Efficient plotting the functions with discontinuities based on combined sampling

Tomáš Bayer

Department of Applied Geoinformatics and Cartography, Faculty of Science, Charles University, Czech Republic
bayertom@natur.cuni.cz

Abstract. This article presents a new algorithm for interval plotting of the function $y = f(x)$ based on combined sampling. The proposed method synthesizes the uniform and adaptive sampling approaches and provides a more compact and efficient function representation. During the combined sampling, the polygonal approximation with a given threshold $\overline{\alpha}$ between the adjacent segments is constructed. The automated detection and treatment of the discontinuities based on the LR criterion are involved. Two implementations, the recursive-based and stack-based, are introduced. Finally, several tests of the proposed algorithms for the different functions involving the discontinuities and several map projection graticules are presented. The proposed method may be used for more efficient sampling the curves (map projection graticules, contour lines, or buffers) in geoinformatics.

Keywords: function; adaptive sampling; combined sampling; recursive approach; stack; discontinuity; polygonal approximation; visualization; map projection; plotting; GIS; Octave; Mathematica.

1. Introduction

A function $y = f(x)$ on interval $\Omega = [a, b]$ may have different form. For efficient plotting, its polygonal approximation needs to be constructed. A current approach concentrated on uniform sampling with the step δx may not be sufficient. Despite its popularity, the equally spaced points cannot describe its course without errors; the problems of undersampling or oversampling are common. Adaptive sampling brings several benefits, it adapts to a different curvature of the function, reduces the amount of redundant data and provides a natural and smooth plot of the function without the jumps and breaks. This technique is popular in computer graphics; recall the well-known deCasteljau or Chaikin's algorithms for the curve approximation. Combining the uniform and adaptive sampling approaches their advantages can be synthesized. The resulted representation is more compact, efficient and smooth.

A function may contain points of discontinuities that need to be detected. A subdivision of the given interval Ω, to the set of disjoint subintervals Ω_k^g without the internal singularities, containing only "good" data needs to be undertaken. In other words, the polygonal approximation of the function needs to be split into the several disjoint parts. The proposed method works by the requirements mentioned above. A broad set of the singularities can be detected and treated using the multiple criteria. Subsequently, for each interval Ω_k^g, a polygonal approximation of $f(x)$ is constructed using combined sampling. Since there are many sophisticated solutions built-in the high-end systems (Mathematica, Maple), our simple algorithm based on the recursive approach is efficient and easy-to-implement.

This paper is organized as follows. In Section 3, the combined sampling technique for 1D functions is presented. Section 4 deals with the detection of singularities, Section 5 describes the combined sampling of the discontinuous functions. In Section 6, the combined sampling

technique is tested on the set of several functions. Subsequently, its behavior on four map projections is analyzed.

2. Related Work

There are several strategies for plotting the function $y = f(x)$ on interval $\Omega = [a, b]$. The naive approach based on sampling of f in a fixed amount of the equally spaced points is described in [20]. The simple functions suffer from oversampling, while the oscillating curves are under-sampled; these issues are mentioned in [14]. Another approach based on the interval constraint plot constructing a hull of the curve was described in [6], [13], [20]. The automated detection of a useful domain and a range of the function is mentioned in [41]; the generalized interval arithmetic approach is described in [40].

A significant refinement is represented by adaptive sampling providing a higher sampling density in the higher-curvature regions. The are several algorithms for the curve interpolation preserving the speed, for example: [37], [42], [43]. The adaptive feed rate technique is described in [44]. An early implementation in the Mathematica software is presented in [39]. By reducing data, these methods are very efficient for the curve plotting. The polygonal approximation of the parametric curve based on adaptive sampling is mentioned in the several papers. The refinement criteria, as well as the recursive approach, are discussed in [15]. An approximation by the polygonal curves is described in [7], the robust method for the geometric and spatial approximation of the implicit curves can be found in [27], [10], the affine arithmetic working in the triangulated models in [32]. However, the map projections are never defined by the implicit equations. Similar approaches can be used for graph drawing [21].

Other techniques based on the approximation by the breakpoints can be found in many papers: [33], [9], [3]; these approaches are used for the polygonal approximation of the closed curves and applied in computer vision.

3. Combined sampling

In this section, the proposed combined sampling technique providing the polygonal approximation of the parametric curve involving the discontinuities will be presented. The modified method will be used for the function $f(x)$ reconstruction and plot. Based on the ideas of splitting the domain into the subintervals without the discontinuities, it represents a typical problem solvable by the recursive approach.

3.1. Polygonal approximation of the curve

Let $y = f(x)$, $M \subset \mathbb{R}$, $f : M \to \mathbb{R}$ be a function of a real variable x, and the set M represents the domain of the function f. Let $\Omega = [a, b]$, $a \in M$, $b \in M$ be the subdomain inside which the polynomial approximation $p_i = (x_i, f(x_i))$, $1 \leq i \leq n$ of the curve f is constructed, where $x_1 = a < x_1 < ... < x_n = b$. This approach leads to a discrete reconstruction of f from the set of sampled points.

The behavior of f should be reconstructed concerning its curvature. The classical approach based on uniform sampling from the equidistant points x_i with the step δ, where $\delta = x_{i+1} - x_i$, provides a good approximation only if $\delta \to 0$. For the straight parts of the curve, many

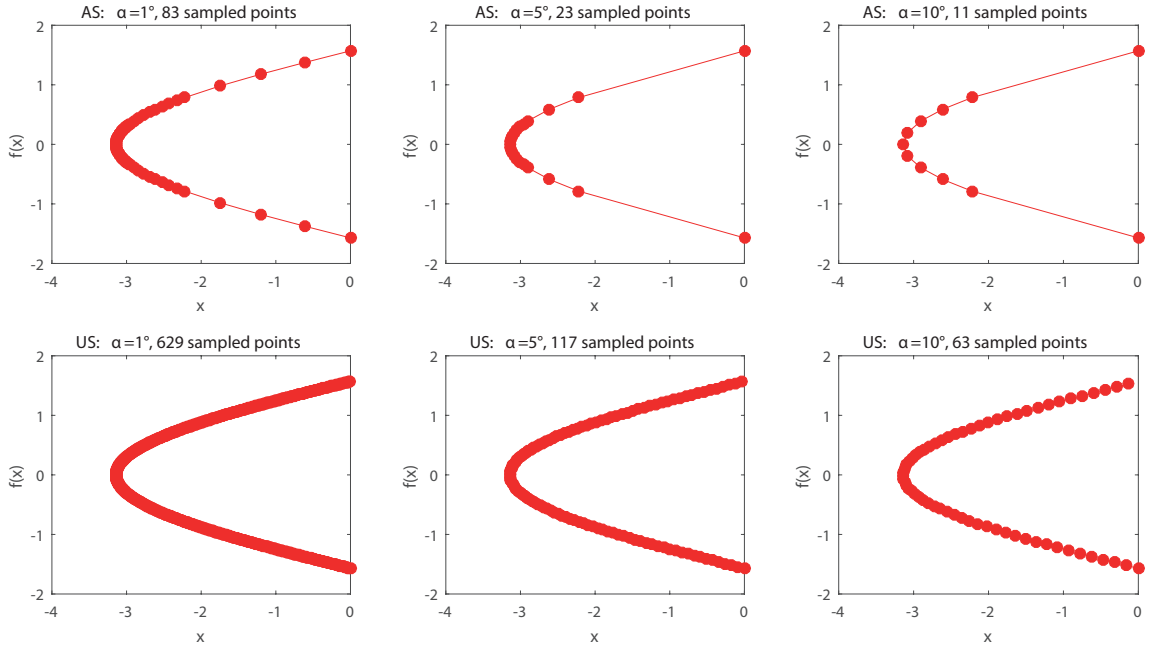

Figure 3.1: Adaptive (AS) and uniform (US) sampling of the meridian of the longitude $\lambda = -180°$ *in the Sanson projection for angles* $\alpha_i = 1°, 5°, 10°$.

almost colinear segments are constructed; too much redundant data is generated. Conversely, for larger δ, the shape of the function in the high-curvature areas may not be captured adequately, which is discussed in [15].

In general, the main disadvantages of uniform sampling, the problems of undersampling or oversampling, are referred. For a current density of the sample, the equally spaced points cannot describe the function course without errors.

