Chapter 2
Deltas and Epsilons

I mean the word proof not in the sense of the lawyers, who set two half proofs equal to a whole one, but in the sense of a mathematician, where half proof = 0, and it is demanded for proof that every doubt becomes impossible.
Carl Friedrich Gauss (1777–1855)

The introduction ended with recalling concepts in discrete mathematics as used in this book. This second chapter adds further basic concepts in continuous mathematics that are also relevant for this book, especially in the context of approximate algorithms.

2.1 Exact and δ-Approximate Algorithms

The term *exact algorithm* seems to be clear to everybody: the algorithm computes exactly the true solution for each input. However, the true solution is not always uniquely defined. For example, algorithms for mapping a 2D or 3D binary picture into topologically equivalent *skeletal curves* (see Fig. 2.1) are often based on iterative thinning methods (i.e., object pixels in 2D, or object voxels in 3D pictures are changed into background elements in one iteration if this operation does not change the topology of the picture). Researchers have published different proposals for *thinning* strategies. Correctly implemented algorithms deliver satisfactory results with respect to topology preservation or other predefined rules. But resulting skeletal curves are different for equal inputs and different algorithms. There is no unique "true solution" for thinning algorithms.

We do have a criterion of truth for ESPs. The length of a shortest path is always uniquely defined.

Definition 2.1 An algorithm is *exact* if it delivers a true solution for each input with respect to an existing criterion that is independent from the algorithm itself.

F. Li, R. Klette, *Euclidean Shortest Paths*,
DOI 10.1007/978-1-4471-2256-2_2,

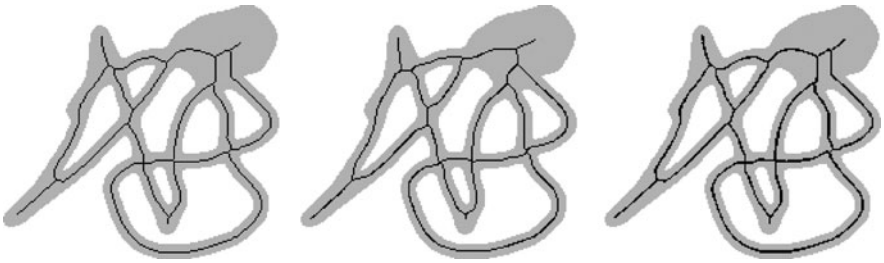

Fig. 2.1 Results of three different iterative thinning algorithms (*dark pixels*) on the same binary picture shown in *grey*. The top part of the *second* result differs significantly from that of the *first* and *third* results. (Courtesy of Gisela Klette, Auckland)

This definition requires exact correctness. Different measures specify approximate algorithms to express "how close" a solution needs to be to a true solution. This book is about minimisation problems. We say in short "a solution" for "the length of the calculated path" for the class of ESP problems.

Definition 2.2 An algorithm is a δ-*approximate algorithm* for a given minimisation problem iff, for each input instance of this problem, the algorithm delivers a solution that is at most δ times the optimum solution.

Obviously, $\delta < 1$ is impossible, and $\delta = 1$ defines an exact algorithm. In general, we may assume that $\delta = 1 + \varepsilon_0$, for some $\varepsilon_0 \geq 0$. For example, $\delta = 1.15$ defines $\varepsilon_0 = 0.15$, and thus an error limit of 15%.

Definition 2.3 An algorithm is *without guarantee* (of exactitude) iff there exists a real number $\varepsilon > 0$ and an input instance such that the difference between the output for this input and the true solution is larger than ε, for any choice of the algorithm's free parameters.

Note that a δ-approximate algorithm might even be an algorithm without guarantee, for $\varepsilon < (\delta - 1)$.

A δ-approximate algorithm does not necessarily calculate a solution arbitrarily close to an optimal solution.

Definition 2.4 An algorithm is *within guaranteed error limits* iff for any $\varepsilon > 0$ and each input instance, there is a choice of the free parameters for this algorithm such that the difference between output for this input and true solution is smaller than or equals ε.

Corollary 2.1 *Any algorithm within guaranteed error limits is also a* δ*-approximation algorithm, for any* $\varepsilon_0 > 0$ *and* $\delta = 1 + \varepsilon_0$.

Algorithm 2 (Papadimitriou algorithm, 1985)
Input: A finite set of simple polyhedra (the obstacles) with n vertices and $\mathcal{O}(n)$ edges, a start point p that is not on the surface of any polyhedron, and an end point q, all in 3D Euclidean space. Let E_Π be the set of all edges of the polyhedral obstacles, and $\varepsilon > 0$ the accuracy constant.
Output: A path (a polyline) from p to q.

1: **for** each edge $e \in E_\Pi$ **do**
2: Compute a point r in e such that $d_2(p, r) = \min\{d_2(p, r') : r' \in e\}$.
3: Break e into shorter line segments with endpoints

$$(x_r + d_i, y_r + d_i, z_r + d_i) \quad \text{and} \quad (x_r - d_i, y_r - d_i, z_r - d_i)$$

where

$$d_i = d_e(p, r) \times \frac{\varepsilon}{4\sqrt{3}n}\left(1 + \frac{\varepsilon}{4n}\right)^{i-1}$$

for $i = 1, 2, \ldots$.
4: Let V be the set of midpoints of such shorter line segments. [A line segment is called the *associated* segment of its midpoint.]
5: Put p and q also into V.
6: **end for**
7: Construct a *visibility-weighted undirected graph* $G = [V, E, w]$ as follows: for any pair of points u and v in V, there is an edge $\{u, v\} \in E$ iff their associated segments are visible from one-another. In this case, define the weight of $\{u, v\}$ as $d_2(u, v)$. Otherwise, let $+\infty$ be the weight of edge $\{u, v\}$.
8: Apply the Dijkstra algorithm for computing a shortest path from p to q in graph G

Fig. 2.2 Papadimitriou's algorithm for solving the general 3D ESP problem. The calculated shortest path in graph G is an approximate ESP for the given problem

Example 2.1 Consider a *general 3D ESP problem* where p and q are points in 3D space, and a shortest path from p to q cannot pass through a finite set of simple polyhedral obstacles having n vertices in total, and $\mathcal{O}(n)$ edges. The start point p is not on the surface of any of those polyhedra. The *Papadimitriou algorithm* is a δ-approximate algorithm for solving this problem; see Fig. 2.2 for a pseudocode of this algorithm.

This algorithm maps the given continuous ESP problem into a discrete problem by subdividing the edges involved into a finite number of segments, where the scale of the subdivisions is defined by increments d_i, depending on the selected ε. The particular definition of increments d_i is chosen for numerical considerations in the original paper that are outside the scope for our discussion here. The visibility between edges was introduced when commenting on Fig. 1.12.

