal, \|\boldsymbol{x} - \boldsymbol{x}_0\| \leq R_0\}.$$

If we set $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{x}_0$, the program translates into simpler terms:

$$\min_{\boldsymbol{v}\in S_0'} \nabla f(\boldsymbol{x}_0)\boldsymbol{v}^\intercal, \quad S_0' = \{\boldsymbol{A}\boldsymbol{v}^\intercal = \boldsymbol{0}^\intercal, \|\boldsymbol{v}\| \leq R_0\}.$$

Note this is a convex program (it is linear, as a matter of fact) with a Slater point at $\boldsymbol{0}$. The KKT conditions for the (DFP) program are therefore necessary and sufficient for optimality.

Notice how much simpler these conditions are: Set $g_1(\boldsymbol{v}) = \|\boldsymbol{v}\|^2 - R_0^2$, $h_k(\boldsymbol{v}) = \langle \boldsymbol{a}_k, \boldsymbol{v}\rangle$ for $1 \leq k \leq \ell$. Find $\boldsymbol{v} \in \mathbb{R}^d$ with $\boldsymbol{A}\boldsymbol{v}^\intercal = \boldsymbol{0}^\intercal$, $\lambda_1 \geq 0$, $\mu_k \in \mathbb{R}$, $1 \leq k \leq \ell$ so that:

$$\lambda_1\big(\|\boldsymbol{v}\|^2 - R_0^2\big) = 0,$$
$$\nabla f(\boldsymbol{x}_0) + 2\lambda_1\boldsymbol{v} + \sum_{k=1}^{\ell} \mu_k \boldsymbol{a}_k = \boldsymbol{0}.$$

This gives two possibilities.

(a) $\lambda_1 = 0$, in which case we have $\nabla f(\boldsymbol{x}_0) + \sum_{k=1}^{\ell} \mu_k \boldsymbol{a}_k = 0$. This means that $\boldsymbol{x}_0$ is already a feasible point that satisfies the KKT conditions for (P).

(b) $\lambda_1 \neq 0$, and $\boldsymbol{v}$ with $\|\boldsymbol{v}\| = R_0$ that satisfies

$$2\lambda_1\boldsymbol{v} + \sum_{k=1}^{\ell} \mu_k \boldsymbol{a}_k = -\nabla f(\boldsymbol{x}_0).$$

A solution is found by the following formulas (it is easy to check why, and left to the reader):

$$\begin{aligned} \boldsymbol{P} &= R_0^2\big(\boldsymbol{I} - \boldsymbol{A}^\intercal(\boldsymbol{A}\boldsymbol{A}^\intercal)^{-1}\boldsymbol{A}\big), \\ \lambda_1 &= \tfrac{1}{2}\mathcal{Q}_{\boldsymbol{P}}\big(\nabla f(\boldsymbol{x}_0)\big)^{1/2}, \\ [\mu_1, \ldots, \mu_\ell]^\intercal &= -(\boldsymbol{A}\boldsymbol{A}^\intercal)^{-1}\big(\boldsymbol{A}\nabla f(\boldsymbol{x}_0)^\intercal\big), \\ \boldsymbol{v}^\intercal &= -\mathcal{Q}_{\boldsymbol{P}}\big(\nabla f(\boldsymbol{x}_0)\big)^{-1/2} P \nabla f(\boldsymbol{x}_0)^\intercal. \end{aligned} \tag{24}$$

The solution $\boldsymbol{v}_0$ of the (DFP) points to a *direction of steepest descent* for the program (P) starting at $\boldsymbol{x}_0$. We proceed to restrict f on the half line starting at $\boldsymbol{x}_0$ in the direction given by $\boldsymbol{v}_0$, and search for a global minimum:

$$t_0 = \operatorname*{argmin}_{t\geq 0} \varphi_0(t) = \operatorname*{argmin}_{t\geq 0} f(\boldsymbol{x}_0 + t\boldsymbol{v}_0).$$

Set $\boldsymbol{x}_1 = \boldsymbol{x}_0 + t_0\boldsymbol{v}_0$. We iterate this process to obtain a sequence of feasible points. Given $\boldsymbol{x}_n \in \mathbb{R}^d$ a feasible guess, and $R_n > 0$:

$$\boldsymbol{v}_n = \operatorname*{argmin}_{\boldsymbol{v}\in S_n'} \nabla f(\boldsymbol{x}_n)\boldsymbol{v}^\intercal, \quad S_n' = \{\boldsymbol{v} \in \mathbb{R}^d : \|\boldsymbol{v}\| \leq R_n, \boldsymbol{A}\boldsymbol{v}^\intercal = \boldsymbol{0}^\intercal\}$$

$$t_n = \operatorname*{argmin}_{t\geq 0} \varphi_n(t) = \operatorname{argmin} t \geq 0 f(\boldsymbol{x}_n + t\boldsymbol{v}_n)$$
$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + t_n \boldsymbol{v}_n.$$

EXAMPLE 5.2. Let's illustrate this technique with the running example 5.1. Assume the initial guess is the feasible point $(3/2, -3/2, 0)$. The corresponding (DFP) for a direction search with unit vectors reads as follows:

$$\min_{\boldsymbol{v}\in S'} \langle [3,-3,0], \boldsymbol{v}\rangle, \quad S' = \left\{\boldsymbol{v} \in \mathbb{R}^3 : \|\boldsymbol{v}\| \leq 1, \left[\begin{smallmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \end{smallmatrix}\right] \boldsymbol{v}^\intercal = \boldsymbol{0}^\intercal\right\}$$

The KKT conditions request a feasible $\boldsymbol{v} \in \mathbb{R}^3$, $\lambda_1 \geq 0$, $\mu_1, \mu_2 \in \mathbb{R}$ so that

$$\lambda_1(\|\boldsymbol{v}\|^2 - 1) = 0,$$
$$2\lambda_1 \boldsymbol{v} + \mu_1[1,1,1] + \mu_2[1,-1,2] = [-3,3,0].$$

The formulas in (24) give

$$\begin{aligned}
\boldsymbol{P} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3/7 & -1/7 \\ -1/7 & 3/14 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 2/7 & 4/7 & 1/7 \\ 1/14 & -5/14 & 2/7 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 5/14 & 3/14 & 3/7 \\ -3/14 & 1/14 & 1/7 \\ -3/7 & 1/7 & 2/7 \end{bmatrix} = \begin{bmatrix} 9/14 & -3/14 & -3/7 \\ -3/14 & 1/14 & 1/7 \\ -3/7 & 1/7 & 2/7 \end{bmatrix}
\end{aligned}$$

and thus,

$$\begin{aligned}
\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} &= \left(\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} \right)^{-1} \left(\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ -3 \\ 0 \end{bmatrix} \right) \\
&= \begin{bmatrix} 3/7 & -1/7 \\ -1/7 & 3/14 \end{bmatrix} \begin{bmatrix} 0 \\ 6 \end{bmatrix} = \begin{bmatrix} -6/7 \\ 9/7 \end{bmatrix}, \\
\lambda_1 &= \frac{1}{2} \left(\begin{bmatrix} 3 & -3 & 0 \end{bmatrix} \begin{bmatrix} 9/14 & -3/14 & -3/7 \\ -3/14 & 1/14 & 1/7 \\ -3/7 & 1/7 & 2/7 \end{bmatrix} \begin{bmatrix} 3 \\ -3 \\ 0 \end{bmatrix} \right)^{1/2} = \tfrac{3}{7}\sqrt{14} \\
\boldsymbol{v}_0{}^\intercal &= -\tfrac{1}{12}\sqrt{14} \begin{bmatrix} 9/14 & -3/14 & -3/7 \\ -3/14 & 1/14 & 1/7 \\ -3/7 & 1/7 & 2/7 \end{bmatrix} \begin{bmatrix} 3 \\ -3 \\ 0 \end{bmatrix}
\end{aligned}$$

$$= -\tfrac{1}{12}\sqrt{14}\begin{bmatrix} 18/7 \\ -6/7 \\ -12/7 \end{bmatrix} = \tfrac{1}{14}\sqrt{14}\begin{bmatrix} -3 \\ 1 \\ 2 \end{bmatrix}$$

Notice how $\boldsymbol{v}_0$ satisfies all required constraints, and $\|\boldsymbol{v}_0\| = 1$. This vector is an optimal solution of (DFP), and therefore a *direction of steepest descent* for (P) from the point $(3/2, -3/2, 0)$.

We perform now the line search from $\boldsymbol{x}_0$ in this direction:

$$\varphi_0(t) = f(\boldsymbol{x}_0 + t\boldsymbol{v}_0) = t^2 - \tfrac{6}{7}\sqrt{14}\,t + \tfrac{9}{2}$$
$$t_0 = \operatorname*{argmin}_{t\geq 0} \varphi_0(t) = \tfrac{3}{7}\sqrt{14}$$

We have then

$$\boldsymbol{x}_1 = \underbrace{\big(\tfrac{3}{2}, -\tfrac{3}{2}, 0\big)}_{\boldsymbol{x}_0} + \underbrace{\tfrac{3}{7}\sqrt{14}}_{t_0}\,\underbrace{\tfrac{1}{14}\sqrt{14}\big[-3, 1, 2\big]}_{\boldsymbol{v}_0} = \big(\tfrac{3}{14}, -\tfrac{15}{14}, \tfrac{6}{7}\big),$$

which happens to be the optimal solution of the program (P).