3.2. Combined sampling technique

By avoiding colinear segments as well as a better adaptation to the different curvature, adaptive sampling respects the behavior of the function more naturally. It leads to the more compact data representation of the curve while its shape, as well as its aesthetic look, are preserved. A comparison of adaptive and uniform sampling for the meridian curve is illustrated in Fig. 3.1. Uniform sampling requires more data to maintain the same curvature represented by the angle α_i between the adjacent segments (p_{i-1}, p_i) and (p_i, p_{i+1}). The difference increases depending on the curvature. In general, for adaptive sampling, the required amount of points is about one order less. Unfortunately, two issues are referred in [15]:

1. For specific functions, some narrow subintervals in the early iterations may be skipped and stay unprocessed.
2. For some periodic functions, a refinement based on the iterative subdivision into the segments of the same length may not be successful, if the function values at these points are equal.

To avoid the problems in the first case, it is natural to take advantage of both methods

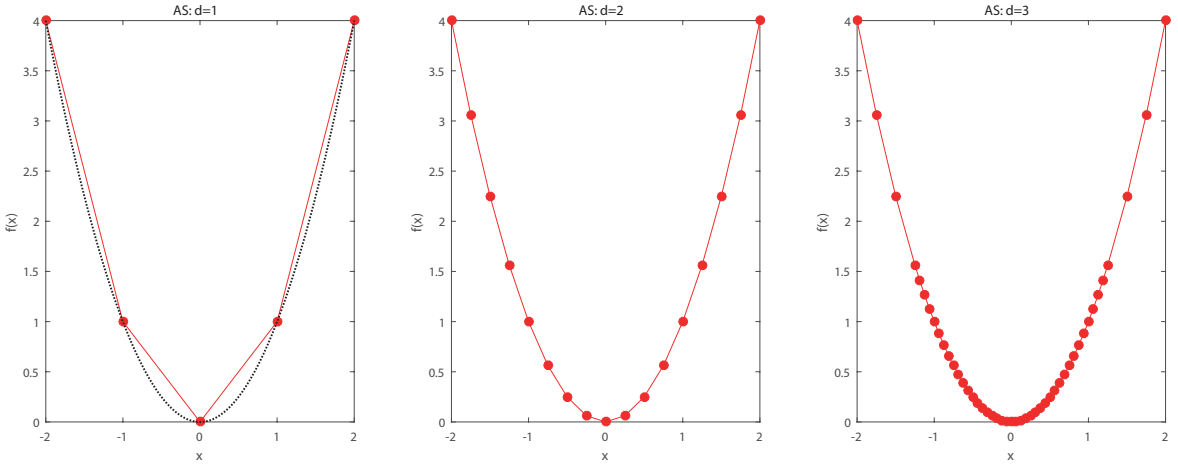

Figure 3.2: Illustration of the proposed algorithm: an adaptive sampling of the function $y = x^2$, $x \in [-2, 2]$ *for the depths of recursion* $d = 1, 2, 3$.

and propose the *combined sampling method*. Uniform sampling represents the initial step, later steps refining the curve approximation are provided by adaptive sampling. The second problem may overcome adding the partial randomness to the generated segments.

Refinement criteria. An important role is played by the refinement criteria smoothing the polyline [15]. Suppose (p_{i-1}, p_i, p_{i+1}) to be three consecutive sampled points of the curve. The primary criterion is represented by the angular difference of both segments

$$\alpha_i = \alpha(p_{i-1}, p_i, p_{i+2}) = \arccos \frac{|u \cdot v|}{\|u\| \, \|v\|} \pm \pi, \qquad u = p_{i+1} - p_i, \qquad v = p_{i-1} - p_i.$$

Different criteria can provide the analogous results: recall the distance of p_i from p_{i-1}, p_{i+1}, or, the local length ratio [29].

Combined sampling algorithm without the singularities

The combined sampling algorithm is based on the idea of the hierarchical reconstruction of the curve shape, which follows the recursive approach with the multiple calls mixing the uniform and adaptive techniques. Unlike a simple algorithm discussed in [15], the proposed method can handle the singularities and discontinuities of f and requires a lower recursion depth. Initially, sampling without the treatment of singularities will be presented.

Suppose d to be the current depth of the recursion, $\underline{d}$ to be the minimum and $\overline{d}$ to be the maximum recursion depth. Combined sampling returns a polynomial approximation of the curve by the refinement criteria $\overline{\alpha}$ and the recursion depth d. Our algorithm combines the uniform and adaptive sampling techniques and subdivides Ω into a specified number of the disjoint subintervals Ω_k of the similar size during each recursive step, where $k = 4$. Hence, Ω is split into the approximate quarters with the randomly shifting borders, which are not a direct multiple of 0.25.

Initially, if $d \leq \underline{d}$ the interval is subdivided into four subintervals regardless of α; four new segments of the polygonal approximation are created. Subsequently, when $d > \underline{d}$, between

Algorithm 1 Combined sampling, the initial phase.

```
1: function csInit(f, L, a, b, d, d̲, d̄, ε, ᾱ)
2:     L = ∅
3:     y_a = f(a), y_b = f(b)
4:     if discontinuity in a then
5:         throw SingularityException (a)
6:     if discontinuity in b then
7:         throw SingularityException (b)
8:     L ← Point(a, y_a)
9:     as(f, L, a, b, y_a, y_b, d, d̲, d̄, ε, α)
10:    L ← Point(b, y_b)
```

each pair of four new consecutive segments, the refinement criterion α_k, $k = 1, ..., 3$, is evaluated and compared to $\overline{\alpha}$. If $\alpha_k > \overline{a}$ and $b - a > \varepsilon$, two adjacent segments are created. The interval is subdivided into 2-4 new until the visually "smooth" polynomial approximation of the curve is obtained. In other words, uniform sampling is followed by adaptive sampling refining the properties of the insufficiently estimated segments.

Let $L = \{p_i\}_{i=1}^n$ be the polynomial approximation of $f(x)$ and $\Omega = [a, b]$ a subdomain. The algorithm may be summarized as follows:

1. *The initial phase*

   Let $L = \emptyset$ be the empty set. Compute $y_a = f(a)$ and $y_b = f(b)$. If a singularity in a or b is detected, throw the exception. Add the initial vertex to L: $L \leftarrow p_a$, where $p_a = (x_a, y_a)$. Set the recursion depth $d = 1$.

2. *The recursive step*

   Enter the recursive procedure and do the following substeps:

   (a) If $d > \overline{d}$ or $b - a < \varepsilon$ stop the recursive procedure and go to Step 3.

   (b) For a given $\Omega = [a, b]$, the interval is split by the three points

   $$x_1 = a + \frac{1}{2} r_1 (b - a), \qquad x_2 = a + r_2 (b - a), \qquad x_3 = a + \frac{3}{2} r_3 (b - a),$$

   into the approximate quarters

   $$\Omega_1 = [a, x_1], \qquad \Omega_2 = [x_1, x_2] \qquad \Omega_3 = [x_2, x_3], \qquad \Omega_4 = [x_3, b],$$

   where r_1, r_2, r_3 are the random numbers inside the interval $[0.45, 0.55]$. This step prevents a situation, when $f(a) = f(b) = f(x_1) = f(x_2) = f(x_3)$, but their intermediate points do not held this condition; it is typical for some periodic functions (for example, if $y = \sin 2x$, and $\Omega = [0, 2\pi]$).

   (c) If a singularity in x_1, x_2, or x_3 occurs, throw the new exception with the argument indicating the singularity.

   (d) Evaluate the function values $y_1 = f(x_1)$, $y_2 = f(x_2)$, and $y_3 = f(x_3)$ at new vertices p_1, p_2, p_3.

