Two Algorithms for Adaptive Approximation of Bivariate Functions by Piecewise Linear Polynomials on Triangulations

Nira Dyn

School of Mathematical Sciences
Tel Aviv University, Israel

First algorithm — from fine to coarse
Designed for approximating images given by luminances (grey levels) over pixels.

Joint work with L. Demaret, M. Floater and A. Iske.

Second algorithm — from coarse to fine
Aims at approximating bivariate functions given everywhere in a rectangular domain.

Work in progress with A. Cohen and F. Hecht.

Both algorithms are greedy and allow for error control.

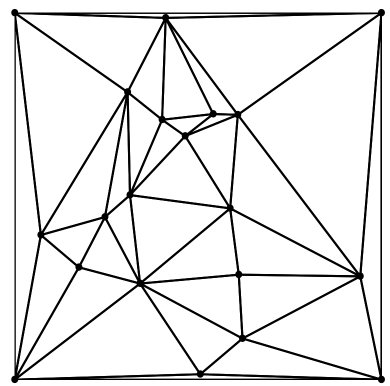
3 Linear Splines over Triangulations

Definition. A triangulation of a planar point set $Y = \{y_1, \ldots, y_N\}$ is a collection $\mathcal{T}(Y) = \{T\}_{T \in \mathcal{T}(Y)}$ of triangles in the plane, such that

(T1) the vertex set of $\mathcal{T}(Y)$ is Y;

(T2) any pair of two distinct triangles in $\mathcal{T}(Y)$ intersect at most at one common vertex or along one common edge;

(T3) the convex hull $[Y]$ of Y coincides with the area covered by the union of the triangles in $\mathcal{T}(Y)$.



A triangulation of a point set.

Approximation Spaces.

- Given any triangulation $\mathcal{T}(Y)$ of Y, we denote by

$$\mathcal{S}_Y = \left\{ s : s \in C([Y]) \text{ and } s\big|_T \text{ linear for all } T \in \mathcal{T}(Y) \right\},$$

  the `spline space` containing all continuous functions over $[Y]$ whose restriction to any triangle in $\mathcal{T}(Y)$ is linear.

- Any element in $\mathcal{S}_Y$ is referred to as a `linear spline` over $\mathcal{T}(Y)$.

- For given function values $\{I(y) : y \in Y\}$, there is a unique linear spline, $L(Y, I) \in \mathcal{S}_Y$, which interpolates I at the points of Y, i.e.,

$$L(Y, I)(y) = I(y), \qquad \text{for all } y \in Y.$$

4 Outline of our Approach

On input image $\mathrm{I} = \{(x, \mathrm{I}(x)) : x \in X\}$,

- determine a *good* adaptive spline space $\mathcal{S}_Y$, where $Y \subset X$;
- determine from $\mathcal{S}_Y$ the unique best approximation $L^*(Y, \mathrm{I}) \in \mathcal{S}_Y$ satisfying

$$\sum_{x \in X} |L^*(Y, \mathrm{I})(x) - \mathrm{I}(x)|^2 = \min_{s \in \mathcal{S}_Y} \sum_{x \in X} |s(x) - \mathrm{I}(x)|^2.$$

- Encode the linear spline $L^* \in \mathcal{S}_Y$;
- Decode $L^* \in \mathcal{S}_Y$, and so obtain the reconstructed image $\tilde{\mathrm{I}} = \{(x, L(Y, \tilde{\mathrm{I}})(x)) : x \in X\}$, where $L(Y, \tilde{\mathrm{I}}) \approx L^*(Y, \mathrm{I})$.

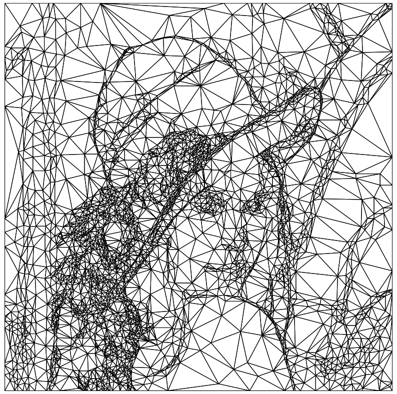
OBS! Key Step: Selection of significant pixels $Y \subset X$.