1.2. Newton-Raphson. Given a feasible initial guess $\boldsymbol{x}_0 \in \mathbb{R}^d$ (satisfying $\boldsymbol{A}\boldsymbol{x}_0{}^{\intercal} = \boldsymbol{b}^{\intercal}$), we proceed to search for a next iteration $\boldsymbol{x}_1$ *within the feasibility region* that minimizes the quadratic approximant

$$Q_0(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \langle \nabla f(\boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{x}_0\rangle + \tfrac{1}{2}\mathcal{Q}_{\mathbf{Hess}\boldsymbol{f}(\boldsymbol{x_0})}(\boldsymbol{x} - \boldsymbol{x}_0).$$

No further imposition on the distance between $\boldsymbol{x}_1$ and $\boldsymbol{x}_0$ is needed at this point. The corresponding associated program (P') becomes

$$\min_{\boldsymbol{x}\in S} Q_0(\boldsymbol{x}), \quad S = \{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{A}\boldsymbol{x}^{\intercal} = \boldsymbol{b}^{\intercal}\}.$$

Notice that $\nabla Q_0(\boldsymbol{x}) = \nabla f(\boldsymbol{x}_0) + \langle \mathrm{Hess} f(\boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{x}_0\rangle$. Replacing $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{x}_0$, the KKT conditions for the program (P') read as follows: Find $\mu_k \in \mathbb{R}$ $(1 \leq k \leq \ell)$ and $\boldsymbol{v} \in \mathbb{R}^d$ satisfying $\boldsymbol{A}\boldsymbol{v}^{\intercal} = \boldsymbol{0}^{\intercal}$ so that

$$\mathrm{Hess} f(\boldsymbol{x}_0)\cdot \boldsymbol{v}^{\intercal} + \sum_{k=1}^{\ell} \mu_k \boldsymbol{a}_k{}^{\intercal} = -\nabla f(\boldsymbol{x}_0)^{\intercal}.$$

If $\det \mathrm{Hess} f(\boldsymbol{x}_0) \neq 0$, the system has a unique solution. To compute it, set first $\boldsymbol{H} = [\mathrm{Hess} f(\boldsymbol{x}_0)]^{-1}$. The solution in this case can be written according to the following formulas:

$$\begin{aligned} \boldsymbol{v}_0{}^{\intercal} &= \big[\big(\boldsymbol{H}\boldsymbol{A}^{\intercal}\big)\big(\boldsymbol{A}\boldsymbol{H}\boldsymbol{A}^{\intercal}\big)^{-1}\big(\boldsymbol{A}\boldsymbol{H}\big) - \boldsymbol{H}\big]\cdot \nabla f(\boldsymbol{x}_0)^{\intercal} \\ [\mu_1, \ldots, \mu_\ell]^{\intercal} &= -\big[\big(\boldsymbol{A}\boldsymbol{H}\boldsymbol{A}^{\intercal}\big)^{-1}\big(\boldsymbol{A}\boldsymbol{H}\big)\big]\cdot \nabla f(\boldsymbol{x}_0)^{\intercal} \end{aligned} \tag{25}$$

The optimal solution $\boldsymbol{v}_0$ of this system points to a *Newton-Raphson direction* for the program (P) starting at $\boldsymbol{x}_0$. Set then $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{v}_0$. We iterate this process to find a sequence of feasible points. Given $\boldsymbol{x}_n \in \mathbb{R}^d$ a feasible guess,

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \underbrace{\operatorname*{argmin}_{\boldsymbol{A}\boldsymbol{v}^{\intercal}=\boldsymbol{0}^{\intercal}} \big(f(\boldsymbol{x}_n) + \langle \nabla f(\boldsymbol{x}_n), \boldsymbol{v}\rangle + \tfrac{1}{2}\mathcal{Q}_{\mathbf{Hess}\boldsymbol{f}(\boldsymbol{x_n})}(\boldsymbol{v})\big)}_{\boldsymbol{v}_n}.$$

EXAMPLE 5.3. Let's illustrate this technique with the running example 5.1. Assume once again that the initial guess is the feasible point $(3/2, -3/2, 0)$. The corresponding program (P') to search for the Newton-Raphson direction is

$$\min_{\boldsymbol{v}\in S}\left(\tfrac{9}{2} + \langle[3,-3,0],\boldsymbol{v}\rangle + \boldsymbol{v}\boldsymbol{v}^{\intercal}\right), \quad S = \left\{\boldsymbol{v}\in\mathbb{R}^3 : \left[\begin{smallmatrix}1 & 1 & 1\\ 1 & -1 & 2\end{smallmatrix}\right]\boldsymbol{v}^{\intercal} = \boldsymbol{0}^{\intercal}\right\}$$

The KKT conditions request multipliers $\mu_1, \mu_2 \in \mathbb{R}$ and a feasible $\boldsymbol{v}\in\mathbb{R}^3$ so that

$$2\boldsymbol{v} + \mu_1[1,1,1] + \mu_2[1,-1,2] = [-3,3,0].$$

Since $\operatorname{Hess} f\left(\frac{3}{2}, -\frac{3}{2}, 0\right) = 2\boldsymbol{I}_3$ (non-singular), we have $\boldsymbol{H} = \frac{1}{2}\boldsymbol{I}_3$. The formulas in (25) give

$$\begin{aligned}
\boldsymbol{v}_0^{\intercal} &= \left[\begin{bmatrix}1/2 & 1/2\\ 1/2 & -1/2\\ 1/2 & 1\end{bmatrix}\begin{bmatrix}3/2 & 1\\ 1 & 3\end{bmatrix}^{-1}\begin{bmatrix}1/2 & 1/2 & 1/2\\ 1/2 & -1/2 & 1\end{bmatrix} - \tfrac{1}{2}\boldsymbol{I}_3\right]\begin{bmatrix}3\\ -3\\ 0\end{bmatrix}\\
&= \begin{bmatrix}-9/28 & 3/28 & 3/14\\ -3/28 & -1/28 & -1/14\\ 3/14 & -1/14 & -1/7\end{bmatrix}\begin{bmatrix}3\\ -3\\ 0\end{bmatrix} = \begin{bmatrix}-9/7\\ 3/7\\ 6/7\end{bmatrix}\\
\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix} &= -\begin{bmatrix}3/2 & 1\\ 1 & 3\end{bmatrix}^{-1}\begin{bmatrix}1/2 & 1/2 & 1/2\\ 1/2 & -1/2 & 1\end{bmatrix}\begin{bmatrix}3\\ -3\\ 0\end{bmatrix}\\
&= -\begin{bmatrix}2/7 & 4/7 & 1/7\\ 1/14 & -5/14 & 2/7\end{bmatrix}\begin{bmatrix}3\\ -3\\ 0\end{bmatrix} = \begin{bmatrix}6/7\\ -9/7\end{bmatrix}
\end{aligned}$$

At this point, we may calculate

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{v}_0 = \left(\tfrac{3}{2}, -\tfrac{3}{2}, 0\right) + \left[-\tfrac{9}{7}, \tfrac{3}{7}, \tfrac{6}{7}\right] = \left(\tfrac{3}{14}, -\tfrac{15}{14}, \tfrac{6}{7}\right),$$

which happens to be the optimal solution of the program (P).

2. Linear Programing: The simplex method

A general linear program (LP) is usually expressed in the following standard form:

$$(LP): \begin{cases}\max\limits_{\boldsymbol{x}\in\mathbb{R}^d}\langle \boldsymbol{c}, \boldsymbol{x}\rangle & \\ \langle \boldsymbol{a}_j, \boldsymbol{x}\rangle = b_j & (1\le k\le \ell)\\ x_k \ge 0 & (1\le k\le d)\end{cases}$$

for a given $\boldsymbol{c} = [c_1,\dots,c_d], \boldsymbol{a}_k = [a_{k1},\dots,a_{kd}] \in \mathbb{R}^d$, $b_k\in\mathbb{R}$ for $1\le k\le \ell$. We accomplish this by performing the following operations, where necessary:

Objective function: If the original program requests $\min f(\boldsymbol{x})$, convert it to $\max\left(-f(\boldsymbol{x})\right)$.

Slack variables: If the original program contains an inequality constraint of the form $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \leq b$ with $\boldsymbol{a} = [a_1, \ldots, a_d]$, convert it to an equality constraint by adding a non-negative *slack variable* s. The resulting constraint is

$$a_1 x_1 + \cdots + a_d x_d + s = b, \quad s \geq 0.$$

Surplus variables: If the original program contains an inequality constraint of the form $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \geq b$, convert it to an equality constraint by subtracting a non-negative *surplus variable* s. The resulting constraint is

$$a_1 x_1 + \cdots + a_d x_d - s = b, \quad s \geq 0.$$

Unrestricted variables in sign: If some variable x_k is unrestricted in sign, replace it everywhere in the formulation by $x_k^+ - x_k^-$, where $x_k^+ \geq 0$ and $x_k^- \geq 0$.