Algorithm 2 Combined sampling, the recursive phase.

```
1: function CS(f, L, a, b, y_a, y_b, d, d_, d¯, α¯)
2:     if d > d¯ ∨ (b − a < ε) then
3:         return
4:     r_1 = rand(0.45, 0.55), r_2 = rand(0.45, 0.55), r_3 = rand(0.45, 0.55)
5:     x_1 = a + (1/2) r_1 (b − a), x_2 = a + r_2 (b − a), x_3 = a + (3/2) r_3 (b − a)
6:     if discontinuity in x_i then
7:         throw SingularityException (x_i), i = 1, 2, 3
8:     y_1 = f(x_1), y_2 = f(x_2), y_3 = f(x_3)
9:     p_a = Point(a, y_a), p_b = Point(b, y_b), p_i = Point(x_i, y_i), i = 1, 2, 3
10:    α_1 = α(p_a, p_1, p_2), α_2 = α(p_1, p_2, p_3), α_3 = α(p_2, p_3, p_b)
11:    if (α_1 > α¯) ∨ (d <= d_) then
12:        cs(f, L, a, x_1, y_a, y_1, d + 1, d_, d¯, α¯);
13:    L ← Point(x_1, y_1)
14:    if (α_1 > α¯) ∨ (α_2 > α¯) ∨ (d <= d_) then
15:        cs(f, L, x_1, x_2, y_1, y_2, d + 1, d_, d¯, α¯);
16:    L ← Point(x_2, y_2)
17:    if (α_2 > α¯) ∨ (α_3 > α¯) ∨ (d <= d_) then
18:        cs(f, L, x_2, x_3, y_3, y_3, d + 1, d_, d¯, α¯);
19:    L ← Point(x_3, y_3)
20:    if (α_3 > α¯) ∨ (d <= d_) then
21:        cs(f, L, x_3, x_b, y_3, y_b, d + 1, d_, d¯, α¯);
```

(e) Check the refinement criteria $\alpha_1 = \alpha(p_a, p_1, p_2)$, $\alpha_2 = \alpha(p_1, p_2, p_3)$, $\alpha_3 = \alpha(p_2, p_3, p_b)$, and the recursive depth d. When the curve is not sufficiently smooth, or $d \leq \underline{d}$, it needs to be refined. For $d \leq \underline{d}$, this step begins with uniform sampling; for $d > \overline{d}$ it transforms to adaptive sampling.

(f) If $\alpha_1 > \overline{\alpha}$, call the recursive procedure with the increased depth $d = d + 1$ for the interval $[a, x_1)$.

(g) Add new point p_1 to the polynomial approximation of $f(x)$: $L \leftarrow p_1$.

(h) If $\alpha_1 > \overline{\alpha} \vee \alpha_2 > \overline{\alpha}$, call the recursive procedure with the increased depth $d = d + 1$ for the interval (x_1, x_2).

(i) Add new point p_2 to the polynomial approximation of $f(x)$: $L \leftarrow p_2$.

(j) If $\alpha_2 > \overline{\alpha} \vee \alpha_3 > \overline{\alpha}$, call the recursive procedure with the increased depth $d = d + 1$ for the interval (x_2, x_3).

(k) Add new point p_3 to the polynomial approximation of $f(x)$: $L \leftarrow p_3$.

(l) If $\alpha_3 > \overline{\alpha}$, call the recursive procedure for the interval $(x_3, x_b]$.

3. *Final step*

Add the last point p_b to the polynomial approximation of $f(x) : L \leftarrow p_b$ and finish the combined sampling procedure.

Method M3. By summarizing the facts mentioned above the proposed algorithm uses a triple recursion. In each step, the refined $f(x)$ is approximated by at least three new points. For the pseudocode, see Algs. 1, 2. Compared to [15], this solution has a dramatically improved performance and requires fewer subdivisions. Due to the significantly higher recursion depth d, for fast and efficient estimation of the $f(x)$ curvature, a single recursive step is not sufficient. This issue is illustrated in Tab. 1; our method is labeled as M3, a single recursion step method as M1.

4. Singularity detection

This section describes several simple strategies for handling and detecting the discontinuities; their overview can be found in [24], [2], [26]. Early methods are based on the Markov models [18], [28], [25]. Currently, there are several approaches: applications of the Fourier transformation [5], [12], [4], [23], [17], [16], [11], wavelets [35], [45], Chebyshev series [31], triangulations [19]. Geometric measures of the curvature are described in [38], [22], [34], [1], [30], [36], the statistical-based methods in [8], [26].

Unlike the solutions searching for all discontinuities of f at entire Ω, each sampled point x_i is checked for a discontinuity; the removable, jump and infinite discontinuities are involved. These testing criteria are local, they analyze behavior of the function in a boundary $B(x_i, \varepsilon)$ of x_i, covered by the equally spaced points $x_{i-2} = x_i - 2h$, $x_{i-1} = x_i - h$, and, $x_{i+1} = x_i + h$, $x_{i+2} = x_i + 2h$, where $h = \varepsilon/2$. For practical computation $\varepsilon = 0.001$. An infinite discontinuity is detected if

$$(|f_{i-k}| > \overline{y}) \vee (f_{i-k} \equiv \pm Inf) \vee (f_{i-k} \equiv NaN), \quad k = -2, .., 2.$$

where $\overline{y}$ is the given threshold, NaN and Inf are the symbols for the positive infinity and the result of the undefined operation. The remaining discontinuities may be found using the criteria measuring the smoothness by changes in the variation. Two criteria, WENO and LR, described in [30], are presented. The WENO criterion $w_j(x)$, $j = 0, 1, 2$, is written as

$$w_j(x) = \frac{\alpha_i}{\sum_{j=0}^{2} \alpha_i}, \quad \alpha_j = \frac{1}{(IS_j + \epsilon)^2},$$

where

$$\begin{aligned} IS_0 &= \frac{13}{12}(f_{i-2} - 2f_{i-1} + f_i)^2 + \frac{1}{4}(f_{i-2} - 4f_{i-1} + f_i)^2, \\ IS_1 &= \frac{13}{12}(f_{i-1} - 2f_i + f_{i+1})^2 + \frac{1}{4}(f_{i-1} - f_{i+1})^2, \\ IS_2 &= \frac{13}{12}(f_{i+2} - 2f_{i+1} + f_i)^2 + \frac{1}{4}(f_{i+2} - 4f_{i+1} + f_i)^2. \end{aligned}$$

If $w_0(x) > 1/3 \vee w_1(x) > 1/3 \vee w_2(x) > 1/3$, f is probably not smooth at x_i. The LR criterion has the following form

$$LR(x) = \frac{|f_r^2 - f_l^2|}{f_r^2 + f_l^2}, \tag{4.1}$$

where $f_r = 3f_i - 4f_{i+1} + f_{i+2}$, $f_l = 3f_i - 4f_{i-1} + f_{i-2}$. If $LR > 0.8$, f is probably not smooth at x_i. For practical computations, the criteria provide similar results. However, the WENO criterion seems to be more sensitive, and a steep slope classifies as the jump discontinuity. This issue is widely discussed in [30].

Table 1: The depth of recursion for different values of the refinement criteria α in the methods M1 and M3 (proposed).

Method	Refinement criterion $\alpha[^\circ]$								
	20	15	10	5	2	1	0.5	0.2	0.1
M1	7	9	9	19	55	105	217	535	1071
M3	3	3	3	7	15	27	57	121	297

5. Combined sampling with the singularities

The proposed method checks for a discontinuity at each sampled point $x_i \in \Omega$ using the rules mentioned above, where the amount of discontinuities denoted as k is not a priori known. It does not represent a rigid mathematical solution describing the behavior of the analyzed function, but only a simple method applicable to the technical computing. Our approach follows a heuristic sufficient for most functions, especially for the meridian or parallel coordinate functions. As a result, the set of disjoint subsets Ω^g, $\Omega^g \subseteq \Omega$, containing "good" data that allows for adaptive sampling is constructed; this technique was used in [26]. The point x_i is classified as "good" if no singularity at $f(x_i)$ occurs. An interval Ω containing only "good" points is classified as "good" and labeled as Ω^g. Unfortunately, the procedure cannot be generalized for higher-dimensional problems.

Suppose the $j-th$ interval $\Omega_j = [a_j, b_j]$ containing a singularity c, $c \in \Omega_j$, and ε, $\varepsilon > 0$, representing the numerical threshold. In general, several cases need to be distinguished:

- *Case 1: the coincidence with the lower bound*

  If $a_j \equiv c$, the discontinuity c coincides with the lower bound of the interval Ω_j.

- *Case 2: the proximity to the lower bound*

  If $a_j < c \wedge |c - a_j| < \varepsilon$, the discontinuity c is close the lower bound of the interval Ω_j.

- *Case 3: the coincidence with the upper bound*

  If $b_j \equiv c$, the discontinuity c coincides with the upper bound of the interval Ω_j.

- *Case 4: the proximity to the upper bound*

  If $b_j > c \wedge |c - b_j| < \varepsilon$, the discontinuity c is close the upper bound of the interval Ω_j.

- *Case 5: the interior singularity*

  If $a_j < c < b_j$, where $|c - a_j| > \varepsilon \wedge |c - b_j| > \varepsilon$, the discontinuity c lies inside the interval Ω_j.

For practical computation in the floating-point arithmetic, Cases 1, 2 and Cases 3, 4 may be joined, and we search for the discontinuity close to the lower or upper bounds. The modified conditions are:

- *Cases 1+2: the proximity/coincidence to the lower bound*

  If $a_j \leq c \wedge |c - a_j| < \varepsilon$, the discontinuity c coincides or it is close to the lower bound of the interval Ω_j.