- This is done by using an "adaptive thinning algorithm".
- For the triangulation in $\mathcal{S}_Y$, we take the *Delaunay triangulation* $\mathcal{D}(Y)$ of Y.

Popular Example: Test Image Lena.

Original Image (512×512).

3244 significant pixels.

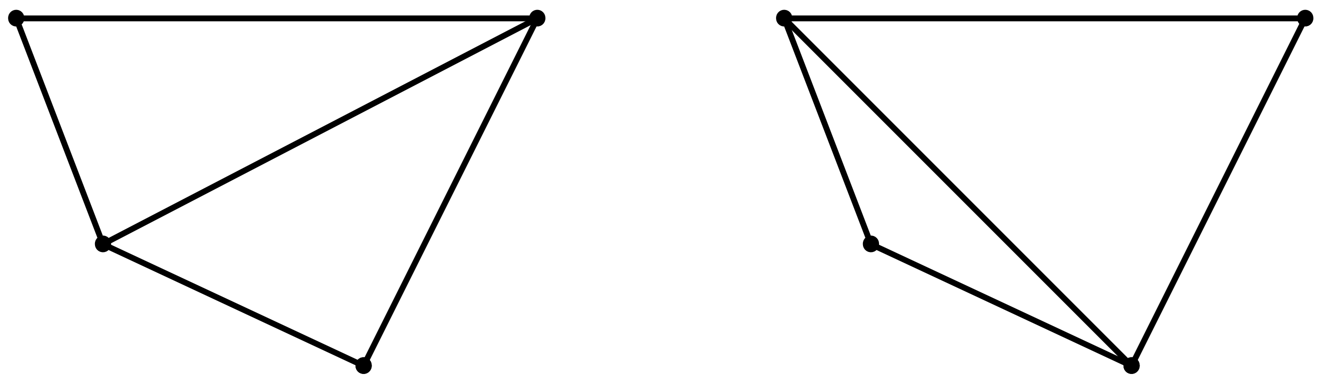


Delaunay Triangulation.

Image Reconstruction.

1 Delaunay Triangulations.

Definition. The `Delaunay triangulation` $\mathcal{D}(X)$ of a discrete planar point set X is a triangulation of X, such that the circumcircle for each of its triangles does not contain any point from X in its interior.



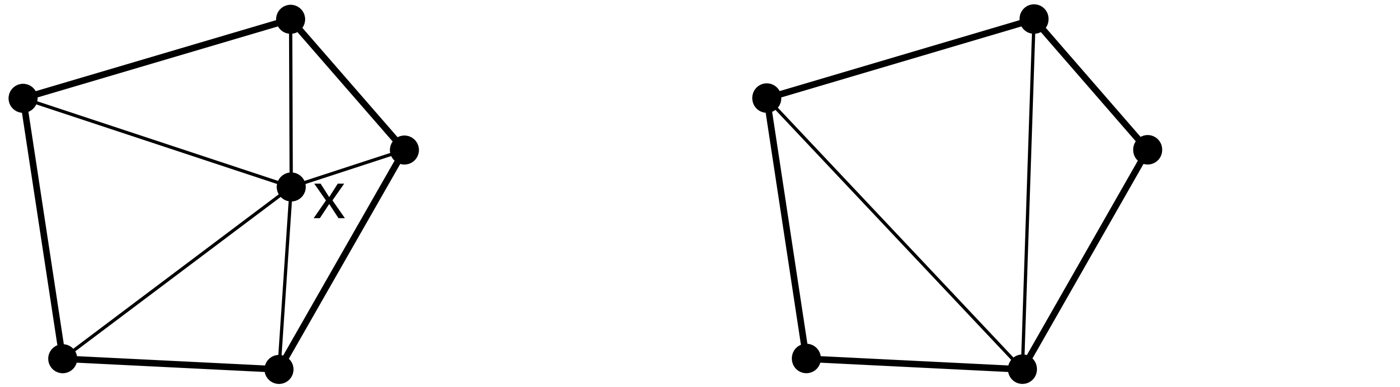
Two triangulations of a convex quadrilateral, $\mathcal{T}$ (left) and $\tilde{\mathcal{T}}$ (right).