Once in standard form, we usually represent a linear program by its corresponding *tableau*:

$$\begin{bmatrix} 1 & -\boldsymbol{c} & 0 \\ \boldsymbol{0}^\intercal & \boldsymbol{A} & \boldsymbol{b}^\intercal \end{bmatrix},$$

where

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{\ell 1} & \cdots & a_{\ell d} \end{bmatrix}$$

and $\boldsymbol{b} = [b_1, \ldots, b_\ell]$.

EXAMPLE 5.4. Consider the linear program (LP) given by the no-standard formulation

$$(LP) : \begin{cases} \min\limits_{(x,y,z)\in\mathbb{R}^3} \big(3y - 2x\big) \\ x - 3y + 2z \leq 3, \\ 2y - x \geq 2, \\ y \geq 0, z \geq 0 \end{cases}$$

We can easily convert the objective function to a maximum, and the first two inequality constraints into equality constraints by introducing a slack variable $s_1 \geq 0$ and a surplus variable $s_2 \geq 0$. Notice that the variable x is unrestricted in sign. We convert it to two non-negative variables $x^+ - x^-$:

$$\begin{cases} \max\limits_{(x,y,z)\in\mathbb{R}^3} & \big(2x^+ & -2x^- & -3y\big) & & & & & \\ & x^+ & -x_- & -3y & +2z & +s_1 & & = 3, \\ & -x^+ & +x^- & +2y & & & -s_2 & = 2, \\ & x^+ \geq 0, & x^- \geq 0, & y \geq 0, & z \geq 0, & s_1 \geq 0, & s_2 \geq 0 & \end{cases}$$

The corresponding tableau is as follows:

1	-2	2	3	0	0	0	0
0	1	-1	-3	2	1	0	3
0	-1	1	2	0	0	-1	2
z	x^+	x^-	y	z	s_1	s_2	

The *Simplex method* to find the optimal solution of a linear program in standard form is based on the following two rules:

Rule 1: If all variables have a nonnegative coefficient on the first row, the current basic solution (the last column) is optimal. Otherwise, pick a variable x_k with a negative coefficient $(-c_k)$ in the first row—the *entering variable*—and *pivot* it with another row.

Rule 2: The selection of row to perform the pivot for the entering variable x_k is performed by choosing among the rows j for which $a_{jk} > 0$, the one with the minimum ratio b_j/a_{jk}.

EXAMPLE 5.5. Let's illustrate this process with the following linear program

$$(LP): \begin{cases} \max\limits_{(x,y)\in\mathbb{R}^2} (x+y) \\ 2x+y \leq 4 \\ x+2y \leq 3 \\ x \geq 0, y \geq 0 \end{cases}$$

We start by converting to standard. For ease of computations below, we first rename $x = x_1$ and $y = x_2$. We introduce the slack variables x_3 and x_4 as they are needed. We finish the preparation step by finding the tableau of this program.

$$(LP): \begin{cases} \max\limits_{(x,y)\in\mathbb{R}^2} & (x_1 & +x_2) & & & \\ & 2x_1 & +x_2 & +x_3 & & = 4 \\ & x_1 & +2x_2 & & +x_4 & = 3 \\ & x_1 \geq 0, & x_2 \geq 0, & x_3 \geq 0, & x_4 \geq 0 & \end{cases}$$

$$\left[\begin{array}{cccccc} 1 & \mathbf{-1} & -1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 & 0 & 4 \\ 0 & 1 & 2 & 0 & 1 & 3 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right]$$

At this stage, we have the following initial situation:

$$z = 0 \qquad \underbrace{x_3 = 4, \quad x_4 = 3}_{\text{basic solutions}} \qquad \underbrace{x_1 = x_2 = 0}_{\text{non-basic solutions}}$$

The first entering variable is x_1. We have two choices to pivot. Rule 2 indicates that we must use the second row, since for this row, we have the

ratio $4/2 = 2$, while the third row offers a bigger ratio: $3/1 = 3$.

$$\left[\begin{array}{cccccc} 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1/2 & 1/2 & 0 & 2 \\ 0 & 1 & 2 & 0 & 1 & 3 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right] \rightarrow \left[\begin{array}{cccccc} 1 & 0 & \mathbf{-1/2} & 1/2 & 0 & 2 \\ 0 & 1 & 1/2 & 1/2 & 0 & 2 \\ 0 & 0 & 3/2 & -1/2 & 1 & 1 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right]$$

At this stage, we have the following situation:

$$z = 2 \qquad \underbrace{x_1 = 2, \quad x_4 = 1}_{\text{basic solution}} \qquad \underbrace{x_2 = x_3 = 0}_{\text{non-basic solution}}$$

The second entering variable is x_2. We again have two choices to pivot. Rule 2 indicates that we must use the third row, since for this row the ratio is $1/(3/2) = 2/3$. For the second row, the ratio is larger: $2/(1/2) = 4$.

$$\left[\begin{array}{cccccc} 1 & 0 & -1/2 & 1/2 & 0 & 2 \\ 0 & 1 & 1/2 & 1/2 & 0 & 2 \\ 0 & 0 & 1 & -1/3 & 2/3 & 2/3 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right] \rightarrow \left[\begin{array}{cccccc} 1 & 0 & 0 & 1/3 & 1/3 & 7/3 \\ 0 & 1 & 0 & 2/3 & -1/3 & 5/3 \\ 0 & 0 & 1 & -1/3 & 2/3 & 2/3 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right]$$

There are no more negative coefficients on the first row. This leads to an solution of the tableau given by $z = 7/3$, $x_1 = 5/3$. $x_2 = 2/3$, $x_3 = x_4 = 0$. The global maximum of the function $f(x, y) = x+y$ on the set $S = \{(x, y) \in \mathbb{R}^2 : x \geq 0, y \geq 0, 2x + y \leq 4, x + 2y \leq 3\}$ is attained at the point $(5/3, 2/3)$. The corresponding maximum value is thus $7/3$. An illustration of the three steps carried in these computations can be observed in Figure 5.1.

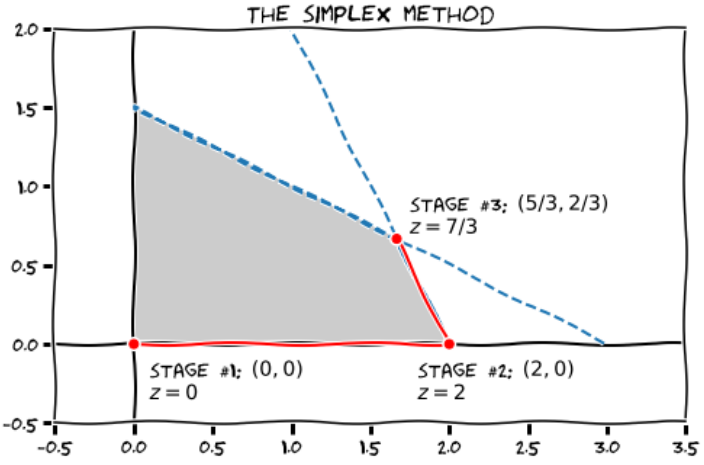


FIGURE 5.1. Illustration of the simplex method for Example 5.4

EXAMPLE 5.6. What happens if the program does not have a unique solution? Can the simplex method offer this information? Consider for

$f(x,y) = x + \frac{1}{2}y$ the program

$$(LP): \begin{cases} \max\limits_{(x,y)\in\mathbb{R}^s} f(x,y) \\ 2x + y \leq 4 \\ x + 2y \leq 3 \\ x \geq 0, y \geq 0 \end{cases}$$

Once in standard form, this program has the tableau

$$\left[\begin{array}{cccccc} 1 & -1 & -1/2 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 & 0 & 4 \\ 0 & 1 & 2 & 0 & 1 & 3 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right]$$

The initial solution gives

$$z = 0 \qquad \underbrace{x_3 = 4, \quad x_4 = 3}_{\text{basic solutions}} \qquad \underbrace{x_1 = x_2 = 0}_{\text{non-basic solutions}}$$

There is an entering variable at x_1, that has to be pivoted with the second row:

$$\left[\begin{array}{cccccc} 1 & -1 & -1/2 & 0 & 0 & 0 \\ 0 & 1 & 1/2 & 1/2 & 0 & 2 \\ 0 & 1 & 2 & 0 & 1 & 3 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right] \rightarrow \left[\begin{array}{cccccc} 1 & 0 & 0 & 1/2 & 0 & 2 \\ 0 & 1 & 1/2 & 1/2 & 0 & 2 \\ 0 & 0 & 3/2 & -1/2 & 1 & 1 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right]$$

The solution at this stage—which already offers an optimal solution of the program (LP)—gives

$$z = 2 \qquad \underbrace{x_1 = 2, \quad x_4 = 1}_{\text{basic solutions}} \qquad \underbrace{x_2 = x_3 = 0}_{\text{non-basic solutions}}$$

Notice now that at this point we could increase the value of the coefficient of the variable x_2 without changing the value of z.

$$\left[\begin{array}{cccccc} 1 & 0 & 0 & 1/2 & 0 & 2 \\ 0 & 1 & 1/2 & 1/2 & 0 & 2 \\ 0 & 0 & 1 & -1/3 & 2/3 & 2/3 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right] \rightarrow \left[\begin{array}{cccccc} 1 & 0 & 0 & 1/2 & 0 & 2 \\ 0 & 1 & 0 & 2/3 & -1/3 & 5/3 \\ 0 & 0 & 1 & -1/3 & 2/3 & 2/3 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right]$$