- *Cases 3+4: the proximity/coincidence to the upper bound*

  If $b_j \geq c \wedge |c - b_j| < \varepsilon$, the discontinuity c coincides or it is close to the upper bound of the interval Ω_j.

Another two essential cases need to be resolved:

- *Case 6: Too narrow interval*

  If $b_j - a_j < \varepsilon$, an "empty" interval Ω_j arises.

- *Case 7: The incorrect interval*

  If $a_j > b_j$, the incorrect interval Ω_j appears.

In general, they occur due to the behavior of the sampled function as well as a result of the floating-point arithmetic. For our problem, the empty interval is "unpromising" and (probably) does not contain any important data[1]. Moreover, there is no chance of the possible improvement; the interval cannot be expanded. In most cases, the "incorrect" intervals are also empty, or they become empty during the next processing. In general, these intervals may be rejected from further processing; Cases 6, 7 can be solved simultaneously.

Depending on the position of the singularity c, three types of operations are performed:

- *Delete empty/incorrect Ω_j*

  The empty or incorrect interval Ω_j is deleted.

- *Shrinking Ω_j*

  The interval Ω_j with the discontinuity c close to the bounds is shrunk from the left/right. For Cases 1+2, $a_j = a_j + \varepsilon$, for Cases 3+4, $b_j = b_j - \varepsilon$.

- *Splitting Ω_j*

  The interval Ω_j with the internal discontinuity c is split so that the new disjoint intervals $\Omega_{j,1} = [a_j, c - \varepsilon]$, and $\Omega_{j,2} = [c + \varepsilon, b_j)$ are created.

It is obvious that the Case 6 appears as the result of the incorrect split or shrink operations, while the Case 7 is the result of the incorrect shrink operation.

5.1. Combined sampling algorithm involving the singularities

Let us summarize the facts mentioned above into the algorithm. Our implementation is based on the stack $\boldsymbol{S}$, see Alg. 3, the amount of Ω_j splits denoted s represents the recursion depth. The basic idea is to set $\Omega^g \equiv \Omega$, to loop over all the adaptively sampled points p_i, to check for a singularity c in the boundary $B(x_i, \varepsilon)$ of p_i and to make a decision about the Ω_j boundaries. If no singularity occurs, all points are classified as good, and the polygonal approximation L_j of the curve is constructed. All disjoint polygonal approximations L_j are stored in the list $\boldsymbol{L}$. The discontinuities are localized successively, one by one. The stack-based approach consists of two phases (initial and recursive):

[1] However, for some specific types of non-continuous functions this data may be important (functions with isolated points) and cannot be skipped.

Algorithm 3 Combined sampling with the singularities, the stack implementation.

```
1: function csSingStack(f, L, a_j, b_j, s̄, d̲, d̄, ε, ᾱ)
2:     S = ∅, s = 0
3:     S ← Ω_j = [a_j, b_j]
4:     while S ≠ ∅ do
5:         Ω_j = S.pop()
6:         try
7:             L_j = ∅
8:             csInit(f, L_j, a_j, b_j, 1, 1, d̲, d̄, ε, ᾱ)
9:             L ← L_j
10:        catch (SingularityException e)
11:            c = e.x
12:            if (s > s̄) then
13:                L = ∅
14:                return
15:            else
16:                k = 0, Ω_{j,1} = Ω_j, Ω_{j,2} = Ω_j
17:                processInt(Ω_j, c, Ω_{j,1}, Ω_{j,2}, k, s, ε)
18:                if (k > 0) then
19:                    S ← Ω_{j,1}
20:                else if (k > 1) then
21:                    S ← Ω_{2,1}
```

1. *The initial phase*

   Initialize the empty stack $\boldsymbol{S} = \emptyset$. Create $\Omega^g = [a, b]$ and push $S \leftarrow \Omega^g$.

2. *The recursive steps*

   Repeat the following steps until $\boldsymbol{S}$ is empty:

   (a) Pop the actual good interval $\Omega_j^g \leftarrow S$ from $\boldsymbol{S}$ and get a_j, b_j.

   (b) Create the empty list $L_j = \emptyset$.

   (c) Create the temporary polygonal approximation of f on $\Omega_j = [a_j, b_j]$ using combined sampling stored in L_j.

   (d) If no discontinuity appears, add L_j to $\boldsymbol{L}$: $\boldsymbol{L} \leftarrow L_j$ and go to step (a). Otherwise, c represents the discontinuity that must be treated in Steps e-h).

   (e) If $s > \overline{s}$, the maximum allowed recursion depth is exceeded without a reasonable solution. Clear the polygonal approximation $\boldsymbol{L}$.

   (f) Otherwise, initialize the newly created intervals $\Omega_{j,1} = [a_{j,1}, b_{j,1}]$, $\Omega_{j,2} = [a_{j,2}, b_{j,2}]$, as $\Omega_{j,1} = \Omega_j$, $\Omega_{j,2} = \Omega_j$, and the amount of created intervals as $k = 0$.

   (g) Call the function `processInt` with parameters Ω_j, c, $\Omega_{j,1}$, $\Omega_{j,2}$, k, $\boldsymbol{S}$, ε refining the interval Ω_j; see Alg. 4. The subintervals $\Omega_{j,1}$, $\Omega_{j,2}$ are passing by reference.

(h) If at least one new interval needs to be created, then $k > 0$. Push $\Omega_{j,1}$ to the stack: $\boldsymbol{S} \leftarrow \Omega_{j,1}$. If $k > 1$, push the second interval $\Omega_{j,2}$ to the stack: $\boldsymbol{S} \leftarrow \Omega_{j,2}$.

Processing the interval. The procedure `processInt`$(\Omega_j, c, \Omega_{j,1}, \Omega_{j,2}, k, \boldsymbol{S}, \varepsilon)$ refines the interval Ω_j. Depending on the c value, the Ω_j bounds are shifted, or Ω_j is split to $\Omega_{j,1}$, $\Omega_{j,2}$. It can be summarized as follows:

1. If $a_j > b_j$, Ω_j represents an incorrect interval, return.
2. If $|b_j - a_j| < \varepsilon$, Ω_j represents an "empty" interval, return. As mentioned above, for some functions with the multiple jump discontinuities the empty interval cannot be skipped.
3. If $a_j \leq c \wedge |c - a_j| < \varepsilon$, the discontinuity c is close to the lower bound of the interval Ω_j. Shift the lower bound $a_{j,1} = a_j + \varepsilon$ of $\Omega_{j,1}$ and increase the amount of created intervals $k = k + 1$.
4. If $b_j \geq c \wedge |c - b_j| < \varepsilon$, the discontinuity c is close to the upper bound of the interval Ω_j. Shift the upper bound $b_{j,1} = b_j - \varepsilon$ of $\Omega_{j,1}$ and increase the amount of created intervals $k = k + 1$.
5. If $a_j < c < b_j$, then $|c - a_j| > \varepsilon \wedge |c - b_j| > \varepsilon$, the discontinuity c is inside the interval Ω_j which needs to be split to $\Omega_{j,1}$, $\Omega_{j,2}$. Shift the upper bound $b_{j,1} = c - \varepsilon$ of $\Omega_{j,1}$ and the lower bound $a_{j,2} = c + \varepsilon$ of $\Omega_{j,2}$. Increase the amount of the created intervals $k = k + 2$ and splits $s = s + 1$.
6. If at least one new interval needs to be created, then $k > 0$. Push $\Omega_{j,1}$ to the stack: $\boldsymbol{S} \leftarrow \Omega_{j,1}$. If $k > 1$, push the second interval $\Omega_{j,2}$ to the stack: $\boldsymbol{S} \leftarrow \Omega_{j,2}$.

For the implementation, see Alg. 4. Analogously, the singularity value c is stored in the thrown exception, and processed in the `try-catch` block. In general, the stack-based implementation is more stable than a common recursion and does not suffer from too high value of the recursion depth d; especially for the small values of ε.

5.2. Utilization in geoinformatics

The proposed methods may be used for the polygonal approximation of curves when a more compact and efficient representation is required. The circles, circular arcs, ellipses or offsets of curves (e.g., buffers) cannot be internally stored in the *.shp files. It can also be used for the contour lines simplification (removing the adjacent segments, where $\alpha_i < \overline{\alpha}$). Another utilization for which the algorithm was originally developed, is a more efficient reconstruction of the map projection graticule. The coordinate functions $F(\varphi, \lambda)$, $G(\varphi, \lambda)$ of the map projection depend on φ, λ and may contain several discontinuities. Therefore, the problem needs to be generalized, and its solution must be adapted for these facts. The algorithm for the map projection graticule construction will be presented in the next paper.