The Papadimitriou algorithm is an algorithm within guaranteed error limits. Let $\delta = 1 + \varepsilon_0$. The algorithm produces a path guaranteed to be not longer than $(1 + \varepsilon_0)$

Algorithm 3 (Control structure of an iterative ESP algorithm)
Input: A search domain, obstacles, attractions, a start point p and an end point q, all in 2D or 3D Euclidean space.
Output: A path (a polyline) from p to q.

1: INITIALISATION: path ρ_0 from p to q, and L_0 be the length of ρ_0; $i = 1$.
2: **while** STOP CRITERION = false **do**
3: UPDATE produces a new path ρ_i.
4: Let L_i be the length of ρ_i.
5: Let $i = i + 1$.
6: **end while**

Fig. 2.3 Defining control structure of an iterative ESP algorithm

times the length of a true ESP. Without proof we just state that it has the time complexity

$$\mathcal{O}\left(\frac{n^4[b + \log(n/\varepsilon_0)]^2}{\varepsilon_0^2}\right) \tag{2.1}$$

where b is the number of bits representing the coordinates of the vertices in the polyhedral search domain Π (i.e., b is the base-2 logarithm of the largest integer appearing as a coordinate for one of the vertices of Π), and n is the total number of edges of Π. □

The Papadimitriou algorithm within guaranteed error limits is able to come arbitrarily close to an optimum solution, at the cost of an increase in time complexity of the algorithm: for small ε_0, the numerator in Eq. (2.1) is characterised by $n^4 \cdot b$ and the denominator by a very small number ε_0^2. Note that the given time complexity $f(n, \varepsilon_0)$ in Eq. (2.1) cannot be split into a product of two functions $f_1(n)$ and $f_2(\varepsilon)$, such that f_1 is independent of ε, and f_2 independent of n.

2.2 Approximate Iterative ESP Algorithms

Numerics makes frequent use of the concept of repeated iterations for defining algorithms within guaranteed error limits. An *iterative ESP algorithm* is defined by performing at first a constant number of operations before running into a loop; the loop performs a finite number of iterations, each time ending with the specification of an ESP of length L_i, where i runs from 0 to the maximum number of iterations; see Fig. 2.3.

This is a general control structure. Any iterative ESP algorithm needs to be specified by explaining the initialisation of path ρ_0, defining the stop criterion, and the update (i.e., how to obtain path ρ_{i+1} from path ρ_i and some optimisation strategy).

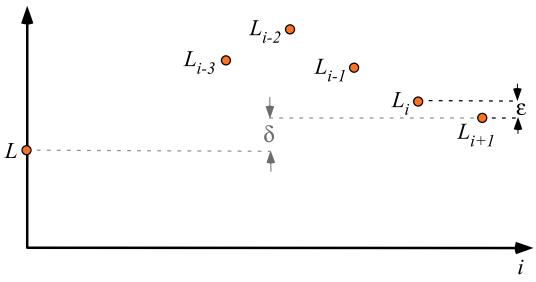

Fig. 2.4 A sketch of a distribution of L_i-values, in relation to the exact length L. The distance between L_i and L_{i+1} is shown to be less than ε, but the distance δ between L_{i+1} and L remains unknown

The stop criterion should ensure that the algorithm is not running into an infinite loop. The time complexity of an iterative ESP algorithm is then simply given as follows:

$$f(\mathbf{x}) = f_{\text{initialisation}}(\mathbf{x}) + f_{\text{stop}}(\mathbf{x}) \sum_{i=1}^{i_{\max}(\mathbf{x})} f_{\text{update}}(\mathbf{x}, i)$$

where $\mathbf{x}$ is a vector combining all the algorithm's parameters (i.e., the problem complexity $n > 0$ and possibly some free parameters). This further simplifies if $f_{\text{update}}(\mathbf{x}, i)$ is basically independent of the iteration number i, and $f_{\text{stop}}(\mathbf{x})$ assumed to be a constant $c > 0$:

$$f(\mathbf{x}) = f_{\text{initialisation}}(\mathbf{x}) + c \cdot i_{\max}(\mathbf{x}) \cdot f_{\text{update}}(\mathbf{x}).$$

Because we are interested in asymptotic complexities, we can also ignore the constant c (see Definition 1.4) and obtain

$$f(\mathbf{x}) = f_{\text{initialisation}}(\mathbf{x}) + i_{\max}(\mathbf{x}) \cdot f_{\text{update}}(\mathbf{x}). \tag{2.2}$$

The update should guarantee that the new path is a better solution (i.e., of reduced length) compared to the previous path. If implementing both ideas properly, then we may stop the algorithm by using an *accuracy parameter* $\varepsilon > 0$:

Definition 2.5 An iterative ESP algorithm is *approximate* iff, for any $\varepsilon > 0$, there exists a natural number i_ε such that if $i \geq i_\varepsilon$ then

$$|L_{i+1} - L_i| < \varepsilon. \tag{2.3}$$

Let L be the (exact) length of an ESP. Definition 2.5 does not compare L_i with L, but L_i with L_{i+1}. This is illustrated in Fig. 2.4. The figure also shows a case where L_{i-2} is larger than L_{i-3}. This should actually be avoided by a proper specification of the algorithm.

Because we do not know L, we can only compare previously calculated L_i-values. Definition 2.5 is thus different to that of an algorithm within guaranteed error limits. The figure also illustrates a distance ε between L_i and L_{i+1}, and a distance δ between L_{i+1} and L, indicating that δ could possibly still be larger than ε.

Instead of Eq. (2.3), the unscaled *stop criterion* in Definition 2.5, consider a scaled stop criterion

$$\frac{|L_{i+1} - L_i|}{L_{i+1}} < \varepsilon. \tag{2.4}$$

This is a common stop criterion, for example, in robotics, where values L_i are measurements (e.g., by some sensors). In such a case, the reason for applying Eq. (2.4) is that, if the error in measuring L_i is of the order of machine accuracy and the error in measuring L_{i+1} is also of the order of machine accuracy, then the resulting error in unscaled $|L_{i+1} - L_i|$ can even be of the order twice of that of machine accuracy. However, we stay with Eq. (2.3) for quantities L_i calculated in our algorithms.

The following section shows that any approximate ESP algorithm is also an algorithm within guaranteed error limits, and thus also an $(1 + \varepsilon_0)$-approximate algorithm, for any $\varepsilon_0 > 0$.