Properties of Delaunay Triangulations.

- **Uniqueness.**
  Delaunay triangulation $\mathcal{D}(X)$ is *unique*, if no four points in X are co-circular.

- **Complexity.**
  For any point set X, its Delaunay triangulation $\mathcal{D}(X)$ can be computed in $\mathcal{O}(N \log N)$ steps, where $N = |X|$.

- **Local Updating.**
  For any X and $x \in X$, the Delaunay triangulation $\mathcal{D}(X \setminus x)$ of the point set $X \setminus x$ can be computed from $\mathcal{D}(X)$ by retriangulating the *cell* $\mathcal{C}(x)$ of x.



Removal of the node x, and retriangulation of its cell $\mathcal{C}(x)$.

2 Adaptive Thinning Algorithm

INPUT. $I = \{0, 1, \ldots, 2^r - 1\}^X$, pixels and luminances, where X set of pixels, r number of bits for representation of luminances.

(1) Let $X_N = X$;

(2) **FOR** $k = 1, \ldots, N - n$

(2a) Find a `least significant` pixel $x \in X_{N-k+1}$;

(2b) Let $X_{N-k} = X_{N-k+1} \setminus x$;

- **OUTPUT:** Data hierarchy

$$X_n \subset X_{n+1} \subset \cdots \subset X_{N-1} \subset X_N = X$$

of nested subsets of X.

Controlling the Mean Square Error.

- For a given mean square error (MSE), $\bar{\eta}^*$, the adaptive thinning algorithm can be changed in order to terminate when for the first time, the MSE value corresponding to the current linear spline $L(X_p, I)$ is above $\bar{\eta}^*$, for some X_p in the data hierarchy, $n = p$ a posteriori.

- We take as the final approximation to the image the linear spline $L^*(X_{p+1}, I)$, and so we let $Y = X_{p+1}$.

- Observe that $L^*(X_{p+1}, I)$ satisfies

$$\sum_{x \in X} |L^*(X_{p+1}, I)(x) - I(x)|^2 / |X_{p+1}| \leq \bar{\eta}^*,$$

as desired.

Greedy Two-Point-Removal.

Anticipated Error for the Removal of two Points.

$$e(y_1, y_2) = \eta(Y \setminus \{y_1, y_2\}; X) - \eta(Y; X), \qquad \text{for } y_1, y_2 \in Y.$$

Can be simplified as

$$e(y_1, y_2) = e_\delta(y_1) + e_\delta(y_2), \qquad \text{for } [y_1, y_2] \notin \mathcal{D}(Y), \tag{1}$$

provided that $y_1, y_2 \in Y$ are *not* connected by an edge in $\mathcal{D}(Y)$.

Definition. (Adaptive Thinning Algorithm AT^2).
For $Y \subset X$, a point pair $y_1^*, y_2^* \in Y$ is said to be `least significant` in Y, iff it satisfies

$$e(y_1^*, y_2^*) = \min_{y_1, y_2 \in Y} e(y_1, y_2).$$

Implementation of Algorithm $\mathbf{AT}^2$.

- Due to the representation

$$e_\delta(y_1, y_2) = e_\delta(y_1) + e_\delta(y_2), \qquad \text{for } [y_1, y_2] \notin \mathcal{D}(Y),$$

  the maintenance of the significances $\{e_\delta(y_1, y_2) : \{y_1, y_2\} \subset Y\}$ can be reduced to the maintenance of $\{e_\delta(y_1, y_2) : [y_1, y_2] \in \mathcal{D}(Y)\}$ and $\{e_\delta(y) : y \in Y\}$.

- For the efficient implementation of Algorithm $\mathbf{AT}^2$ we use two different priority queues, one for the significances e_δ of pixels, and one for the significances e_δ of edges in $\mathcal{D}(Y)$.

- Each priority queue has a least significant element (pixel or pixel pair) at its head, and is updated after each pixel removal.

- The resulting algorithm has also complexity $\mathcal{O}(N \log N)$.

Further Computational Details

- We do not remove corner points from X, so that the image domain [X] is invariant during the performance of adaptive thinning.

Uniqueness of Delaunay triangulation.