6. Experiments and results.

The proposed methods for combined sampling have been tested for the set of 9 functions; the ability to detect and treat the discontinuities represents an important factor.

Algorithm 4 Processing the interval: shift bounds or split.

```
1: function processInt(Ω_j, c, Ω_{j,1}, Ω_{j,2}, k, s, ε)
2:     if (a_j > b_j) then
3:         return
4:     if |b_j − a_j| < ε then
5:         return
6:     if (a_j ≤ c) ∧ (|c − a_j| ≤ ε) then
7:         a_{j,1} = a_j + ε
8:         k = k + 1
9:     else if (b_j ≥ c) ∧ (|c − b_j| ≤ ε) then
10:        b_{j,1} = b_j − ε
11:        k = k + 1
12:    else if (a_j < c < b_j) ∧ (|c − a_j| > ε) ∧ (|c − b_j| > ε) then
13:        b_{j,1} = c − ε, a_{j,2} = c + ε
14:        k = k + 2
15:        s = s + 1
```

6.1. List of functions

In all cases is $\Omega = [-5\pi, 5\pi]$, $\varepsilon = 0.001$, $\overline{\alpha} = 1°$, the thresholds $\overline{s}$, $\overline{d}$ have not been set. The first function

$$f_1(x) = \begin{cases} 1 + x, & x \geq 0, \\ 0, & x < 0, \end{cases}$$

has the jump discontinuity at $x = 0$. The polyline representation contains only 4 points divided into two intervals $\Omega_1^g = [-5\pi, -1.3 \cdot 10^{-4}]$, $\Omega_2^g = [1.6 \cdot 10^{-4}, 5\pi]$, the amount of splits is $s = 1$, no recursion is required $d = 0$. The second function

$$f_2(x) = e^{-500x^2},$$

is quite steep and does not contain any discontinuity. These facts led to the polygonal approximation formed by 266 points, $\Omega^g = \Omega$, no splits, $s = 0$, the maximum recursion depth is $d = 3$. The third function

$$f_3(x) = \sin(x)/x,$$

has a removable discontinuity at $x = 0$, where $f(0) = 1$. Ω is divided into two subintervals $\Omega_1^g = [-5\pi, -1.3 \cdot 10^{-4}]$, $\Omega_2^g = [1.6 \cdot 10^{-4}, 5\pi]$, the amount of splits is $s = 1$, no recursion required $d = 0$, the polygonal approximation contains 122 points. The fourth function

$$f_4(x) = x/\sin(x),$$

has jump discontinuities at $x = \pm\pi \pm k\pi$. Hence, Ω is split into 9 sub intervals: $\Omega_1^g = [-15.692, -12.579]$, $\Omega_2^g = [-12.554, -9.434]$, $\Omega_3^g = [-9.415, -6.290]$, $\Omega_4^g = [-6.277, -3.145]$, $\Omega_5^g = [-3.138, 3.138]$, $\Omega_6^g = [3.145, 6.277]$, $\Omega_7^g = [6.290, 9.415]$, $\Omega_8^g = [9.434, 12.554]$, $\Omega_9^g = [12.579, 15.692]$. The amount of splits $s = 8$ as well as the recursion depth $d = 6$, provide the polygonal approximation containing 984 points. The next function

$$f_5(x) = x \sin(5/x),$$

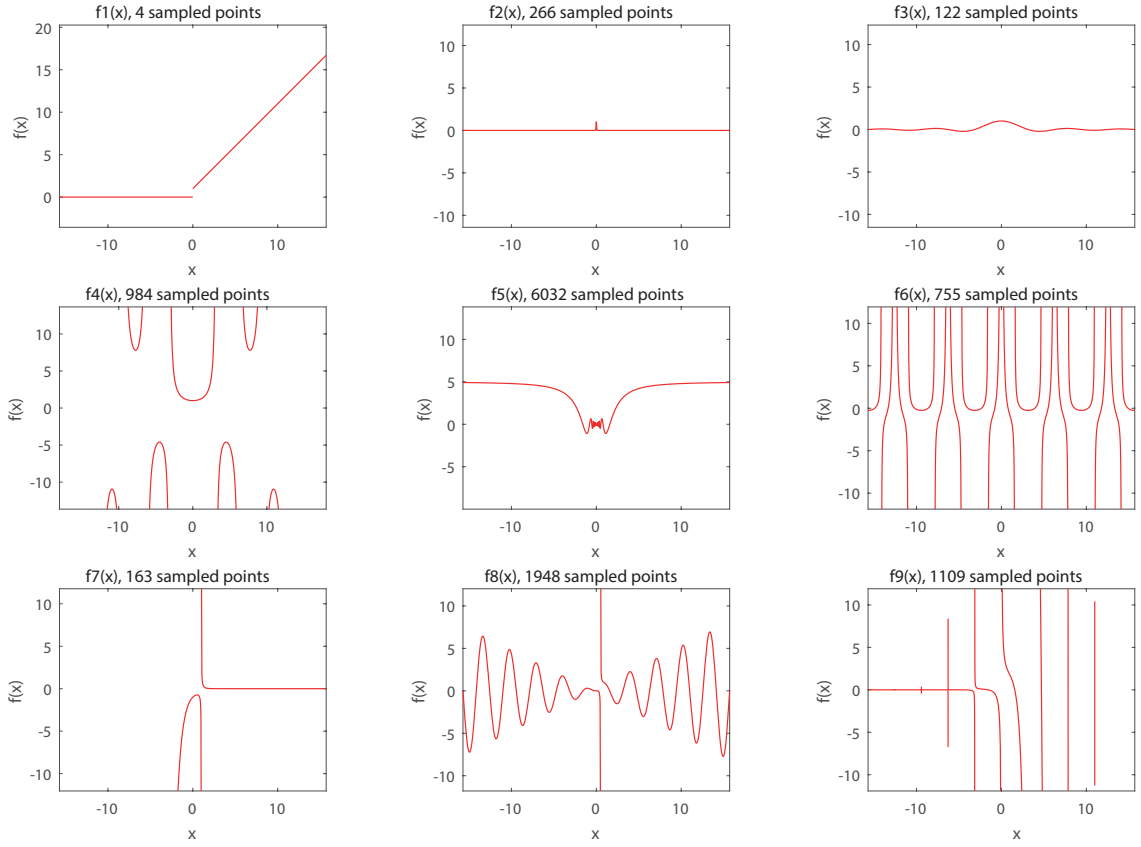

Figure 5.1: The polygonal approximation of functions $f_1(x), ..., f_9(x)$ created by the combined sampling technique; the discontinuities involved.

has a discontinuity at $x = 0$, and $f(0) = 0$. Ω should be divided in two subintervals, but the algorithm creates 12 subintervals in 42 splits: $\Omega_1^g = [-15.708, -0.015]$, $\Omega_2^g = [-0.009, -0.008]$, $\Omega_3^g = [-0.007, -0.006]$, $\Omega_4^g = [-0.005, -0.004]$, $\Omega_5^g = [-0.004, -0.003]$, $\Omega_6^g = [-0.003, 0.003]$, $\Omega_7^g = [0.003, 0.004]$, $\Omega_8^g = [0.004, 0.005]$, $\Omega_9^g = [0.005, 0.006]$, $\Omega_{10}^g = [0.006, 0.007]$, $\Omega_{11}^g = [0.008, 0.009]$, $\Omega_{12}^g = [0.015, 15.708]$. As a result of the recursion depth $d = 9$, the total amount of points is $n = 6032$. While the graphical representation is aesthetically pleasing, the polygonal representation contains redundant data; this is due to the false detection of discontinuities provide by LR criterion. The next function

$$f_6(x) = 1/(\tan 2x \tan 0.5x),$$

has the infinite singularities at $x = \pm 1/2\pi \pm k\pi$, and at $x = 0 \pm k2\pi$ (total 15). The adaptive sampling procedure with the recursion depth $d = 7$ creates 755 points in 16 new intervals $\Omega_1^g = [-15.708, -14.138$, $\Omega_2^g = [-14.137, -12.598]$, $\Omega_3^g = [-12.535, -10.996]$, $\Omega_4^g = [-10.995 - 7.855]$, $\Omega_5^g = [-7.853, -6.315]$, $\Omega_6^g = [-6.251, -4.713]$, $\Omega_7^g = [-4.712, -1.571]$, $\Omega_8^g = [-1.570, -0.032]$, $\Omega_9^g = [0.032, 1.570]$, $\Omega_{10}^g = [1.571, 4.712]$, $\Omega_{11}^g = [4.713, 6.251]$, $\Omega_{12}^g = [6.315, 7.853]$, $\Omega_{13}^g = [7.855, 10.995]$, $\Omega_{14}^g = [10.996, 12.535]$, $\Omega_{15}^g = [12.598, 14.137]$, $\Omega_{16}^g = [14.138, 15.708]$ with $s = 15$ splits. The next function