2.3 Convergence Criteria

Let f be a function that maps values from some subset $S \subseteq \mathbb{R}$ into $\mathbb{R}$, and let $c \in \mathbb{R} \cup \{-\infty, +\infty\}$ be a constant. The notation $x \to c$ is short for 'value x goes arbitrarily close to c': (i) for $c \in \mathbb{R}$ and any $\varepsilon > 0$ there is a value $x \in S$ with $|x - c| < \varepsilon$; (ii) for $c = -\infty$ and any $T < 0$ there is a value $x \in S$ with $x < T$; (iii) for $c = +\infty$ and any $T > 0$ there is a value $x \in S$ with $x > T$. Let

$$\lim_{x \to c} f(x)$$

be defined and equal to a real number a iff $f(x)$ can go arbitrarily close to a as $x \to c$, which means that for any $\delta > 0$ there is

(case $c \in \mathbb{R}$) an $\varepsilon > 0$ such that for any $x \in S$ with $|x - c| < \varepsilon$ it follows that $|f(x) - a| < \delta$;
(case $c = -\infty$) a $T_0 < 0$ such that for any $x \in S$ with $x < T_0$ it follows that $|f(x) - a| < \delta$;
(case $c = +\infty$) a $T_0 > 0$ such that for any $x \in S$ with $x > T_0$ it follows that $|f(x) - a| < \delta$.

Definition 2.6 Function f *converges* (or *is convergent*) as $x \to c$ iff $\lim_{x \to c} f(x)$ is defined. If it is defined and equals a then a is called the *limit* of f as $x \to c$.

For example, f can be defined on the set of natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$, defining a sequence $a_i = f(i)$ of real numbers, for $i \geq 0$. This sequence $\{a_0, a_1, a_2, \ldots\}$ may, for example, 'oscillate somehow' around a value $a \in \mathbb{R}$ by coming closer and closer, say, but for index $i + 1$ a bit more than for index i:

$$|a - a_{i+1}| < |a - a_i|$$

for all $i = 0, 1, 2, \ldots$; see Fig. 2.5. The infinite sequence $\{a_0, a_1, a_2, \ldots\}$ is converging to the limit a. In this case, we have that constant c equals $+\infty$.

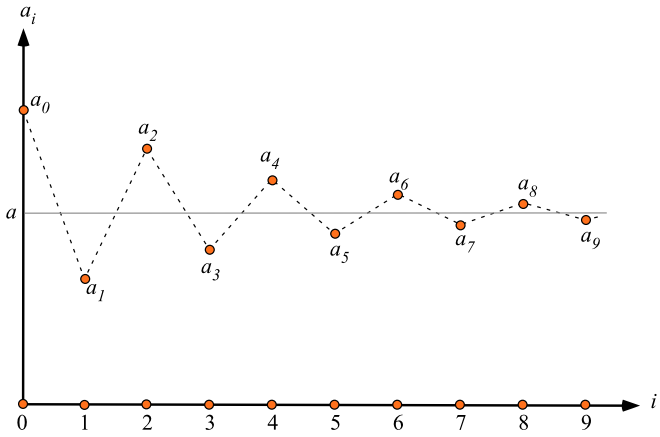

Fig. 2.5 A sequence of reals with constantly decreasing distances from the real number a

Example 2.2 Let b be any real; the sequence $a_{i+1} = (a_i + b/a_i)/2$ of rational numbers, for any given initial rational number a_0, is converging[1] to $a = \sqrt{b}$. Let $b = 2$ and $a_0 = 1$, for example, then the first elements are

$$\begin{aligned} a_0 &= 1, \\ a_1 &= 1.5, \\ a_2 &= 1.41666666666\ldots, \\ a_3 &= 1.41421568627\ldots, \\ a_4 &= 1.41421356237\ldots, \end{aligned}$$

and the sequence converges to the limit $\sqrt{2}$. □

Now let us 'remove' the real a from the definition of convergence, and we just claim for a given sequence $\{a_0, a_1, a_2, \ldots\}$ of real numbers and for all $i = 0, 1, 2, \ldots$ that

$$|a_{i+1} - a_{i+2}| < |a_i - a_{i+1}|.$$

This is a first step towards a *Cauchy sequence*.[2] (In general, a Cauchy sequence can also be defined in terms of a function f defined on $S \subseteq \mathbb{R}$; however, we only need this special case of $S = \mathbb{N}$ in this book.)

Definition 2.7 A *Cauchy sequence* is an infinite sequence $\{a_0, a_1, a_2, \ldots\}$ of real numbers, such that for any $\varepsilon > 0$ there is an index i_0 with $|a_i - a_j| < \varepsilon$, for any $i, j \geq i_0$.

[1] *Heron of Alexandria* (ca. 10–70) described this approximation method, which is also known as *Babylonian method*.

[2] Named after *Baron Augustin-Louis Cauchy* (1789–1857), who was central for establishing the infinitesimal calculus (e.g., of convergence of real numbers).

This definition covers the case of some kind of 'irregular' finite beginning of the sequence where values may not follow any fixed rule. But, after some finite beginning, the elements of the sequence have to come closer to each other, for any increase of the index. With Definition 2.5 we have immediately:

Corollary 2.2 *The sequence L_i of lengths, for $i \geq 0$, calculated by an approximate iterative ESP algorithms, defines a Cauchy sequence.*

Theorem 2.1 (Cauchy Convergence Criterion) *A sequence of real numbers is convergent iff it is a Cauchy sequence.*

A proof of this theorem may be based upon the inequalities

$$|a| - |b| \leq |a - b| \leq |a| + |b|$$

for any reals a and b, and on the *Bolzano–Weierstrass Theorem*, saying that every bounded sequence has a convergent subsequence.[3] A sequence $\{a_0, a_1, a_2, \ldots\}$ of real numbers is *bounded* iff there are real numbers a and b such that $a \leq a_i \leq b$, for all $i \geq 0$.

Let $a_1, a_2, a_3, \ldots$ be a convergent sequence. Thus, for any $\varepsilon > 0$ there is a k_ε such that $|a_k - a| < \varepsilon$, for all $k \geq k_\varepsilon$. Now consider $\varepsilon/2$. Let $i, j > k_{\varepsilon/2}$, with $|a_i - a| < \varepsilon/2$ and $|a_j - a| < \varepsilon/2$. Applying the above second inequality, we obtain that

$$|a_i - a_j| = \big|(a_i - a) - (a_j - a)\big| \leq |a_i - a| + |a_j - a| \leq \varepsilon/2 + \varepsilon/2 = \varepsilon.$$

This shows that the sequence is Cauchy.

The other direction of the proof is not so short, and we just mention that every Cauchy sequence is bounded, and possesses thus a convergent subsequence (see the Bolzano–Weierstrass Theorem above). The proof can then be completed by showing that, if a Cauchy sequence has a subsequence that is convergent to a, then the whole sequence is also convergent to a.