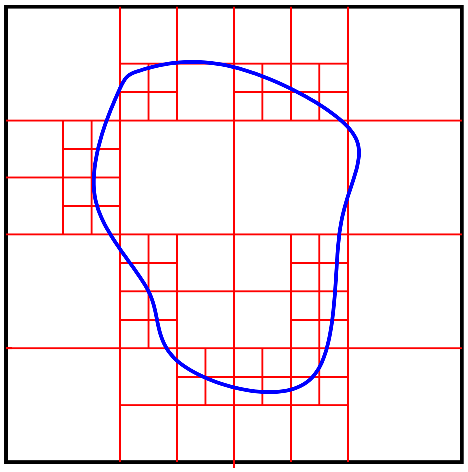
- Recall that the Delaunay triangulation $\mathcal{D}(Y)$ of $Y \subset X$, is unique, provided that no four points in Y are co-circular.
- Since neither X nor its subsets satisfy this condition, we initially perturb the pixel positions in order to guarantee the uniqueness of $\mathcal{D}(Y)$, for any $Y \subset X$.
- The pertubation rules are known at the encoder and at the decoder.

From now, we denote the set of perturbed pixels by X,
and the set of unperturbed pixels (with integer positions) by $\tilde{X}$.

Likewise, any subset $Y \subset X$ corresponds to a subset $\tilde{Y} \subset \tilde{X}$ of unperturbed pixels.

Wavelets and edges

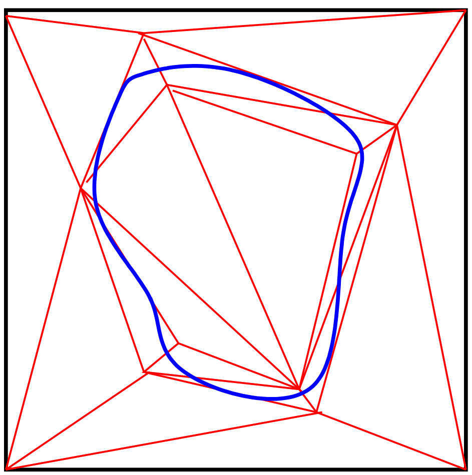
Image: $f = \chi_\Omega$, with $\partial\Omega$ smooth.