$$f_7(x) = e^{-2x}/(x - 1),$$

Algorithm 5 Plotting the function $f_9(x)$ involving discontinuities in the Octave v. 4.4 scripting language.

```
xmin = -5*pi; xmax = -xmin;
eps = 0.001; lrmax = 0.8;
x = xmin:0.001:xmax;
f = @(x)exp(x)./tan(x);
y = f(x);
d = lr( f, x, eps) > lrmax;
x(d==1) = nan;
y(d==1) = nan;
plot(x,y,'-r');
xlim([xmin,xmax])
ylim([xmin,xmax])
set(gca,'XLim',[xmin xmax])
set(gca,'YLim',[xmin xmax]])
xlabel('x');
ylabel('f(x)');
axis equal;
```

has the infinite singularity at $x = 1$, Ω is divided into two subintervals $\Omega_1^g = [-4.286, 1.000]$, $\Omega_2^g = [1.000, 15.708]$. The amount of splits is $s = 1$ as well as the recursion depth $d = 6$, provide the polygonal approximation by 163 points. The next function

$$f_8(x) = x^2 \sin 2x/(2x - 1),$$

has the infinity singularity at $x = 0.5$, Ω is divided into two subintervals $\Omega_1^g = [-15.708, 0.500]$, $\Omega_2^g = [0.500, 15.708]$. The amount of splits is $s = 1$ and the recursion depth $d = 7$ bring the polygonal approximation by 1948 points. The last function

$$f_9(x) = e^x / \tan x,$$

has infinite singularities at $x = \pm k\pi$ (total 11). The adaptive sampling procedure with the recursion depth $d = 6$ creates 1109 points in 10 new intervals $\Omega_1^g = [-15.708, -12.567]$, $\Omega_2^g = [-12.566, -9.425]$, $\Omega_3^g = [-9.425, -6.283]$, $\Omega_4^g = [-6.2837, -3.142]$, $\Omega_5^g = [-3.141, -0.001]$, $\Omega_6^g = [0.001, 3.119]$, $\Omega_7^g = [3.165, 5.925]$, $\Omega_8^g = [7.224, 8.138]$, $\Omega_9^g = [10.979, 11.012]$, $\Omega_{10}^g = [14.137, 14.138]$, with $s = 10$ splits.

The polygonal approximations of all analyzed functions can be found in Fig. 5.1. It is evident that their courses have been estimated correctly. However, for some functions, the proposed combined sampling algorithm brings less redundant representation (f_5). In general, this method cannot compete with the well-known high-end solutions (Mathematica) involving the robust numerical techniques, but it may provide a reasonable representation of functions in technical computing.

6.2. Comparison with other systems

For comparison, the functions will be plotted in two well-known systems. While the open-source software is represented by Octave (v. 4.4), the commercial software by Wolfram Mathematica (v. 11).

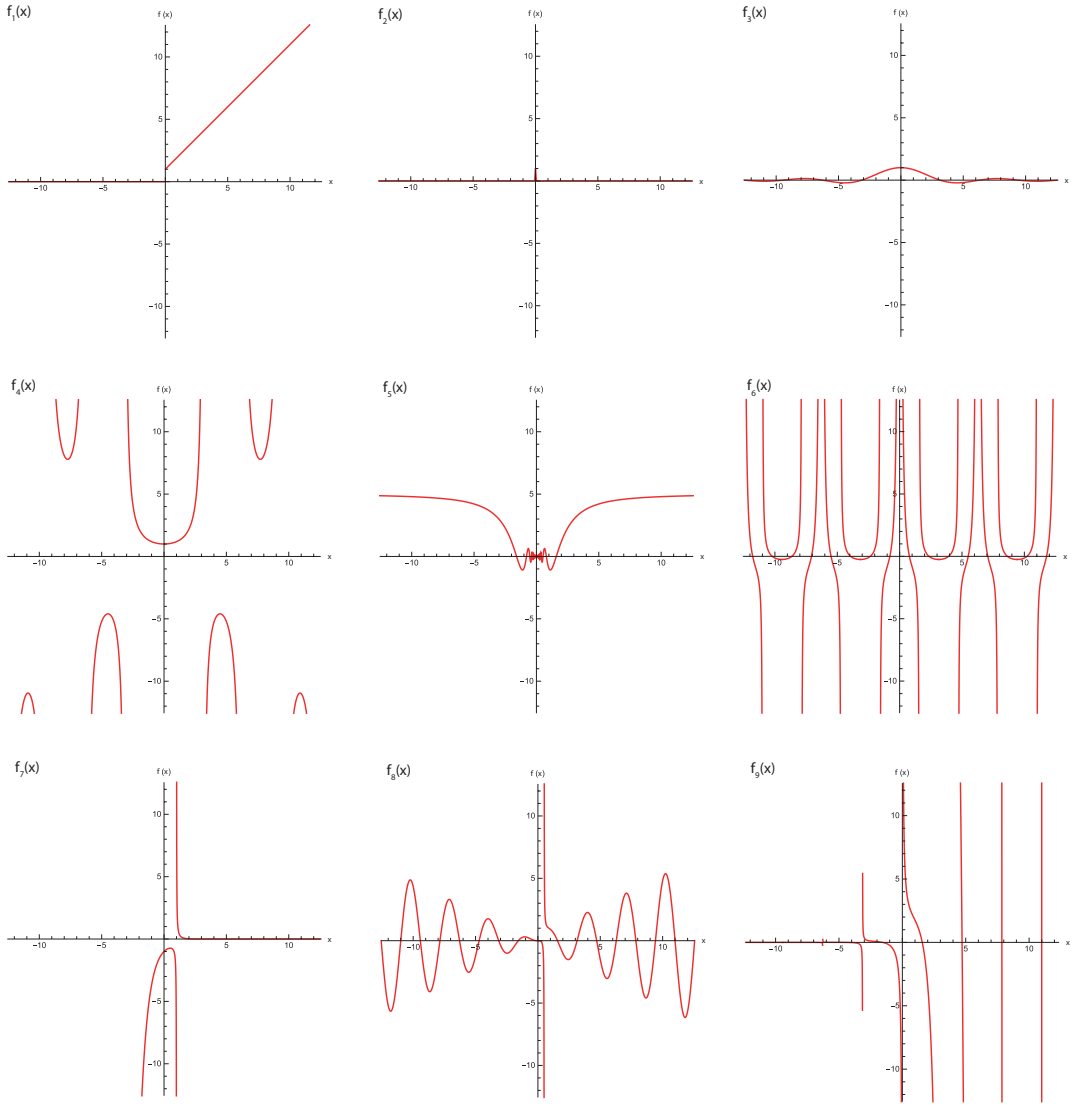

Figure 6.1: Functions $f_1(x), ..., f_9(x)$ plotted in the Wolfram Mathematica software, v. 11.

Octave. Unfortunately, the open-source software Octave (v. 4.4) does not support a correct plotting the functions involving discontinuities. The main idea of our solution is based on splitting Ω to the subintervals Ω^g by putting NaN numbers between Ω^g, where the function is undefined. The discontinuities are detected by the LR criterion given by Eq. 4.1. From a mathematic point of view, more characteristics of the function behavior need to be studied (first and second derivatives, asymptotes, local/global minima, ...), but this is outside the scope of the paper. The script can be found in Alg. 5.

The previous results set the numerical characteristics: $\Omega = [-5\pi, 5\pi]$, $\varepsilon = 0.001$, $\overline{LR} = 0.8$, sampling step $h = 0.001$. A curve is sampled uniformly by 10000π (approx. 31 000) points; this "relatively large" value has been set empirically. In general, the obtained results are analogous to the proposed method. Unfortunately, the algorithm is sensitive to the values of h, and $\overline{LR}$. For the larger values of h, $h = 0.01$, only a subset of functions is sampled

Table 2: Uniform and combined sampling of the graticules of the equal area cylindrical, conic, azimuthal and Werner-Staab projections using combined sampling. The quantitative parameters are presented.