We state the Cauchy Convergence Criterion above, and the following basic results of infinitesimal calculus without (complete) proof, but for possible reference later in the book. Theorem 2.1 and Corollary 2.2 show that any approximate iterative ESP algorithm is also an algorithm with guaranteed error limits.

Definition 2.8 A sequence $\{a_0, a_1, a_2, \ldots\}$ of real numbers is *monotonically decreasing* (*monotonically increasing*) iff $a_i \geq a_{i+1}$ ($a_i \leq a_{i+1}$), for all $i \geq 0$. Such a sequence is called *monotone* iff it is either monotonically de- or increasing.

Theorem 2.2 (Monotone Convergence Criterion) *A monotone sequence of real numbers is convergent iff it is bounded.*

Corollary 2.3 *A monotonically decreasing sequence of real numbers is convergent iff it is lower bounded.*

[3] Named after *Bernard Placidus Johann Nepomuk Bolzano* (1781–1848) and *Karl Theodor Wilhelm Weierstrass* (1815–1897).

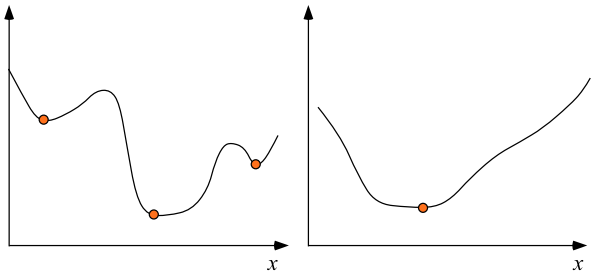

Fig. 2.6 *Left*: Graph of a non-convex function. *Right*: Graph of a convex function. The *points* shown are local minima

In this book, convergence proofs for iterative ESP algorithms are always done by showing that a sequence is monotonically decreasing and lower bounded.

Let L_i, for $i \geq 0$, be the sequence of lengths calculated by an iterative ESP algorithm. In particular, according to Corollary 2.3, we only have to show the existence of a lower bound if a sequence is monotonically decreasing; then the sequence is convergent, also a Cauchy sequence, and the algorithm is approximate, thus also an algorithm with guaranteed error limits.

There is always a uniquely specified minimum length L for any input instance of the considered ESP problem (i.e., a lower bound). However, Corollary 2.3 does not say that the sequence is converging towards this lower bound; it might be another real number larger than the lower bound.

2.4 Convex Functions

Measurements during an iterative solution define discrete samples L_i, for iterations $i \geq 0$. We may assume that a continuous function $f(x)$, defined on the interval $\mathbb{R}^+$ of all non-negative real numbers, is fitting these samples accurately: $f(i) = L_i$, for all $i \geq 0$.

Definition 2.9 A function f, defined on an interval of real numbers, is *convex* if

$$f\big(\lambda x_1 + (1-\lambda)x_2\big) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

for any two reals x_1, x_2 in this interval, and $0 \leq \lambda \leq 1$.

(In general, a convex function is defined over a convex subset of $\mathbb{R}^n$.)

A convex function possesses the important property that every local minimum is also a global minimum. This property is used in this book for showing whether solutions obtained by iterative ESP algorithms are always the global solutions. If f is a convex function then $-f$ is called a *concave function*. For example, the graph of $y = x^3$ is convex for $x \geq 0$ but concave for $x < 0$.

See Fig. 2.6 for examples of two unary functions $f(x)$; the graph of a convex function may also run parallel to the x-axis in one segment, because the relation in

Definition 2.9 also allows for equality. This would define a convex function that is *not strictly convex*. If we replace the relation in Definition 2.9 by

$$f\big(\lambda x_1 + (1-\lambda)x_2\big) < \lambda f(x_1) + (1-\lambda) f(x_2) \tag{2.5}$$

then f is *strictly convex*, and $-f$ is *strictly concave*.

For showing that there is a unique minimum of a function (defined by measurements during an iterative solution), we have to prove that this function is not only convex but even strictly convex.

2.5 Topology in Euclidean Spaces

For showing that there is a unique minimum of a function, we have to prove that this function is not only convex but even strictly convex; this is sometimes possible by applying results from topology in Euclidean spaces. We also need topology for defining "topological equivalence", or for describing accurately the "interior" of a set and its "frontier". (We already used those two notions in the context of polygons and polyhedra.)

We start our considerations in the 1-dimensional set $\mathbb{R}$ of reals. Let $a < b$ be two reals. The interval $[a, b] = \{x : a \le x \le b\}$ is *closed* (it also contains both its endpoints a and b), and the interval $(a, b) = \{x : a < x < b\}$ is *open*. We can also define "half-open" (or "half-closed") intervals such as $[a, b) = \{x : a \le x < b\}$. Points a and b define the *frontier* of the open (i.e., (a, b)) or closed (i.e., $[a, b]$) interval; the open interval (a, b) is also called the *interior* of $[a, b]$.

For a real c and $\varepsilon > 0$ we define with $N_\varepsilon(c) = \{x : |x - c| < \varepsilon\}$ the ε-*neighbourhood* of c, which is an open interval.

Definition 2.10 A set $S \subseteq \mathbb{R}$ is *open* if, for any point $c \in S$, there is an $\varepsilon > 0$ such that $N_\varepsilon(c) \subseteq S$.

Note that this is not true for the points a and b in the frontier of $[a, b]$. The property is true for any real in the open interval (a, b), and also for any real in an arbitrary (finite or infinite) union

$$(a_0, b_0) \cup (a_1, b_1) \cup (a_2, b_2) \cup \cdots$$

of open intervals, which do not have to be pairwise disjoint, or a finite intersection

$$(a_0, b_0) \cap (a_1, b_1) \cap (a_2, b_2) \cap \cdots \cap (a_m, b_m)$$

of open intervals.

If S is an open set, then $\mathbb{R} \setminus S$ is called a *closed set*. Note that closed intervals, as defined above, are also closed sets according to this definition. The set $\mathbb{R}$ is open, and the empty set $\emptyset$ is also open; thus, these two sets are also closed. These are the only two examples where a set is both closed and open. We summarise:

- Any ε-neighbourhood of a point $c \in \mathbb{R}$ is open.

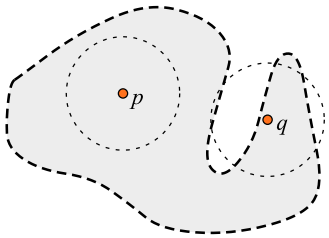

Fig. 2.7 The *dashed line* shows the frontier of an open set S. The shown neighbourhood of p is completely contained in S, but the shown neighbourhood for point q is not. For point q, a smaller radius of the neighbourhood needs to be chosen for having also a neighbourhood completely in S

- The union of any finite or infinite number of open sets is again open.
- The intersection of any finite number of open sets is again open.
- $\mathbb{R}$ and $\emptyset$ are both open and closed.