The solution at this stage gives

$$z = 2 \qquad \underbrace{x_1 = 5/3, \quad x_2 = 2/3}_{\text{basic solutions}} \qquad \underbrace{x_3 = x_4 = 0}_{\text{non-basic solutions}}$$

We have found two different optimal solutions of the program (LP) using the simplex method: $(2,0)$ and $(5/3, 2/3)$. Notice that in this case, any other point in the segment joining those two points, must also be a solution. Namely: for any $t \in [0,1]$, the point $\left(2 - \frac{1}{3}t, \frac{2}{3}t\right)$ satisfies

$f\left(2 - \frac{1}{3}t, \frac{2}{3}t\right) = 2 - \frac{1}{3}t + 2\frac{1}{3}t = 2.$ (The value is always 4)

$2\left(2 - \frac{1}{3}t\right) + \frac{2}{3}t = 4$ (The first constraint is satisfied)

$2 - \frac{1}{3}t + 2\frac{2}{3}t = 2 + t \leq 3$ (The second constraint is satisfied)

$2 - \frac{1}{3}t \geq \frac{5}{3} > 0$ (The third constraint is satisfied)

$\frac{2}{3}t \geq 0$ (The fourth constraint is satisfied)

EXAMPLE 5.7. What happens if we are unable to employ Rule 1 from the simplex method? This situation arises on unbounded programs. Consider the one below:

$$(LP): \begin{cases} \max\limits_{(x,y)\in\mathbb{R}^2} (2x+y) \\ -x + y \leq 1 \\ x - 2y \leq 2 \\ x \geq 0, y \geq 0 \end{cases}$$

Once in standard form, its tableau is as follows:

$$\left[\begin{array}{cccccc} 1 & -2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 & 1 \\ 0 & 1 & -2 & 0 & 1 & 2 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right]$$

At this stage, an initial solution is given by

$$z = 0 \qquad \underbrace{x_3 = 1, \quad x_4 = 2}_{\text{basic solutions}} \qquad \underbrace{x_1 = x_2 = 0}_{\text{non-basic solutions}}$$

The entering variable x_1 must be pivoted with the third row.

$$\left[\begin{array}{cccccc} 1 & 0 & -5 & 0 & 2 & 4 \\ 0 & 0 & -1 & 1 & 1 & 3 \\ 0 & 1 & -2 & 0 & 1 & 2 \\ \hline z & x_1 & x_2 & x_3 & x_4 & \end{array}\right]$$

At this stage, we have

$$z = 4 \qquad \underbrace{x_1 = 2, \quad x_3 = 3}_{\text{basic solution}} \qquad \underbrace{x_2 = x_4 = 0}_{\text{non-basic solutions}}$$

But notice that at this new stage we are unable to apply rule 1, since there are no positive coefficients for x_2. Any pivot operation that we apply using the second row will change the values of z; in particular, we should be able to perform enough changes to make z as large as we desire. For instance: what row operations would you perform to get $z = 19$, with the feasible point $(8, 3)$?

EXAMPLE 5.8. In Python there is an implementation of the simplex algorithm in the libraries `sympy.optimize`: the routine `linprog` with the option `method='simplex'`. It is smart enough to indicate, among other things,

- If the program terminates successfully, is infeasible or unbounded.
- The value of the slack or surplus variables, when these are used.

- In case of failure, the coordinates of the last point obtained by the algorithm.
- The different tableau computed on each stage.
- The index of the tableau selected as pivot on each stage.

The following session illustrates how to use it to solve some of the examples we have shown in this section.

```
import numpy as np, matplotlib.pyplot as plt
from scipy.optimize import linprog, lingrog_verbose_callback

# First program: max(x + y) with 2x + y ≤ 4, x + 2y ≤ 3, x ≥ 0, y ≥ 0
# We obtained the value 7/3 at the point (5/3, 2/3)
c1 = [-1, -1]
A1 = [[2,1], [1,2]]
b1 = [4,3]
x0_bnds = (0, None) # This means literally 0 ≤ x_0 < ∞
x1_bnds = (0, None) # and this, 0 ≤ x_1 < ∞

# Second program: max(2x + y) with −x + y ≤ 1, x − 2y ≤ 2, x ≥ 0, y ≥ 0
# The program is unbounded
c2 = [-2, -1]
A2 = [[-1, 1], [1, -2]]
b2 = [1, 2]
```

We call the optimization `linprog` with the values of $\boldsymbol{c}$ first (as a list), then the values of the matrix $\boldsymbol{A}$ (as a list of lists), and the values of $\boldsymbol{b}$ (as a list). We use the option `bounds=` to provide a collection of bounds `(min_value, max_value)` for each of the relevant variables. For instance if we request the first variable, x_0 to satisfy $a \leq x_0 \leq b$, we input `(a,b)`. If any of the bounds is infinite, we signal it with `None`. A simple output looks like these.

```
>>> linprog(c1, A1, b1, bounds=(x0_bnds, x1_bnds), method='simplex')
     fun: -2.3333333333333335
 message: 'Optimization terminated successfully.'
     nit: 2
   slack: array([ 0.,  0.])
  status: 0
 success: True
       x: array([ 1.66666667,  0.66666667])

>>> linprog(c2, A2, b2, bounds=(x0_bnds, x1_bnds), method='simplex')
     fun: -4.0
 message: 'Optimization failed. The problem appears to be unbounded.'
     nit: 1
   slack: array([ 3.,  0.])
  status: 3
 success: False
       x: array([ 2.,  0.])
```

For extra information, we issue the option `callback=` with a user-defined callback function, or the default callback `linprog_verbose_callback` already defined in `scipy.optimize`

```
>>> linprog(c1, A1, b1, bounds=(x0_bounds, x1_bounds), \
...         method='simplex', callback=linprog_verbose_callback)
--------- Initial Tableau - Phase 1 ----------

[[      2.0000       1.0000       1.0000       0.0000       4.0000]
 [      1.0000       2.0000       0.0000       1.0000       3.0000]
 [     -1.0000      -1.0000       0.0000       0.0000       0.0000]
 [      0.0000       0.0000       0.0000       0.0000       0.0000]]

Pivot Element: T[nan, nan]

Basic Variables: [2 3]

Current Solution:
x =  [      0.0000       0.0000]

Current Objective Value:
f =  -0.0

--------- Initial Tableau - Phase 2 ----------

[[      2.0000       1.0000       1.0000       0.0000       4.0000]
 [      1.0000       2.0000       0.0000       1.0000       3.0000]
 [     -1.0000      -1.0000       0.0000       0.0000       0.0000]]

Pivot Element: T[0, 0]

Basic Variables: [2 3]

Current Solution:
x =  [      0.0000       0.0000]

Current Objective Value:
f =  -0.0

--------- Iteration 1  - Phase 2 --------

Tableau:
[[      1.0000       0.5000       0.5000       0.0000       2.0000]
 [      0.0000       1.5000      -0.5000       1.0000       1.0000]
 [      0.0000      -0.5000       0.5000       0.0000       2.0000]]

Pivot Element: T[1, 1]

Basic Variables: [0 3]

Current Solution:
x =  [      2.0000       0.0000]
```

```
Current Objective Value:
f =  -2.0

--------- Iteration Complete - Phase 2 -------

Tableau:
[[      1.0000       0.0000       0.6667      -0.3333       1.6667]
 [      0.0000       1.0000      -0.3333       0.6667       0.6667]
 [      0.0000       0.0000       0.3333       0.3333       2.3333]]

Basic Variables: [0 1]

Current Solution:
x =  [      1.6667       0.6667]

Current Objective Value:
f =  -2.33333333333
```

3. The Frank-Wolfe Method

Also known as the *Conditional-Gradient method*, it is widely used to solve programs where the feasibility region S is described by a system of linear inequalities.

$$(P): \begin{cases} \min\limits_{\boldsymbol{x}\in S} f(\boldsymbol{x}) \\ S = \{\boldsymbol{x} \in \mathbb{R}^d : \langle \boldsymbol{a}_k, \boldsymbol{x}\rangle \leq b_k \text{ with } \boldsymbol{a}_k \in \mathbb{R}^d, b_k \in \mathbb{R}, 1 \leq k \leq \ell\} \end{cases}$$

This is an iterative method that, at any feasible initial guess $\boldsymbol{x}_0 \in S$, considers an associated linear program (LP_0) that minimizes the linear approximation $L_0(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \langle \nabla f(\boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{x}_0\rangle$ on the same feasible region S.

$$(LP_0): \begin{cases} \min\limits_{\boldsymbol{x}\in S} \langle \nabla f(\boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{x}_0\rangle \\ S = \{\boldsymbol{x} \in \mathbb{R}^d : \langle \boldsymbol{a}_k, \boldsymbol{x}\rangle \leq b_k \text{ with } \boldsymbol{a}_k \in \mathbb{R}^d, b_k \in \mathbb{R}, 1 \leq k \leq \ell\} \end{cases}$$

Once an optimal solution $\bar{\boldsymbol{x}}_0$ of (LP_0) has been obtained, a line-search is performed on the segment joining $\boldsymbol{x}_0$ with $\bar{\boldsymbol{x}}_0$ (which by hypothesis is contained in the feasibility region S).

$$t_0 = \operatorname*{argmin}_{0\leq t\leq 1} f\big(\boldsymbol{x}_0 + t(\bar{\boldsymbol{x}}_0 - \boldsymbol{x}_0)\big).$$

Set then $x_1 = \boldsymbol{x}_0 + t_0(\bar{\boldsymbol{x}}_0 - \boldsymbol{x}_0)$. We repeat this process to obtain a sequence $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ of feasible points.