Projection	Sampling	n_{mer}	n_{par}	$\overline{\alpha}_m$	$\overline{\alpha}_p$	$\widetilde{\alpha}_m$	$\widetilde{\alpha}_p$
Equal area cylindrical	Uniform	1443	1406	0.00	0.00	0.00	0.00
	Combined	195	190	0.00	0.00	0.00	0.00
Equal area conic	Uniform	1443	1406	0.00	2.20	0.00	2.20
	Combined	195	1612	0.00	5.01	0.00	1.89
Equal area azimuthal	Uniform	1443	1406	0.00	5.00	0.00	5.00
	Combined	195	2476	0.00	4.74	0.00	2.81
Werner-Staab	Uniform	1443	1406	9.27	5.00	3.86	2.93
	Combined	2334	1747	4.99	5.00	2.36	2.34

correctly.

Mathematica. Wolfram Mathematica v. 11, the well-known system for technical computing, developed for three decades, supports automatic plotting the functions with the discontinuities. The script has a straightforward form; see Alg. 6. For the functions $f_1(x), ..., f_8(x)$ the analogous discontinuities have been detected. Unfortunately, for f_9, the asymptote $x = -3\pi$ has not been recognized; see Fig. 6.1. Moreover, the asymptote $x = -2\pi$ is hard to distinguish. If we understand Mathematica as the reference software, our proposed algorithm may be found as a simple and efficient tool for plotting a general function of the one variable. However, many situations may appear when it fails.

Algorithm 6 Plotting the function $f_9(x)$ involving the discontinuities in the Mathematica v. 11 scripting language.

```
xmin = -5*Pi; xmax = -xmin;
F[x_] := Exp[x]/Tan[x]};
Plot[F[x], x, xmin, xmax, PlotRange -> xmin, xmax, xmin, xmax, AxesLabel -> x, f[x],
AxesOrigin -> 0, 0, AspectRatio -> Automatic, PlotStyle -> Red];
```

6.3. Construction of the map projection graticule

In the last test, where the graticules of several map projections are reconstructed, the uniform and adaptive sampling techniques are compared regarding the data representation compactness. It is measured by the amount of the sampled meridian points n_{mer}, and parallel points n_{par}. Maximum angles $\overline{\alpha}_m, \overline{\alpha}_p$ between the sampled meridian and parallel segments together with their mean values $\widetilde{\alpha}_m, \widetilde{\alpha}_p$ are measured. The graticule is constructed over the entire planisphere, so $\Omega = \Omega_\varphi \times \Omega_\lambda$, where $\varphi \in [-\pi/2, \pi/2]$, and $\lambda \in [-\pi, \pi]$, the offsets between the meridians and parallels are $\Delta\varphi = \Delta\lambda = 10°$. In uniform sampling, the sampling steps of the meridians and parallels are $\delta_\varphi = \delta_\lambda = 2°$; combined sampling uses $\overline{\alpha} = 2°$. For the circular arcs, uniform and combined sampling provide the analogous density of points representing the polygonal approximation. On the contrary, uniform sampling brings a higher density of the sampled points for the straight lines and vice versa for most of the curves. Four map

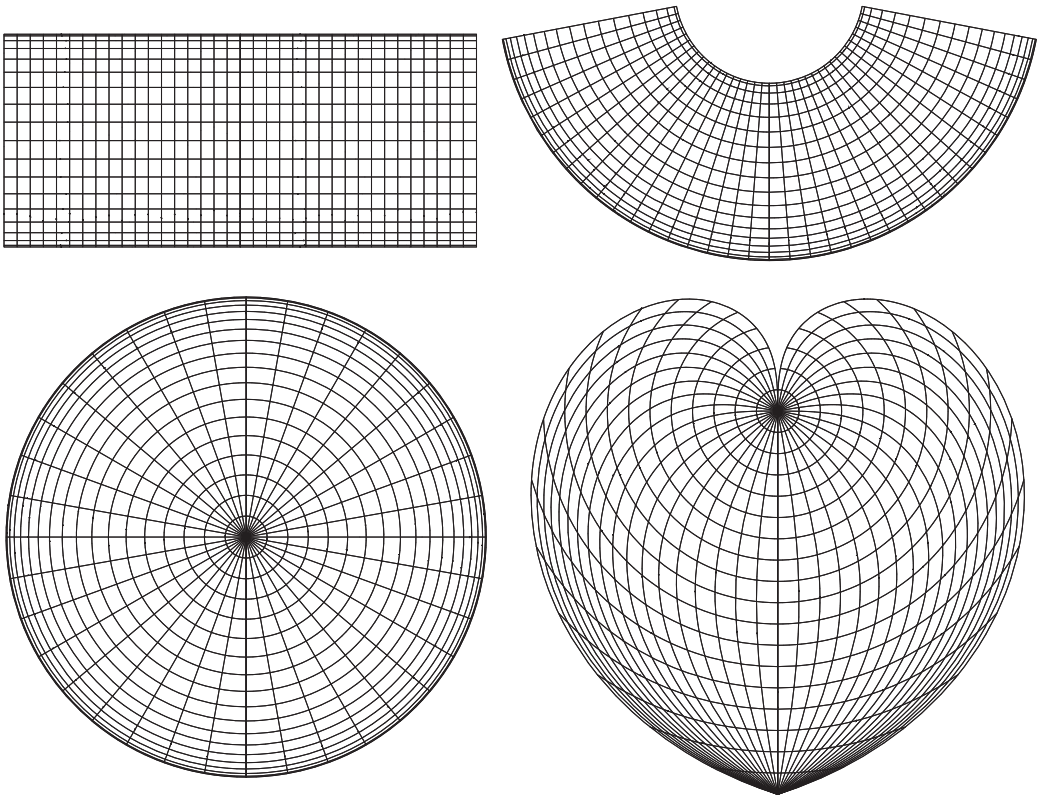

Figure 6.2: The reconstructed graticules of the equal area cylindrical, conic, azimuthal and Werner-Staab projections using the combined sampling technique.

projections are involved in testing: equal-area cylindrical, conic ($\varphi_1 = 45°$), azimuthal, and Werner-Staab, their coordinate functions are continuous on Ω. The results are summarized in Tab. 2.

For the straight segments, uniform sampling provides the redundant data; this issue refers to the cylindrical projection as well as to the meridians of the conic and azimuthal projections. While the constant curvature leads to the similar results (parallels of conic, azimuthal and Werner-Staab projections), in the higher-curvature regions, the combined sampling provides a smoother approximation (meridians of Werner-Staab projection). In general, combined sampling preserves the curvature better (see $\overline{\alpha}_m, \overline{\alpha}_p$, $\widetilde{\alpha}_m, \widetilde{\alpha}_p$ values) and brings less redundant data which requires more sampled points. The reconstructed graticules can be found in Fig. 6.2. The modified version of the algorithm treating the discontinuities in the coordinate functions F, G will be presented in the next paper.

7. Conclusion

This article presented a new algorithm combining the uniform and adaptive sampling techniques applicable to the functions involving the discontinuities, which are detected by the LR criterion. It can be used for the polygonal approximation of the curves when a more compact, efficient and less redundant representation is required. A typical example is represented by the circles, circular arcs, ellipses or offsets of curves, but it can be easily applicable to the map

projection graticule and similar problems in GIS and cartography. The illustrating examples indicate that the functions involving the discontinuities widely applied in technical practice can be plotted efficiently. However, there are many more complex functions, where the proposed solutions are not sufficient and need to be refined. A typical situation is represented by the isolated point, which is currently thrown. Another benefit is the relatively simple stack-based implementation. The source code written in Java can be found in the GitHub repository

https://github.com/bayertom/sampling.

The next paper brings an extension of this algorithm for combined sampling of the map projection graticules, when the coordinate functions F, G involve the discontinuities.

References

[1] Francesc Arandiga et al. "Interpolation and Approximation of Piecewise Smooth Functions". In: *SIAM Journal on Numerical Analysis* 43.1 (2005), pp. 41–57. DOI: 10.1137/S0036142903426245.

[2] Rick Archibald, Anne Gelb, and Jungho Yoon. "Determining the locations and discontinuities in the derivatives of functions". In: *Applied Numerical Mathematics* 58.5 (2008), pp. 577–592.

[3] André Ricardo Backes and Odemir Martinez Bruno. "Polygonal approximation of digital planar curves through vertex betweenness". In: *Information Sciences* 222 (2013), pp. 795–804.