Now we generalise these notions of open or closed sets to the Euclidean plane or 3D space using the Euclidean metric $d_e = d_2$ for defining ε-neighbourhoods

$$N_\varepsilon(p) = \{q : d_e(p, q) < \varepsilon\}$$

of points $p \in \mathbb{R}^m$, for $m = 2$ or $m = 3$. These neighbourhoods form open disks in the plane (see Fig. 2.7) or open spheres in 3D space.

Definition 2.11 Let $m = 2$ or $m = 3$. A set $S \subseteq \mathbb{R}^m$ is *open* iff, for any point $p \in S$, there is an $\varepsilon > 0$ such that $N_\varepsilon(p) \subseteq S$.

Again, if $S \subseteq \mathbb{R}^m$ is an open set, then $\mathbb{R}^m \setminus S$ is called a *closed set*. It follows that

- Any ε-neighbourhood of a point $p \in \mathbb{R}^m$ is open.
- The union of any finite or infinite number of open sets is again open.
- The intersection of any finite number of open sets is again open.
- $\mathbb{R}^m$ and $\emptyset$ are both open and closed.

This defines a consistent approach for dealing with open or closed sets in $\mathbb{R}^m$, for $m = 1$ and also for $m = 2$ or $m = 3$, and this can actually be extended to any $m \geq 1$ for defining topologies in Euclidean spaces $[\mathbb{R}^m, d_e]$.

A set $S \subseteq \mathbb{R}^m$ is *bounded* if there is some point $p \in \mathbb{R}^m$ and a radius $r > 0$ such that S is completely contained in the r-neighbourhood of p: $S \subseteq N_r(p)$. A set is *compact* if it is bounded and closed. For example, simple polygons or simple polyhedra are all bounded sets.

We conclude this section with introducing some commonly used notation. Let $S \subseteq \mathbb{R}^m$. We define:

$$\begin{aligned}
&(\textit{interior}) \quad S^\circ = \{p \in S : \exists \varepsilon (\varepsilon > 0 \wedge N_\varepsilon(p) \subseteq S)\}, \\
&(\textit{frontier}) \quad \partial S = \{p \in \mathbb{R}^m \setminus S^\circ : \forall \varepsilon (\varepsilon > 0 \rightarrow N_\varepsilon(p) \cap S \neq \emptyset)\}, \\
&(\textit{closure}) \quad S^\bullet = S^\circ \cup \partial S.
\end{aligned}$$

For any $S \subseteq \mathbb{R}^m$ it follows that

- S is open iff $S = S^\circ$.
- S is closed iff $S = S^\bullet$.
- $S^\circ \cap \partial S = \emptyset$.
- If S is closed then $S \setminus \partial S$ is open.

The set ∂S defines the *frontier* of a set S. For example, in the case $m = 3$, a simple polyhedron is defined by a finite number of polygonal faces; the union of those faces is the surface of the polyhedron, and also the frontier of the polyhedron. In the case $m = 2$, a simple polygon is defined by a polygonal loop; this loop is the frontier of the polygon. In the case $m = 1$, the frontier of an interval (a, b), $(a, b]$, $[a, b)$, or $[a, b]$ is the set $\{a, b\}$.

Definition 2.12 S is called (*topologically*) *connected* iff it is not the union of two disjoint nonempty open subsets (or, equivalently, closed subsets) of S.

Maximum connected subsets of S are called *components* of S.

A path *visits* a set S iff this path and S do have a nonempty intersection. Because a path does have an orientation (i.e., from start to end, or by a defined loop), it also visits S at some point *for the first time*, if there is a nonempty intersection, and if S is topologically closed. If the path does not start in S, then the first visit of a closed set S will be in ∂S.

Definition 2.13 A family $\mathcal{F}$ of subsets of a set $S_0 \subseteq \mathbb{R}^m$ defines a *topological space* or a *topology* iff it satisfies the following axioms:

T1 $\{\emptyset, S_0\} \subseteq \mathcal{F}$.
T2 The union of any finite or infinite number of sets in $\mathcal{F}$ is again in $\mathcal{F}$.
T3 The intersection of any finite number of sets in $\mathcal{F}$ is again in $\mathcal{F}$.

A set $S \in \mathcal{F}$ is *open*, and $S_0 \setminus S$ is *closed*.

According to axiom T1, the base set S_0 is open in its own topology. However, S_0 does not need to be open in another topology. For example, a compact set S_0 in $\mathbb{R}^m$ is not open in the topology of $\mathbb{R}^m$, but may define its own topology (i.e., all the open subsets of S_0 only), and S_0 is then open in its own topology.

Definition 2.14 A topological space or topology $\mathcal{F}$, defined by all open subsets of a set $S_0 \subseteq \mathbb{R}^m$, is called the *topology on* S_0.

The important concept of "topological equivalence" will be specified in the following section, after defining continuous mappings between topological spaces.

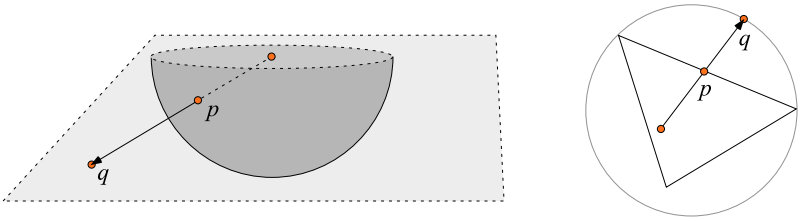

Fig. 2.8 *Left*: A perspective projection from the centre of the open (i.e., not containing its frontier) *half-sphere* onto a *plane tangent* to the *half-sphere* (which is parallel to the base of the hemisphere) defines a homeomorphism; point p maps onto point q. *Right*: Projection of a *triangle* onto a *circle* [R. Klette and A. Rosenfeld, 2004]

2.6 Continuous and Differentiable Functions; Length of a Curve

Consider a function f from a domain $D \subseteq \mathbb{R}$ into $\mathbb{R}$. Informally speaking, f is continuous on D iff, for $a_1, a_2 \in D$, values $f(a_1)$ and $f(a_2)$ are "close" to each other if a_1 and a_2 are close to each other. More formally:

> for any $\varepsilon > 0$ there is some $\delta > 0$ such that $|f(a_1) - f(a_2)| < \varepsilon$, for any $a_2 \in D$ with $|a_1 - a_2| < \delta$.