We usually devise a *stopping criteria* (given a fixed tolerance $\varepsilon > 0$) to guarantee that we are close enough to the optimal solution of (P). An example of such a process is illustrated below:

Initialization: Let $\boldsymbol{x}_0 \in S$ be an initial feasible guess, and set $LB = -\infty$, $UB = f(\boldsymbol{x}_0)$—the *lower* and *upper* bounds (respectively) for the stopping criteria.

Iteration: Assume we have $\boldsymbol{x}_n$, LB, UB. Set

$$L_n(\boldsymbol{x}) = f(\boldsymbol{x}_n) + \langle \nabla f(\boldsymbol{x}_n), \boldsymbol{x} - \boldsymbol{x}_n \rangle.$$

Find

$$\begin{aligned}
\bar{\boldsymbol{x}}_n &= \operatorname*{argmin}_{\boldsymbol{x}\in S} L_n(\boldsymbol{x}) = \operatorname*{argmin}_{\boldsymbol{x}\in S} \langle \nabla f(\boldsymbol{x}_n), \boldsymbol{x} - \boldsymbol{x}_n \rangle, \\
\xi_n &= \min_{\boldsymbol{x}\in S} L_n(\boldsymbol{x}) = L_n(\bar{\boldsymbol{x}}_n), \\
t_n &= \operatorname*{argmin}_{0\le t\le 1} f\big(\boldsymbol{x}_n + t(\bar{\boldsymbol{x}}_n - \boldsymbol{x}_n)\big), \\
\boldsymbol{x}_{n+1} &= \boldsymbol{x}_n + t_n(\bar{\boldsymbol{x}}_n - \boldsymbol{x}_n), \\
LB &= \max(LB, \xi_n)
\end{aligned}$$

Stopping Criteria: If $|UB - LB| \le \varepsilon$, then stop. Otherwise, update the upper bound, $UB = f(\boldsymbol{x}_{n+1})$, and perform the next **Iteration**.

EXAMPLE 5.9. Let us use this technique to try and find the minimum value of the function $f(x, y) = (x-3)^2 + (y-2)^2$ over the square with vertices at $(-3/2, 0)$, $(0, 3/2)$, $(3/2, 0)$ and $(0, -3/2)$. We start by defining the inequality constraints:

$$\begin{aligned}
g_1(x, y) &= x + y - \tfrac{3}{2} = \langle [1, 1], [x, y] \rangle - \tfrac{3}{2} \\
g_2(x, y) &= x - y - \tfrac{3}{2} = \langle [1, -1], [x, y] \rangle - \tfrac{3}{2} \\
g_3(x, y) &= -x - y - \tfrac{3}{2} = \langle [-1, -1], [x, y] \rangle - \tfrac{3}{2} \\
g_4(x, y) &= -x + y - \tfrac{3}{2} = \langle [-1, 1], [x, y] \rangle - \tfrac{3}{2}
\end{aligned}$$

We also need the gradient of f: $\nabla f(x, y) = \big[2(x-3), 2(y-2)\big]$. At any point (x_0, y_0), the linear approximation needed for this method is given by

$$\begin{aligned}
L_0(x, y) &= f(x_0, y_0) + \langle \nabla f(x_0, y_0), (x - x_0, y - y_0) \rangle \\
&= (x_0 - 3)^2 + (y_0 - 2)^2 + 2(x_0 - 3)(x - x_0) + 2(y_0 - 2)(y - y_0).
\end{aligned}$$

Let's assume that the initial guess is $(0, 0)$, and we are looking for an exact solution ($\varepsilon = 0$). At this point, we set up $LB = -\infty$, and $UB = f(0, 0) = 13$. According to our description of Frank-Wolfe, we have to solve at this step the program

$$(LP_0) : \begin{cases} \min\limits_{(x,y)\in S} \big(-6x - 4y\big) \\ S = \{(x, y) \in \mathbb{R}^2 : g_k(x, y) \le 0, 1 \le k \le 4\} \end{cases}$$

By performing the simplex method, we find the optimal solution of this program attained at the point $(\bar{x}_0, \bar{y}_0) = (3/2, 0)$, with $\xi_0 = L_0(3/2, 0) = -9$. We proceed to perform a line-search on the segment joining $(0, 0)$ and $(3/2, 0)$:

$$t_0 = \operatorname*{argmin}_{0\le t\le 1} f\big(\tfrac{3}{2}t, 0\big) = \operatorname*{argmin}_{0\le t\le 1} \big[4 + \big(\tfrac{3}{2}t - 3\big)^2\big] = 1.$$

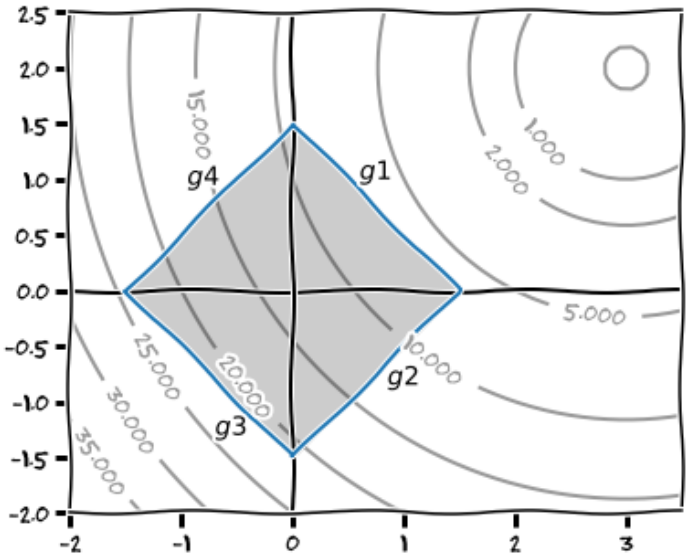


FIGURE 5.2. Set up for Example 5.9

It is then $(x_1, y_1) = (x_0, y_0) + t_0(\bar{x}_0 - x_0, \bar{y}_0 - y_0) = \left(\frac{3}{2}, 0\right)$, and $LB = \max(-\infty, \xi_0) = -9$.

At this point the stopping criteria gives $|UB - LB| = |13 - (-9)| = 22$. We proceed to the second iteration step, but we update the upper bound first: $UB = f(x_1, y_1) = f(3/2, 0) = 25/4$.

We need the approximation to f at $(3/2, 0)$:

$$\begin{aligned} L_1(x, y) &= f\left(\tfrac{3}{2}, 0\right) + \left\langle \nabla f\left(\tfrac{3}{2}, 0\right), \left[x - \tfrac{3}{2}, y\right] \right\rangle \\ &= \tfrac{25}{4} + \left\langle [-3, -4], \left[x - \tfrac{3}{2}, y\right] \right\rangle = \tfrac{25}{4} - 3\left(x - \tfrac{3}{2}\right) - 4y \end{aligned}$$

The corresponding associated program at this stage is as follows:

$$(LP_1) : \begin{cases} \min\limits_{(x,y)\in S} \left(-3x - 4y\right) \\ S = \{(x, y) \in \mathbb{R}^2 : g_k(x, y) \leq 0, 1 \leq k \leq 4\} \end{cases}$$

After applying the simplex method to this program, we find that the solution is the point $(\bar{x}_1, \bar{y}_1) = (0, 3/2)$, with $\xi_1 = L_1(0, 3/2) = 19/4$.

A line search between (x_1, y_1) and $(\bar{x}_1, \bar{y}_1)$ gives

$$t_1 = \operatorname*{argmin}_{0\leq t\leq 1} f\left(\tfrac{3}{2}(1 - t), \tfrac{3}{2}t\right) = \operatorname*{argmin}_{0\leq t\leq 1} \left(\tfrac{9}{2}t^2 - \tfrac{3}{2}t + \tfrac{25}{4}\right) = \tfrac{1}{6},$$

and therefore $(x_2, y_2) = (x_1, y_1) + t(\bar{x}_1 - x_1, \bar{y}_1 - y_1) = (5/4, 1/4)$. We also have $LB = \max(-9, \xi_1) = 19/4$.

At this point, the stopping criteria gives $|UB - LB| = |25/4 - 19/4| = \frac{3}{2}$. We proceed to the third step, but update the upper bound first: $UB = f(x_2, y_2) = f(5/4, 1/4) = 49/8$.