[4] Nana S. Banerjee, James F. Geer, and Langley Research Center. *Exponential approximations using fourier series partial sums [microform] / Nana S. Banerjee and James F. Geer*. English. National Aeronautics and Space Administration, Langley Research Center ; National Technical Information Service, distributor Hampton, Va. : [Springfield, VA, 1997, p. 1 v.

[5] Robert Bruce Bauer. "Band pass filters for determining shock locations". In: ().

[6] Frédéric Benhamou and William J. Older. "Applying interval arithmetic to real, integer, and boolean constraints". In: *The Journal of Logic Programming* 32.1 (1997), pp. 1–24. ISSN: 0743-1066. DOI: https://doi.org/10.1016/S0743-1066(96)00142-2.

[7] P. Binev et al. "Adaptive approximation of curves". In: *Approximation Theory* (2004), pp. 43–57.

[8] Mira Bozzini and Milvia Rossini. "The detection and recovery of discontinuity curves from scattered data". In: *Journal of Computational and Applied Mathematics* 240.Supplement C (2013). MATA 2012, pp. 148–162. ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2012.06.014.

[9] A Carmona-Poyato et al. "Polygonal approximation of digital planar curves through break point suppression". In: *Pattern Recognition* 43.1 (2010), pp. 14–25.

[10] Filipe de Carvalho Nascimento et al. "Approximating implicit curves on plane and surface triangulations with affine arithmetic". In: *Computers & Graphics* 40 (2014), pp. 36–48.

[11] Dennis Cates and Anne Gelb. "Detecting derivative discontinuity locations in piecewise continuous functions from Fourier spectral data". In: *Numerical Algorithms* 46.1 (2007), pp. 59–84.

[12] Knut S Eckhoff. "Accurate reconstructions of functions of finite regularity from truncated Fourier series expansions". In: *Mathematics of Computation* 64.210 (1995), pp. 671–690.

[13] M.H. van Emden. "Value Constraints in the CLP Scheme". In: *Constraints* 2.2 (Oct. 1997), pp. 163–183. ISSN: 1572-9354. DOI: 10.1023/A:1009705709733.

[14] Richard Fateman. "Honest Plotting, Global Extrema, and Interval Arithmetic". In: *Papers from the International Symposium on Symbolic and Algebraic Computation*. ISSAC '92. Berkeley, California, USA: ACM, 1992, pp. 216–223. ISBN: 0-89791-489-9. DOI: 10.1145/143242.143314.

[15] Luiz Henrique de Figueiredo. "Adaptive sampling of parametric curves". In: *Graphics Gems V* 5 (1995), pp. 173–178.

[16] Anne Gelb and Eitan Tadmor. "Adaptive edge detectors for piecewise smooth data based on the minmod limiter". In: *Journal of Scientific Computing* 28.2 (2006), pp. 279–306.

[17] Anne Gelb and Eitan Tadmor. "Spectral reconstruction of piecewise smooth functions from their discrete data". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 36.2 (2002), pp. 155–175.

[18] Stuart Geman and Donald Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images". In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1984), pp. 721–741.

[19] Tim Gutzmer and Armin Iske. "Detection of discontinuities in scattered data approximation". In: *Numerical Algorithms* 16.2 (1997), pp. 155–170.

[20] Timothy J. Hickey, Zhe Qju, and Maarten H. Van Emden. "Interval Constraint Plotting for Interactive Visual Exploration of Implicitly Defined Relations". In: *Reliable Computing* 6.1 (Feb. 2000), pp. 81–92. ISSN: 1573-1340. DOI: 10.1023/A:1009950630139.

[21] Yifan Hu. "Efficient, high-quality force-directed graph drawing". In: *Mathematica Journal* 10.1 (2005), pp. 37–71.

[22] Guang-Shan Jiang and Chi-Wang Shu. "Efficient implementation of weighted ENO schemes". In: *Journal of computational physics* 126.1 (1996), pp. 202–228.

[23] George Kvernadze. "Determination of the jumps of a bounded function by its Fourier series". In: *Journal of approximation theory* 92.2 (1998), pp. 167–190.

[24] David Lee. "Detection, Classification, and Measurement of Discontinuities". In: *SIAM Journal on Scientific and Statistical Computing* 12.2 (1991), pp. 311–341. DOI: 10.1137/0912018.

[25] David Lee and Grzegorz W Wasilkowski. "Discontinuity detection and thresholding-a stochastic approach". In: *Journal of Complexity* 9.1 (1993), pp. 76–96.

[26] Licia Lenarduzzi and Robert Schaback. "Kernel-based adaptive approximation of functions with discontinuities". In: *Applied Mathematics and Computation* 307.Supplement C (2017), pp. 113–123. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2017.02.043.

[27] Hélio Lopes, João Batista Oliveira, and Luiz Henrique de Figueiredo. "Robust adaptive polygonal approximation of implicit curves". In: *Computers & Graphics* 26.6 (2002), pp. 841–852.

[28] Jose Marroquin, Sanjoy Mitter, and Tomaso Poggio. "Probabilistic solution of ill-posed problems in computational vision". In: *Journal of the american statistical association* 82.397 (1987), pp. 76–89.

[29] Byron Nakos and V Miropoulos. "Local length ratio as a measure of critical points detection for line simplification". In: *Fifth Workshop on Progress in Automated Map Generalization, Paris, France*. 2003.

[30] M Oliveria et al. "Universal high order subroutine with new shock detector for shock boundary layer interaction". In: *In other words* 10 (2009), pp. 1–2.

[31] Ricardo Pachón, Rodrigo B Platte, and Lloyd N Trefethen. "Piecewise-smooth chebfuns". In: *IMA journal of numerical analysis* 30.4 (2009), pp. 898–916.

[32] Afonso Paiva et al. "Approximating implicit curves on triangulations with affine arithmetic". In: *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on*. IEEE. 2012, pp. 94–101.

[33] Mohammad Tanvir Parvez and Sabri A Mahmoud. "Polygonal approximation of digital planar curves through adaptive optimizations". In: *Pattern Recognition Letters* 31.13 (2010), pp. 1997–2005.

[34] Leszek Plaskota and Grzegorz W Wasilkowski. "Adaption allows efficient integration of functions with unknown singularities". In: *Numerische Mathematik* 102.1 (2005), pp. 123–144.

[35] M. Rossini. "2D-discontinuity detection from scattered data". In: *Computing* 61.3 (Sept. 1998), pp. 215–234. ISSN: 1436-5057. DOI: 10.1007/BF02684351.

[36] Yiqing Shen and Gecheng Zha. "Improvement of the WENO scheme smoothness estimator". In: *International Journal for Numerical Methods in Fluids* 64.6 (2010), pp. 653–675. ISSN: 1097-0363. DOI: 10.1002/fld.2168.

[37] Moshe Shpitalni, Yoram Koren, and CC Lo. "Realtime curve interpolators". In: *Computer-Aided Design* 26.11 (1994), pp. 832–838.

[38] Kaleem Siddiqi, Benjamin B Kimia, and Chi-Wang Shu. "Geometric shock-capturing ENO schemes for subpixel interpolation, computation, and curve evolution". In: *Computer Vision, 1995. Proceedings., International Symposium on*. IEEE. 1995, pp. 437–442.

[39] C. Smith and N. Blachman. *The Mathematica graphics guidebook*. Addison-Wesley, 1995. ISBN: 9780201826555. URL: https://books.google.cz/books?id=l%5C_MYAQAAIAAJ.

[40] Jeffrey Allen Tupper. *Graphing equations with generalized interval arithmetic*. University of Toronto, 1996.

[41] Leland Wilkinson. "Algorithms for choosing the domain and range when plotting a function". In: (1991), pp. 1–8.

[42] Daniel C.H. Yang and Tom Kong. "Parametric interpolator versus linear interpolator for precision CNC machining". In: *Computer-Aided Design* 26.3 (1994). Special Issue:NC machining and cutter-path generation, pp. 225–234. ISSN: 0010-4485. DOI: https://doi.org/10.1016/0010-4485(94)90045-0.

[43] S-S Yeh and P-L Hsu. "The speed-controlled interpolator for machining parametric curves". In: *Computer-Aided Design* 31.5 (1999), pp. 349–357.

[44] Syh-Shiuh Yeh and Pau-Lo Hsu. "Adaptive-feedrate interpolation for parametric curves with a confined chord error". In: *Computer-aided design* 34.3 (2002), pp. 229–237.

[45] Tony Chan Zhou and HM Zhou. "Adaptive Eno-Wavelet Transforms For Discontinuous Functions". In: *CAM Report, No. 99-21, Dept. of Math., UCLA, Submit to SIAM Numer. Anal.* Citeseer. 1999.