f_N = approximation by N largest wavelet coefficients

$\Rightarrow \|f - f_N\|_{L^2} \sim N^{-1/2}$

Problem : imposes isotropic refinement

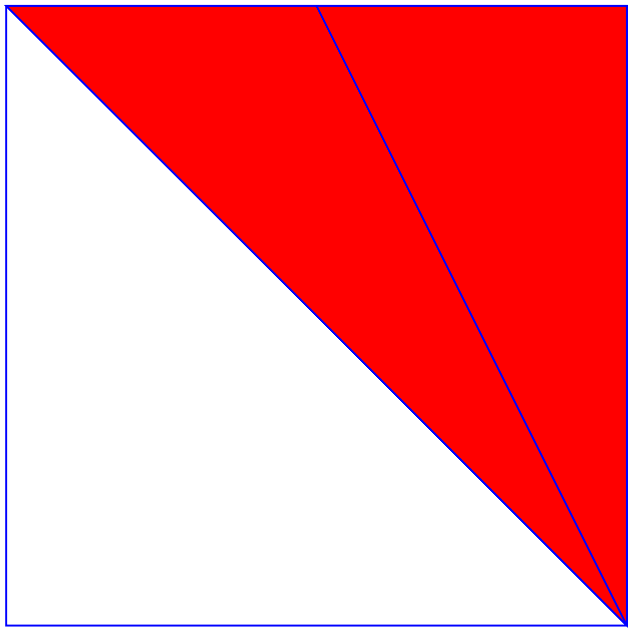


f_N = piecewise linear interpolation on N optimaly selected triangles

$\Rightarrow \|f - f_N\|_{L^2} \sim N^{-1}$

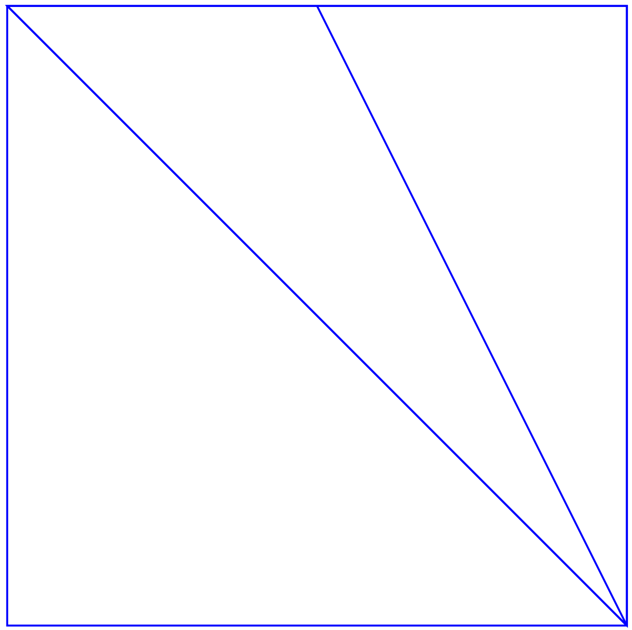
Problem : non-supervised algorithm ?

Greedy approach: adaptive mid-point bisection (Dyn, Hecht, AC)



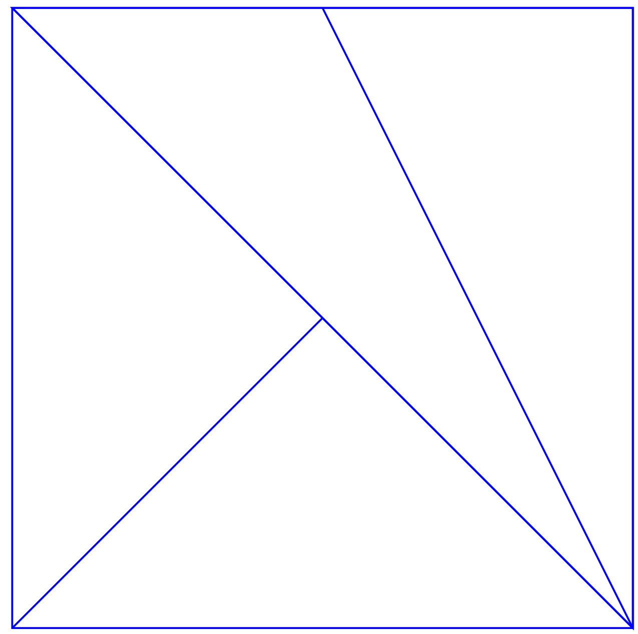
Coarse triangulation ⇒ select triangle with largest local L^2 error ⇒ choose the mid-point bisection that best reduces this error

Greedy approach: adaptive mid-point bisection (Dyn, Hecht, AC)



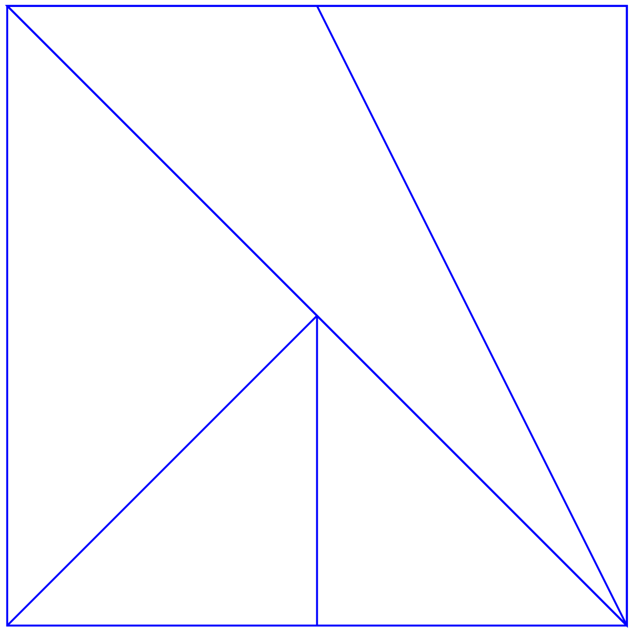
Coarse triangulation $\Rightarrow$ select triangle with largest local L^2 error $\Rightarrow$ choose the mid-point bisection that best reduces this error $\Rightarrow$ split