We need the approximation to f at $(5/4, 1/4)$:

$$L_2(x,y) = f\left(\tfrac{5}{4},\tfrac{1}{4}\right) + \left\langle \nabla f(\tfrac{5}{4},\tfrac{1}{4}), \left[x-\tfrac{5}{4}, y-\tfrac{1}{4}\right]\right\rangle = -\tfrac{7}{2}x - \tfrac{7}{2}y + \tfrac{91}{8}.$$

The corresponding associated program at this stage is as follows:

$$(LP_2): \begin{cases} \min\limits_{(x,y)\in S}\left(-\tfrac{7}{2}x - \tfrac{7}{2}y\right) \\ S = \{(x,y)\in\mathbb{R}^2 : g_k(x,y) \le 0, 1 \le k \le 4\} \end{cases}$$

The solution is again the point $(\bar{x}_2, \bar{y}_2) = (5/4, 1/4)$, with $\xi_2 = L_2(5/4, 1/4) = 49/8$. No further computations are needed at this point to realize that $(x_3, y_3) = (\bar{x}_2, \bar{y}_2) = (x_2, y_2) = (5/4, 1/4)$. Notice that $|UB - LB| = 0$, and the stopping criteria has been satisfied as expected.

The solution of the program (P) is precisely this point.

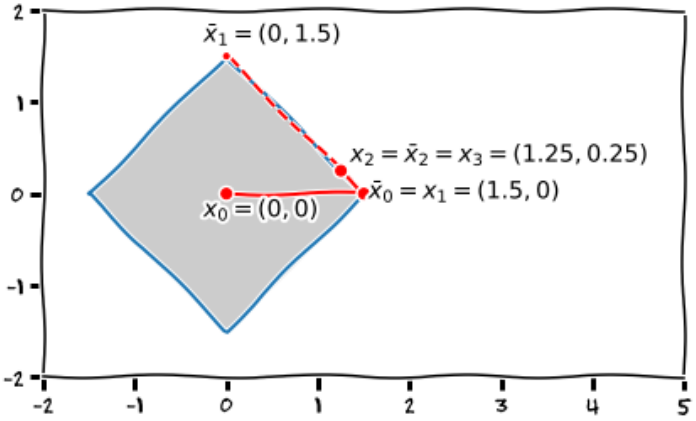


FIGURE 5.3. Frank-Wolfe iteration to solve the program (P) in Example 5.9.

Exercises

PROBLEM 5.1 (Basic). Solve the following linear program by the simplex method:

$$(LP): \begin{cases} \max\limits_{(x,y,z)\in\mathbb{R}^3} \left(4x + y - z\right) \\ x + 3z \le 6 \\ 3x + y + 3z \le 9 \\ x \ge 0, y \ge 0, z \ge 0 \end{cases}$$

Index

Arrow-Hurwicz-Uzawa constraint qualification, 71

BFGS method, *see* Broyden-Fletcher-Goldfarb-Shanno method, 53
Broyden method, 42
  iteration, 42
  recursive formula, 42
Broyden-Fletcher-Goldfarb-Shanno method, 53

Characteristic Polynomial, 16
Cholesky decomposition, 59
Conditional-Gradient method, *see* Frank-Wolfe method
Cone
  of improving directions, 64
  of inward pointing directions for the binding constraints, 64
Conjugate gradient, 38
Constrained optimization, 63
Constraint, 63
  equality, 63
  inequality, 63
Convergence
  cubic, 95
  linear, 95
  logarithmic, 95
  quadratic, 95
  rate of, 95
  sublinear, 95
  superlinear, 95
Convex
  function, 18, 26
  set, 17

Davidon-Fletcher-Powell method, 53
Derivative
  directional, 1, 12
  partial, 12

DFP method, *see* Davidon-Fletcher-Powell method
Direction, 1
  of steepest descent, 44, 75
Divided differences, 32

Eigenvalue, 16
Epigraph, 18, 26
Extreme Value, 6
Extremum, 6

Feasibility region, 63
Feasible
  point, 63
Frank-Wolfe method, 86
  lower bound, 86
  stopping criteria, 86
  upper bound, 86
Function
  coercive, 17, 21, 25
  continuous, 11, 24
  convex, 18, 22, 26
  differentiable, 12
  pseudo-convex, 21
  quasi-concave, 20
  quasi-convex, 20
  Rosenbrock, 2, 25, 37, 46
  strictly convex, 18, 22
  Weierstrass, 13

Gradient, 1
  descent, *see* Steepest descent

Hessian, 14
Horner's
  method, 56
  scheme, 56

image, 11
Indices of the binding inequality constraints, 64

Jacobian, 12

Karush-Kuhn-Tucker
  conditions, 67
  multipliers, 67
kernel, 11

Lagrange Multipliers, 5
Level set, 21
Line-search, 44, 77, 86
Linear map, 11, 70
Linear programming, 78
Linear transformation, *see* Linear map
LU-decomposition, 59

Matrix
  inverse, 36
  inversion, 36
  Jacobian, 12
  leading principal minor, 15
  principal minor, 15
  Symmetric, 14, 25
    Indefinite, 14, 25, 71
    Negative Definite, 14, 25, 71
    Negative Semidefinite, 14, 25, 71
    Positive Definite, 14, 25, 71
    Positive Semidefinite, 14, 25, 71
Maximum
  global, 6
  local, 7
  strict global, 7
  strict local, 7
Minimum
  global, 6, 65
  local, 7, 65
  strict global, 6, 65
  strict local, 7, 65

Newton method, *see* Newton-Raphson method
Newton-Raphson method, 31, 35, 38, 77
  direction, 38, 77
  error, 32
  iteration, 31, 35, 38
  Local convergence for, 33
  recursive formula, 31, 35, 38

Objective function of (P), 63
Optimization
  Constrained, 7
  Unconstrained, 7

Principal minor, 15
  leading, 15
Program, 63
  consistent, 63
  convex, 63
  Direction Finding, 75
  linear, 63
  super-consistent, 63

Quadratic Form, 14, 25

scipy, 38, 53, 83
Secant method, 40
  iteration, 40
  Local convergence for, 41
  recursive formula, 40
Secant property, 42
Set of tangent directions for equality constraints, 64
Simplex method, 78
  entering variable, 80
  slack variable, 79
  surplus variable, 79
  tableau, 79
  Unrestricted variables in sign, 79
Slater point, 63
Steepest descent, 44
  direction of, 44
  error, 50
  sequence of, 44

tableau, 79
Theorem
  Bounded Value, 21, 26
  Clairaut, 14
  Eigenvalue Criteria, 16
  Extreme Value, 21, 26
  Fritz John necessary conditions, 66
  Geometric necessary condition, 66
  Kantorovich estimate, 50, 61
  Karush-Kuhn-Tucker, 67, 68
  KKT necessary conditions, 67
  KKT sufficient conditions, 68
  Local Convergence for Newton-Raphson, 33
  Local Convergence for Secant, 41
  Orthogonal Gradient, 5
  Principal Minor Criteria, 15, 26
  Quadratic Convergence, 38
  Slater condition, 68
  Taylor's formula, 47
  Wolfe, 52

Vector
  unit, 1

Bibliography

[1] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[2] Francisco J Blanco-Silva. *Mastering SciPy*. Packt Publishing Ltd, 2015.

[3] Charles George Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

[4] G Dantzig. Linear programming and extensions, series rand corporation research study princeton univ, 1963.

[5] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

[6] Ross L Finney, Maurice D Weir, and George Brinton Thomas. *Thomas' calculus: early transcendentals*. Addison-Wesley, 2001.

[7] Roger Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970.

[8] Robert Freund. Nonlinear programming. Massachusetts Institute of Technology: MIT OpenCourseWare, 2004. `https://ocw.mit.edu` License: Creative Commons BY-NC-SA.

[9] Walter Gautschi. *Numerical analysis*. Springer Science & Business Media, 2011.

[10] Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.

[11] Godefroy Harold Hardy. Weierstrasss non-differentiable function. *Trans. Amer. Math. Soc*, 17(3):301–325, 1916.

[12] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[13] Anthony L Peressini, Francis E Sullivan, and J Jerry Uhl. *The mathematics of nonlinear programming*. Springer-Verlag New York, 1988.

[14] Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1964.

[15] David F Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

[16] David F Shanno and Paul C Kettler. Optimal conditioning of quasi-newton methods. *Mathematics of Computation*, 24(111):657–664, 1970.

APPENDIX A

Rates of Convergence

DEFINITION. Consider a convergent sequence $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}} \subset \mathbb{R}^d$ with $\boldsymbol{x}^\star = \lim_n \boldsymbol{x}_n$. We say that this sequence exhibits

Linear Convergence: If there exists $0 < \delta < 1$ so that

$$\lim_n \frac{\|\boldsymbol{x}_{n+1} - \boldsymbol{x}^\star\|}{\|\boldsymbol{x}_n - \boldsymbol{x}^\star\|} = \delta.$$

We refer to δ as the *rate of convergence*.

Superlinear Convergence: If

$$\lim_n \frac{\|\boldsymbol{x}_{n+1} - \boldsymbol{x}^\star\|}{\|\boldsymbol{x}_n - \boldsymbol{x}^\star\|} = 0.$$

Sublinear Convergence: If

$$\lim_n \frac{\|\boldsymbol{x}_{n+1} - \boldsymbol{x}^\star\|}{\|\boldsymbol{x}_n - \boldsymbol{x}^\star\|} = 1.$$

If, additionally,

$$\lim_n \frac{\|\boldsymbol{x}_{n+2} - \boldsymbol{x}_{n+1}\|}{\|\boldsymbol{x}_{n+1} - \boldsymbol{x}_n\|} = 1,$$

we say the the sequence exhibits *logarithmic convergence* to $\boldsymbol{x}^\star$.