This says that the open δ-neighbourhood of a_1 is mapped into an open ε-neighbourhood of $f(a_1)$. An exact general definition of continuous functions is as follows:

Definition 2.15 A function f from a topology on S_0 into a topology on S_1 is called *continuous* iff, for any open set $S \subseteq S_1$, the set $f^{-1}(S) = \{a \in S_0 : f(a) \in S\}$ is also open in S_0.

In the previous section, we defined topologies on Euclidean spaces $\mathbb{R}^m$, for $1 \leq m \leq 3$. A continuous function maps one topology (or topological space) into another.

Definition 2.16 A function f from a topology on S_0 into a topology on S_1 is called a *homeomorphism* iff it is one-to-one, onto S_1, continuous, and f^{-1} is also continuous.

A homeomorphism allows us to map sets one-to-one "smoothly" from S_0 into S_1 and also "backward", from all S_1 into S_0.

Definition 2.17 Two sets S_0 and S_1 are *topologically equivalent* iff they can be mapped by a homeomorphism from S_0 onto S_1.

For example, the Euclidean plane $\mathbb{R}^2$ is topologically equivalent to an open half-sphere, and a triangle (i.e., just the polyline) is topologically equivalent to a circle; see Fig. 2.8. A circle with one point removed is topologically equivalent to $\mathbb{R}^1$. The base sets of these examples of topological spaces are all open with respect to

their topology, but not necessarily in the topology on the whole Euclidean space. Any compact (i.e., bounded and closed) subset of $\mathbb{R}^m$, $m \geq 1$, defines a topological space. Simple polygons or simple polyhedra are compact sets. The following is a basic theorem in combinatorial topology:

Theorem 2.3 *All simple polygons are topologically equivalent to a closed disk, and all simple polyhedra are topologically equivalent to a closed sphere.*

Derivatives Let f be a function from $\mathbb{R}$ into $\mathbb{R}$. The *derivative* $f'(x)$ is the slope of the tangent to the graph of a function f at $(x, f(x))$, defined by the limit

$$f'(x) = \frac{\mathrm{d}f(x)}{\mathrm{d}x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

of *Newton's difference quotient*[4] of function f at x. If this limit is defined for all x and $x+\varepsilon$ in $[a, b]$, then f is not only continuous but also *differentiable* at $x \in [a, b]$.

Now assume that f is a function in variables $x_1, x_2, \ldots,$ and x_m; the partial derivative of f with respect to variable x_i is denoted and defined by

$$\frac{\partial f(x_1, x_2, \ldots, x_m)}{\partial x_i} = \lim_{\varepsilon \to 0} \frac{f(x_1, \ldots, x_i + \varepsilon, \ldots, x_m) - f(x_1, \ldots, x_i, \ldots, x_m)}{\varepsilon}.$$

For example, the polynomial $p(\lambda_1, \lambda_2, \lambda_3) = 4\lambda_1^2\lambda_2 + \lambda_1\lambda_3 + 5\lambda_2^3\lambda_3$ has the partial derivative $8\lambda_1\lambda_2 + \lambda_3$ with respect to λ_1, $4\lambda_1^2 + 15\lambda_2^2\lambda_3$ with respect to λ_2, and $\lambda_1 + 5\lambda_2^3$ with respect to λ_3.

Length of a curve In 2D with rectangular Cartesian coordinates, consider a parameterised curve $\gamma(\lambda) = (x(\lambda), y(\lambda))$, for $a \leq \lambda \leq b$. Denote the derivatives of $x(\lambda)$ and $y(\lambda)$ by $\dot{x}$ and $\dot{y}$.

Definition 2.18 The *length* of the curve $\gamma(\lambda)$ in 2D is defined as

$$\mathcal{L}(\gamma) = \int_{x(a)}^{x(b)} \sqrt{1 + \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^2}\, \mathrm{d}x.$$

The integral over x gives signed length of any part of the curve for which y is a single-valued function of x. The following integral over λ gives positive length over a general parameterised curve:

$$\mathcal{L}(\gamma) = \int_a^b \sqrt{\dot{x}^2 + \dot{y}^2}\, \mathrm{d}\lambda.$$

In 3D, consider a parameterised curve $\gamma(\lambda) = (x(\lambda), y(\lambda), z(\lambda))$, for $a \leq \lambda \leq b$. Denote the derivative of $z(\lambda)$ by $\dot{z}$.

[4] Named after *Isaac Newton* (1642–1727 in the Julian calendar, which was then used in England).

Definition 2.19 The *length* of the curve $\gamma(\lambda)$ is defined by

$$\mathcal{L}(\gamma) = \int_a^b \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}\, \mathrm{d}\lambda.$$

In 2D or 3D, the integral over λ gives the same length of a curve for all rectangular Cartesian coordinate axes.

Example 2.3 Consider a line segment between points $p = (p_x, p_y, p_z)$ and $q = (q_x, q_y, q_z)$. A parameterised form of the segment is given by $\gamma(\lambda) = p + \lambda(q - p)$, for $0 \leq \lambda \leq 1$. We obtain that $x(\lambda) = p_x + \lambda(q_x - p_x)$, and $\dot{x} = q_x - p_x$, with similar coordinate differences for $\dot{y}$ and $\dot{z}$. It follows that the length $\mathcal{L}(\gamma)$ equals (as expected) the Euclidean distance between p and q. □

The example shows that the definition of the length of a path (see Definition 1.7) is consistent with the general definition of the length of a curve.

2.7 Calculating a Zero of a Continuous Function

Continuous functions are important in a numerical context. For example, sometimes we want to calculate a zero x of a continuous function f (i.e., $f(x) = 0$), defined on a domain $D \subset \mathbb{R}$, without having a general formula for calculating those zeros (compare Sect. 1.4; such a general formula may even not exist).

Assume a continuous function f that is defined in the interval $[a, b]$, with $a < b$, and that satisfies $f(a) f(b) < 0$. *Bolzano's Theorem* proves that f has at least one zero in $[a, b]$.

n-section method A straightforward method is to subdivide $[a, b]$ uniformly into $n > 0$ sections of equal length and to test f at the resulting endpoints of these sections; see Fig. 2.9.

Binary search is a general strategy for reducing time in a search routine: divide the search space recursively into halves and apply the search criterion at borders of the resulting subspaces.

Binary-search method We replace the subdivision into n equal sections by an iterated division into halves, testing the sign of the product of two f-values once (for the left endpoint of the current segment and its midpoint). We know that there is at least one zero of f somewhere in the current segment, and this product tells us whether we have to continue the search in the left half of the segment or in the right half. The search stops when the value at the midpoint is sufficiently close to zero.

Algorithm 4 (n-Section Method)
Input: Reals a and b, integer n, an accuracy constant $\varepsilon > 0$; we also have a way to calculate $f(x)$, for any $x \in [a, b]$.
Output: Value $c \in [a, b]$ such that $|f(c)| < \varepsilon$.