Greedy approach: adaptive mid-point bisection (Dyn, Hecht, AC)



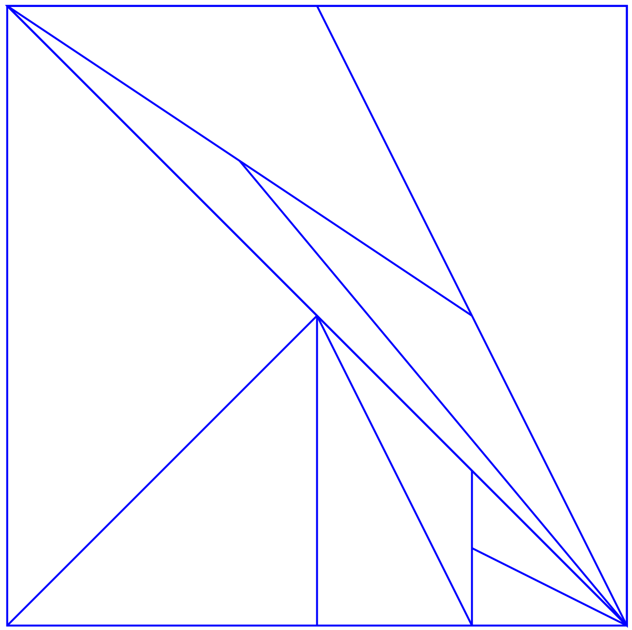
Coarse triangulation $\Rightarrow$ select triangle with largest local L^2 error $\Rightarrow$ choose the mid-point bisection that best reduces this error $\Rightarrow$ split $\Rightarrow$ iterate...

Greedy approach: adaptive mid-point bisection (Dyn, Hecht, AC)



Coarse triangulation $\Rightarrow$ select triangle with largest local L^2 error $\Rightarrow$ choose the mid-point bisection that best reduces this error $\Rightarrow$ split $\Rightarrow$ iterate...

Greedy approach: adaptive mid-point bisection (Dyn, Hecht, AC)

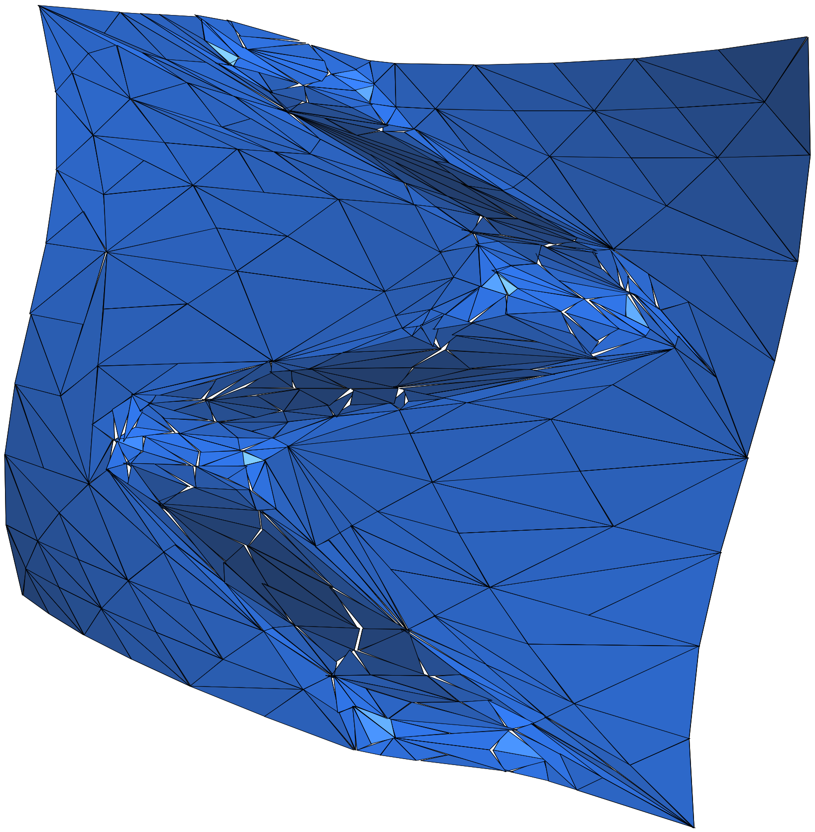


Coarse triangulation $\Rightarrow$ select triangle with largest local L^2 error $\Rightarrow$ choose the mid-point bisection that best reduces this error $\Rightarrow$ split $\Rightarrow$ iterate....