Convergence of order $q > 1$: If $\boldsymbol{x}_n$ is exhibits superlinear convergence, and there exists $q > 1$, $0 < \delta < 1$ so that

$$\lim_n \frac{\|\boldsymbol{x}_{n+1} - \boldsymbol{x}^\star\|}{\|\boldsymbol{x}_n - \boldsymbol{x}^\star\|^q} = \delta.$$

In particular,

- Convergence with $q = 2$ is said to be *quadratic*.
- Convergence with $q = 2$ is said to be *cubic*.
- etc.

In any of these cases, we do refer to δ as the *rate of convergence*, as we did in the linear case.

A practical method to calculate the rate of convergence of a sequence is to calculate the following sequence, which converges to q:

$$q \approx \frac{\log\left|\frac{x_{n+1}-x_n}{x_n-x_{n-1}}\right|}{\log\left|\frac{x_n-x_{n-1}}{x_{n-1}-x_{n-2}}\right|} \tag{26}$$

EXAMPLE A.1. The sequence $x_n = 1/n!$ exhibits superlinear convergence, since $\lim_n \frac{1}{n!} = 0$ and

$$\lim_n \frac{x_{n+1}}{x_n} = \lim_n \frac{1}{n+1} = 0.$$

EXAMPLE A.2. Given $a \in \mathbb{R}$, $0 < r < 1$, the geometric sequence $\boldsymbol{x}_n = ar^n$ exhibits linear convergence, since $\lim_n ar^n = 0$ and

$$\lim_n \frac{x_{n+1}}{x_n} = r < 1.$$

The rate of convergence is precisely r.

EXAMPLE A.3. The sequence $x_n = 2^{-2^n}$ converges to zero and is superlinear:

$$\lim_n \frac{x_{n+1}}{x_n} = \lim_n 2^{-2^n} = 0$$

Using the estimation for q given by the formula in (26), we obtain that this sequence exhibits quadratic convergence. What is its rate of convergence?

EXAMPLE A.4. The sequence $x_n = 1/n$ converges to zero and is sublinear, since

$$\lim_n \frac{x_{n+1}}{x_n} = \lim_n \frac{n}{n+1} = 1.$$

Notice

$$\lim_n \frac{|\boldsymbol{x}_{n+2} - \boldsymbol{x}_{n+1}|}{|\boldsymbol{x}_{n+1} - \boldsymbol{x}_n|} = \lim_n \frac{n}{n+2} = 1;$$

therefore, this sequence exhibits logarithmic convergence.

APPENDIX B

Basic sympy commands for Calculus

A typical **sympy** session usually starts by loading the *symbols* we need, some basic functions, and basic constructors. After that, we proceed to the description of the functions we require.

```
# Symbols, including one for infinity, π and e
from sympy.abc import x,y,t,h
from sympy import oo, pi, E
# Symbols with conditions
from sympy import var
a,b = var('a,b', positive=True)

# Basic functions we may need
from sympy import sqrt, sin, cos, tan, exp, log

# Some basic symbolic manipulations may be needed
from sympy import solve, factor, expand, simplify, limit

# To do vector calculus, we need these two as well
from sympy import Matrix
from sympy.tensor.array import derive_by_array

# If in a jupyter notebook, we may want to render output as LaTeX
from sympy import init_printing
init_printing()

# Description of f
f = sin(x)/x

# A generic Rosenbrock function
# Note the symbols a, b act as parameters, while x and y act as variables
R = (a-x)**2 + b*(y-x**2)**2
```

We are going to use these functions to perform several common operations in Calculus.

1. Function operations

Observe how easily we can perform all of the following:

Function evaluation: with the method

`.subs({variable1: value1, variable2: value2, ...})`

Limits: with the function `limit(object, variable, value)`.

Basic operations: with the usual operators for addition, subtraction, multiplication and division.

Composition: again with the method `.subs()`.

```
>>> f.subs({x: pi}) # f(π)
        0
>>> f.subs({x: 0})  # f(0) --- returns "not a number"
        nan
>>> limit(f, x, 0)  # Compute lim_{x→0} f(x) instead
        1
>>> (f.subs({x: x+h}) - f)/h  # A divided quotient...
        (sin(h + x)/(h + x) - sin(x)/x)/h
>>> limit( (f.subs({x: x+h}) - f)/h, h, 0) # ... and its limit as h → 0
        (x*cos(x) - sin(x))/x**2
```

Notice how smart **sympy** is in regard to the properties of symbols

```
>>> sqrt(x**2) # Square root of the square of a variable without conditions
        sqrt(x**2)
>>> sqrt(a**2) # Square root of the square of a positive variable
        a
```

Directional limits are also possible

```
>>> limit(1/x, x, 0, dir="+")
        oo
>>> limit(1/x, x, 0, dir="-")
        -oo
```

2. Derivatives, Gradients, Hessians

For functions of one variable, to obtain the symbolic derivative of a function (of any order), we usually employ the method

`.diff(variable, order)`

For functions of several variables, we employ instead

`derive_by_array(function, list-of-variables)`

If necessary, we may arrange our outputs as matrices, so we can employ proper matrix operations with them.

```
>>> f.diff(x) # f'(x) without the need to mess with limits
        cos(x)/x - sin(x)/x**2
>>> f.diff(x, 2)  # f''(x)
        (-sin(x) - 2*cos(x)/x + 2*sin(x)/x**2)/x
>>> derive_by_array(R, [x,y]) # The gradient of R, ∇R
        [-2*a - 4*b*x*(-x**2 + y) + 2*x, b*(-2*x**2 + 2*y)]
>>> gradient = _   # Store that in the variable 'gradient'
>>> derive_by_array(gradient, [x,y]) # The Hessian of R, HessR
        [[8*b*x**2 - 4*b*(-x**2 + y) + 2, -4*b*x], [-4*b*x, 2*b]]
>>> hessian = Matrix(2,2, _) # Store that as a matrix, call it 'hessian'
>>> hessian[0,0] # If we want to access the first entry of the matrix
        8*b*x**2 - 4*b*(-x**2 + y) + 2
>>> simplify(_) # Simplify that expression
```

```
        12*b*x**2 - 4*b*y + 2
>>> Delta1 = _  # Store that value as 'Delta1'
>>> hessian.det()  # Compute the determinant of the Hessian
        -16*b**2*x**2 + 2*b*(8*b*x**2 - 4*b*(-x**2 + y) + 2)
>>> Delta2 = simplify(_) # Store that value as 'Delta2'
```

It is then a simple task (in some cases) to search for critical points by solving symbolically $\nabla f = 0$, and checking whether they are local maxima, local minima or saddle points.

```
>>> solve(gradient, [x,y]) # Critical points of R
        [(a, a**2)]
>>> crit_points = _ # This is a list.  We call it 'crit_points'
>>> for point in crit_points:
...     x0,y0 = point
...     print(point)
...     print("Delta1 = ", Delta1.subs({x:x0, y:y0}))
...     print("Delta2 = ", Delta2.subs({x:x0, y:y0}))
...
        (a, a**2)
        Delta1 =  8*a**2*b + 2
        Delta2 =  4*b
>>> 8*a**2*b + 2 > 0  # Is Delta1 > 0? (remember a,b>0)
        True
>>> 4*b > 0 # Is Delta2 > 0?
        True
```

The conclusion after this small session is that any Rosenbrock function $R(x, y) = (a - x)^2 + b(y - x^2)^2$ has a global minimum at the point (a, a^2).

A word of warning. Symbolic differentiation and manipulation of expressions may not work in certain cases. For those, numerical approximation is more suited (and incidentally, *that* is the reason you are taking this course).

```
>>> solve(f.diff(x))
NotImplementedError: multiple generators [x, tan(x/2)]
No algorithms are implemented to solve equation
x**2*(-tan(x/2)**2 + 1)/(tan(x/2)**2 + 1) - 2*x*tan(x/2)/(tan(x/2)**2 + 1)
```

3. Integration

Symbolic integration for the computation of antiderivatives is also possible. Definite integrals, while the symbolic setting allows it in many cases, it is preferably done in a numerical setting.

```
>>> R.integrate(x) # ∫ R(x, y) dx
        -a*x**2 + b*x**5/5 + x**3*(-2*b*y/3 + 1/3) + x*(a**2 + b*y**2)
>>> R.integrate(y) # ∫ R(x, y) dy
        -b*x**2*y**2 + b*y**3/3 + y*(a**2 - 2*a*x + b*x**4 + x**2)
>>> R.integrate(x, (x, 0, 1)).integrate(y, (y, 0, 1)) # ∫_0^1 ∫_0^1 R(x, y) dx dy
        a**2/4 - a/6 + 11*b/360 + 1/24
>>> f.integrate(x) # ∫ sin(x)/x dx
        Si(x)
```

```
>>> f.integrate(x, (x, 0, pi)) # ∫_0^π sin(x)/x dx
        -2 + pi*Si(pi)
>>> _.evalf() # How much is that, actually?
        3.81803183741885
```

4. Sequences, series

```
>>>
```

5. Power series, series expansions

```
>>>
```

APPENDIX C

Basic graphing in Python

One of the advantages of doing scientific computing with Python is the wealth of different libraries we may use to represent data and functions graphically. In this appendix we are going to do a quick overview of some of the most widely used:

(a) `matplotlib`
(b) `bokeh`
(c) `plotly`

1. `matplotlib`

The `matplotlib` libraries are fundamentally focused on 2D plotting. They are open source with license based on the *Python Software Foundation* (PSF) license. If you are planning to use them for your scientific production, it is customary to cite John Hunter's 2007 seminal paper [**12**].