Set *flag* = *false* and $i = 0$.
while $i < n$ **do**
  if $|f(a + i \cdot (b - a)/n)| < \varepsilon$ **then**
    Let $c = a + i \cdot (b - a)/n$, $i = n$, and *flag* = *true*.
  end if
end while
if *flag* = *false* **then**
  "Value of n was to small."
end if

Fig. 2.9 n-section method for finding the zeros of a continuous function f satisfying $f(a) f(b) < 0$

Algorithm 5 (Binary Search Method)
Input and *Output* as for Algorithm 4.

Set $l = a$ and $r = b$.
while $|f(l + (r - l)/2)| \geq \varepsilon$ **do**
  if $f(l) \cdot f(l + (r - l)/2) < 0$ **then**
    $r = l + (r - l)/2$
  else
    $l = l + (r - l)/2$
  end if
end while

Fig. 2.10 Binary-search method for finding zeros of a function f satisfying $f(a) f(b) < 0$

That binary-search method is much more time-efficient than the 'crude' n-section method. See Fig. 2.10 for pseudocode. It finds always a zero within the predefined accuracy limit.

Newton–Raphson method Regarding the calculation of zeros of f, let us suppose that we also have access to a calculation of derivatives $f'(x)$, for $x \in [a, b]$. The algorithm is shown in Fig. 2.11.[5] Values of the derivative may be approximated by difference quotients.

[5] It is named after *Isaac Newton* (see footnote on page 44) and *Joseph Raphson* (about 1648–about 1715).

Algorithm 6 (Newton–Raphson Method)
Input: Reals a and b; we also have a way to calculate $f(x)$ and $f'(x)$, for any $x \in [a, b]$.
Output: Value $c \in [a, b]$ as an approximate zero of f.

1: Let $c \in [a, b]$ be an initial guess for a zero.
2: **while** STOP CRITERION = false **do**
3: Replace c by $c - \frac{f(c)}{f'(c)}$
4: **end while**

Fig. 2.11 Newton–Raphson method for finding one zero of a smooth function f satisfying $f(a)f(b) < 0$, and having a derivative of constant sign in $[a, b]$

If f is a smooth function then the Newton–Raphson method is more time-efficient than the binary search method.

The initial value of c can be specified by a small number of binary-search steps for reducing run-time. A small $\varepsilon > 0$ is used for specifying the stop criterion in the Newton–Raphson method (i.e., "$|f(c)| > \varepsilon$?").

The caption of Fig. 2.11 states a condition which ensures that f has a single zero z in (a, b)—but the Newton–Raphson method will converge only if c is 'sufficiently close' to z. There is no practical way of deciding beforehand that c will give convergence to z, unless f satisfies additional conditions (which are not easy to test in general). For instance, apply Newton–Raphson to a quite simple function, and the values of c for which the algorithm does converge give the Mandelbrot set. If f satisfies the additional condition that $f''(x)$ has constant sign in $[a, b]$, then, if $f(b)$ has the same sign as $f''(x)$, the startpoint $c = b$ gives convergence to z, but otherwise the startpoint $c = a$ gives convergence to z.[6]

2.8 Cauchy's Mean-Value Theorem

The line segment $(a, f(a))(b, f(b))$ is a *chord* of the graph of f, and $\frac{f(b)-f(a)}{b-a}$ is the *slope* of this chord; see Fig. 2.12.

For $a < b$, assume a continuous function f that maps the closed interval $[a, b]$ into $\mathbb{R}$. Let f be differentiable on the open interval (a, b). *Cauchy's Mean-Value Theorem* says that there exists a real c, $a < c < b$, such that

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

[6]This paragraph was provided by *Garry Tee*, who also pointed out that a clear account of such convergence conditions (with illustrations) is given in Sim Borisovich Norkin's textbook *The Elements of Computational Mathematics*, Pergamon Press, Oxford, 1965.

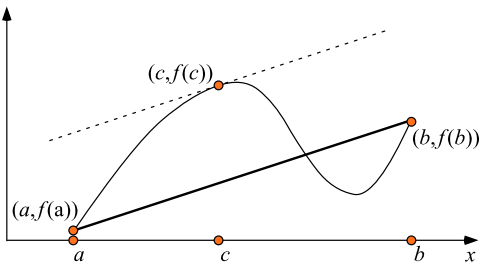

Fig. 2.12 Graph of a function f with a chord $pq = (a, f(a))(b, f(b))$. The *dashed line* shows the tangent at $(c, f(c))$

This theorem is not difficult to show;[7] basically, it says that there is a c such that the tangent at $(c, f(c))$ is parallel to the chord $(a, f(a))(b, f(b))$. In Fig. 2.12, there are two different points $(c, f(c))$ possible and only one of those is shown.

For the Newton–Raphson method it follows (from the mean-value theorem) that, if $f(a)$ and $f(b)$ have different signs and $f'(x)$ has a constant sign on $[a, b]$, then f has exactly one zero in (a, b).

2.9 Problems

Problem 2.1 Assume that we want to measure the length of straight line segments in the plane in a regular orthogonal grid after digitising them as follows (the so-called *grid intersection digitisation*): go on the given straight line segment from one end to the other; for every intersection of a grid line with the straight line, take the closest grid point as the next grid point (if there are two at equal distance, decide for the one closer to the origin); this maps a straight line segment into a polyline with a finite number of vertices (at grid points); see Fig. 2.13 for three examples.

Assume that the regular orthogonal grid has unit distance between grid lines. The resulting polylines have segments either of length 1 or of length $\sqrt{2}$. Use the total length of the polyline (i.e., sums of $1s$ and of $\sqrt{2}s$) as an estimator for the length of the original straight line segments. Does this define a δ-approximation? For what δ? Is this an algorithm with guaranteed error limits?

Problem 2.2 Show that any 2nd order polynomial $f(x) = a_0 + a_1x + a_2x^2$ is convex over the entire real line if $a_2 > 0$, and that any 3rd order polynomial $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ is not convex over the entire real line if $a_3 > 0$.

Problem 2.3 (Programming exercise) Your program compares two approaches for measuring the length of a curve.

(1) Consider a parabola $y = c_0 + c_1x + c_2x^2$, allowing that parameters c_0, c_1 and c_2 be selected by a user of the program.

[7] *Vatasseri Paramesvara* (ca. 1380–1460) studied already mean-value formulas for the sine function. The theorem is due to *A.-L. Cauchy* (see footnote on page 37).

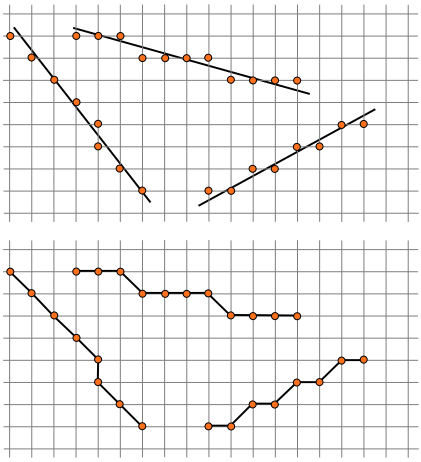

Fig. 2.13 *Top*: Three *straight line* segments with assigned *grid points* when using grid-intersection digitisation. *Bottom*: Resulting polylines

(2) Arclength of a parabola can readily be expressed in terms of elementary functions; see Definition 2.18. Use this analytic approach for calculating the arclength for $a = 0$ and $b = 100$.
(3) Now, in the main part of your program, provide an alternative way for measuring this length based on approximating the graph of the given polynomial on the interval $[0, 100]$ by a polyline at uniformly distributed x-values; take the sum of the lengths of all straight segments of the polyline as your approximate length estimate.

For (3), use different numbers $n_i > 0$ of uniformly distributed x-values, thus producing different length estimates L_i. Discuss the *speed of convergence* of those estimates towards the length obtained in Step (ii) in dependence of an increase in values n_i, using a sufficiently large number of n_i-values.

For an additional challenge, replace (i) by the following: For generating the input (i.e., the curve), now specify a 3rd order polynomial $p(x)$ by selecting parameters c_0, c_1, c_2, and c_3.

Calculate the length $\mathcal{L}(\gamma)$ of the graph $\gamma(\lambda) = (x(\lambda), p(x(\lambda)))$ of this polynomial, with $0 \leq \lambda \leq 1$, $x(0) = 0$, and $x(1) = 100$, now by using a numerical integration procedure for the integral in Definition 2.18. See Fig. 2.14 for one 3rd order polynomial and two examples with $n_1 = 5$ and $n_2 = 10$, as of relevance for Step (iii).

2.10 Notes

There are many different proposals for iterative thinning procedures; see, e.g., [9].

For materials on approximation algorithms, see the books [3, 7, 8, 12, 17] and the website [14]. For Definition 2.2, see, e.g., [7]. [6] describes a so-called "2-approximation linear algorithm" for calculating a shortest path on the surface of

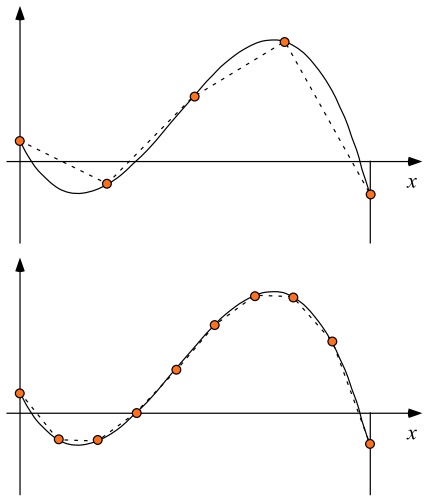

Fig. 2.14 *Top*: Graph of a 3rd-order polynomial approximated by a polyline defined by five uniformly sampled values. *Bottom*: Ten uniformly sampled values

a convex polyhedron, which is an example of an algorithm without guarantee. Example 2.1 (the Papadimitriou algorithm) is due to [13]; published in 1985, this is the first δ-approximation algorithm for solving a general 3D ESP problem.

Definition 2.5 differs from those in [3, 8, 12]; δ-approximation algorithms are considered in [7]. Some books, such as [3, 7, 8, 12, 14], also consider so-called "absolute" or "relative approximation", and so forth. These are schemes that are basically not much different from the concept of δ-approximation; for this reason we will not recall these concepts, and use δ-approximation as *the* concept of approximation algorithms in general in this book.

For basic definitions and theorems of mathematical analysis (Cauchy sequences, convergence, and so forth), see, e.g., [4]. Topology was introduced in [11]; for textbooks on topology, see, e.g., [1, 2]. For topology and the length of curves, see also [10]; this book also discusses topology and the *grid intersection digitisation* as used in Problem 2.1. For a discussion of convex functions, see [5, 15, 16].

References

1. Aleksandrov, P.S.: Combinatorial Topology, vol. 1. Graylock Press, Rochester (1956)
2. Aleksandrov, P.S.: Combinatorial Topology, vol. 2. Graylock Press, Rochester (1957)
3. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation. Springer, New York (1999)
4. Bartle, R.G., Sherbert, D.: Introduction to Real Analysis, 2nd edn. Wiley, New York (2000)
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge, UK (2004)
6. Hershberger, J., Suri, S.: Practical methods for approximating shortest paths on a convex polytope in $\mathbb{R}^3$. In: Proc. ACM-SIAM Sympos. Discrete Algorithms, pp. 447–456 (1995)
7. Hochbaum, D.S. (ed.): Approximation Algorithms for NP-Hard Problems. PWS, Boston (1997)
8. Hromkovič, J.: Algorithms for Hard Problems. Springer, Berlin (2001)

9. Klette, G.: Skeletal Curves in Digital Image Analysis. VDM, Saarbrücken (2010)
10. Klette, R., Rosenfeld, A.: Digital Geometry. Morgan Kaufmann, San Francisco (2004)
11. Listing, J.B.: Vorstudien zur Topologie. Göttinger Studien, 1. Abteilung math. und naturw. Abh., pp. 811–875. Several missing proofs were later published by Tait, P.G.: On knots. Proc. R. Soc. Edinb. **9**, 306–317 (1875–1878). A more recent review: Tripodi, A.: L'introduzione alla topologia di Johann Benedict Listing. Mem. Accad. Naz. Sci. Lett. Arti Modena **13**, 3–14 (1971)
12. Mayr, E.W., Prömel, H.J., Steger, A. (eds.): Lectures on Proof Verification and Approximation Algorithms. Springer, Berlin (1998)
13. Papadimitriou, C.H.: An algorithm for shortest path motion in three dimensions. Inf. Process. Lett. **20**, 259–263 (1985)
14. Rabani, Y.: Approximation algorithms. http://www.cs.technion.ac.il/~rabani/236521.04.wi.html (2006). Accessed July 2011
15. Roberts, A.W., Varberg, V.D.: Convex Functions. Academic Press, New York (1973)
16. Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton (1970)
17. Vazirani, V.V.: Approximation Algorithms. Springer, Berlin (2001)

http://www.springer.com/978-1-4471-2255-5

Euclidean Shortest Paths
Exact or Approximate Algorithms
Fajie, L.; Klette, R.
2011, XVIII, 378 p., Hardcover
ISBN: 978-1-4471-2255-5