In this section we are going to explore just a handful of utilities:

- The module `pyplot`, that allows us to use a similar syntax and interface as in `matlab` or `octave`
- The toolkit `mplot3d` to extend `matplotlib` for simple 3D plotting.
- The toolkits `basemap` and `cartopy` to extend `matplotlib` for projection and Geographic mapping.
- The toolkit `ggplot` for those of you familiar with the `R` plotting system.

Let's start with a simple example or usage of the module `pyplot`. We are going to use exclusively the Rosenbrock function $\mathcal{R}_{1,1}$.

```
1 import numpy as np, matplotlib.pyplot as plt
2
3 def R(x,y): return (1.0-x)**2 + (y - x**2)**2
```

For each plot, we usually indicate in our session the intent to create a *figure*. At that point it is customary to impose the size of the figure, number of subplots (if more than one), kind of axis, usage of grid, etc. This is what we call the *layout* of our diagrams. For instance, to create a simple plot (with size 5×10.5) of the graph of $f(x) = \mathcal{R}(x, 1)$ for $-2 \leq x \leq 2$, but focusing only in the window $[-2.5, 2.5] \times [-0.5, 10]$, we issue the following commands:

```
>>> x = np.linspace(-2, 2)          # −2 ≤ x ≤ 2
```

```
>>> plt.figure(figsize=(5,10.5));  # Create a figure of requested size
... plt.axes(aspect='equal');      # I want x's and y's to be the same size
... plt.grid();                    # I want a grid
... plt.xlim(-2.5, 2.5);           # Set my window
... plt.ylim(-0.5, 10);
... plt.plot(x, R(x,1.0));         # Just plot it...
... plt.show()                     # ...and request it shows in screen
```

See Figure C.1, left.

It is possible to combine several plots on the same figure:

```
>>> plt.figure(figsize=(5,10.5));
... plt.axes(aspect='equal');
... plt.grid();
... plt.xlim(-2.5, 2.5);
... plt.ylim(-0.5, 10);
... for section in range(1,6):     # Do 5 plots: R(x,n)
...     plt.plot(x,R(x,section))   # where n=1,2,3,4,5
...
... plt.show()
```

See Figure C.1, center.

It is hard to see which graph corresponds to which function, in spite of the different chosen colors. We solve this issue this by allowing each graph to be labeled, and placing a legend in the diagram:

```
>>> plt.figure(figsize=(5,10.5));
... plt.axes(aspect='equal');
... plt.grid();
... plt.xlim(-2.5, 2.5);
... plt.ylim(-0.5, 10);
... for section in range(1,6):     # The labels go here
...     plt.plot(x,R(x,section), label=section)
...
... plt.legend()                   # The legend is requested here
... plt.show()
```

See Figure C.1, right.

It is possible to control more detail of your plots. See for example how we modify line width, type, color, etc. As an exercise, try to comment each line indicating what modifications have been performed on the corresponding graphs.

```
>>> plt.figure();                 # Let pyplot choose size and layout
... plt.plot(x, R(x,1), 'r-', lw=0.5);
... plt.plot(x, R(x,2), 'b--',lw=2);
... plt.plot(x, R(x,3), 'go');
... plt.plot(x, R(x,4), 'y-.');
... plt.show()
```

To plot a set of level lines (a countour plot), we can either do that manually, or request `pyplot` to do it for us. For instance to render the level

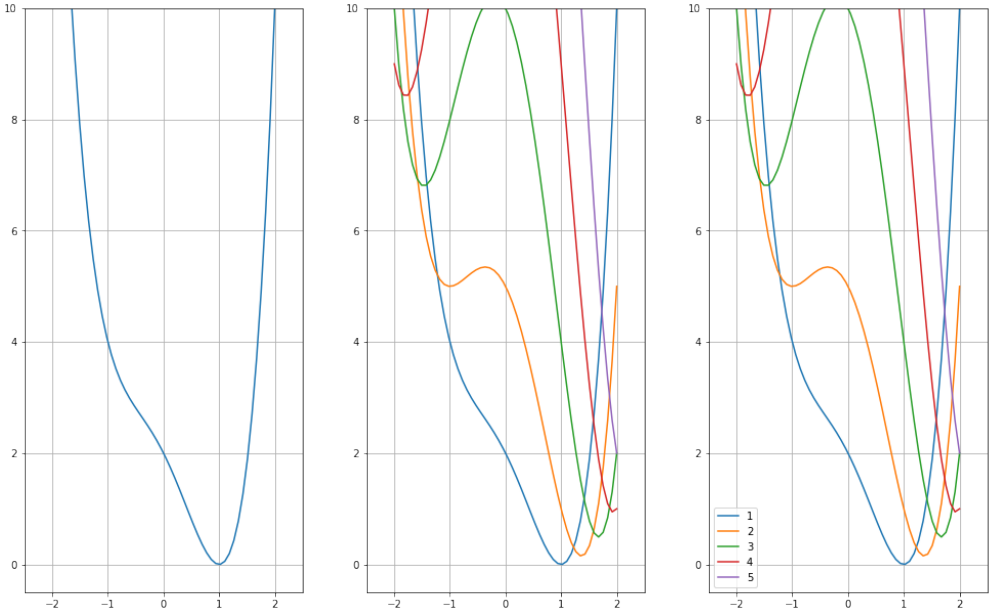


FIGURE C.1. Basic rendering of functions with `pyplot`

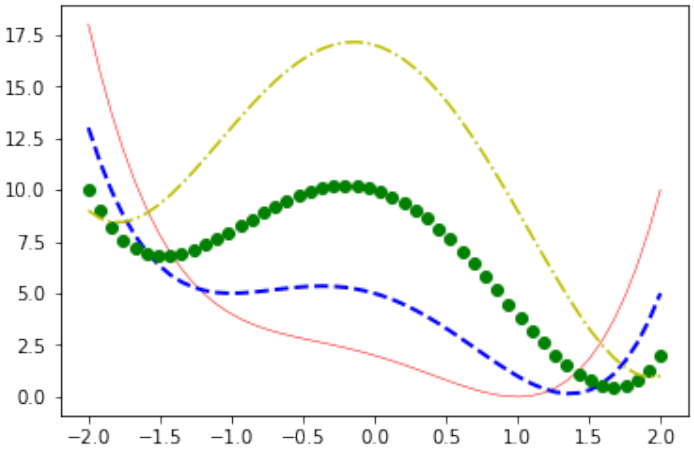


FIGURE C.2. Tinkering with color, style and width of lines in `pyplot`

lines $\mathcal{R}_{1,1}(x, y) = c$ for $c = 1, 2, 3, 5, 10, 15, 20$ over the window $[-2, 2] \times [-2, 3]$, we could issue the following commands:

```
>>> y = np.linspace(-2,3)          # −2 ≤ y ≤ 3
>>> X,Y = np.meshgrid(x,y)         # generate the window

>>> plt.figure();                  # Let pyplot choose size and layout
```

```
... plt.contour(X, Y, R(X,Y), levels=[1,2,3,5,10,15,20]);
... plt.show()
```

See Figure C.3, left.

Although each level line has been rendered with a different color, it is hard to see which one is which. Placing a legend on an already cluttered diagram may not be the best option. Fortunately, there is a neat method to display relevant information of top of each level line:

```
>>> plt.figure();                    # Let pyplot choose size and layout
... CS = plt.contour(X, Y, R(X,Y), levels=[1,2,3,5,10,15,20]);
... plt.clabel(CS, fontsize=9, inline=1);
... plt.show()
```

See Figure C.3, right.

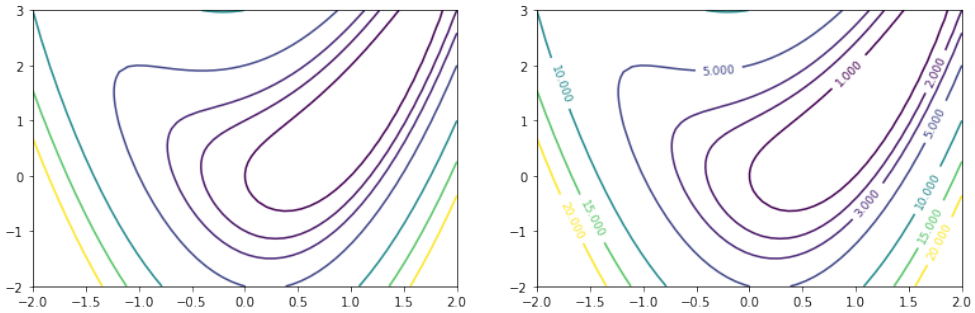


FIGURE C.3. Contour plots with `pyplot`