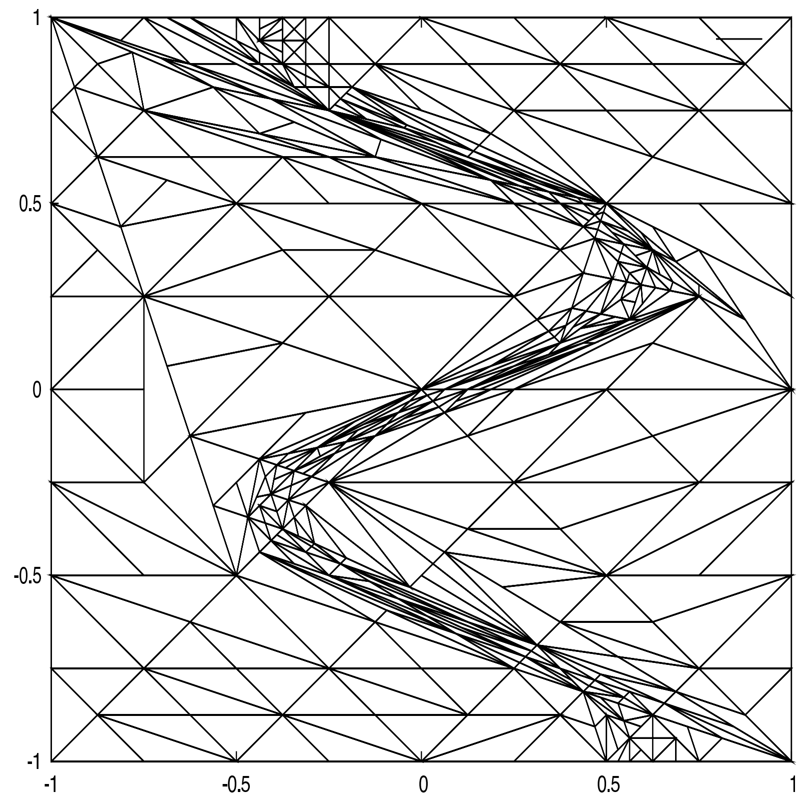
... until prescribed accuracy or number of triangles is met.

Development of anisotropic triangles

Example: sharp gradient transition on a sine curve.



Approximation



Triangulation

Theoretical questions

Algorithm stops when reaching the minimal number of triangles N for which a prescribed L_2 error D is ensured.

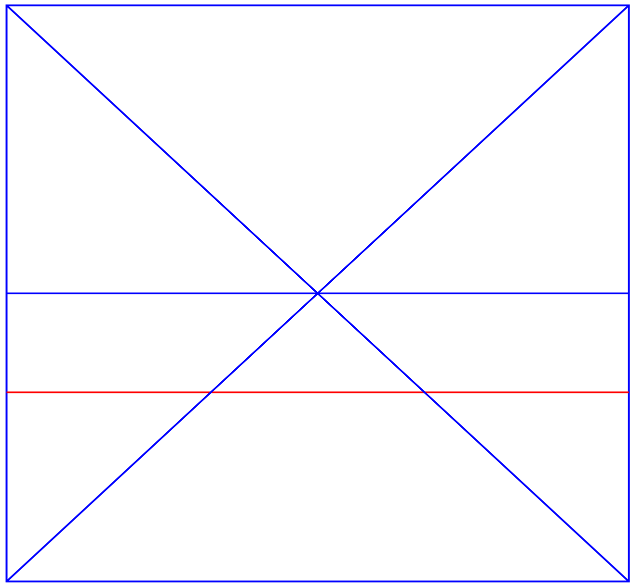
Open problem: do greedy algorithm allow to obtain the optimal convergence rate $D \leq CN^{-1}$ for piecewise smooth functions with smooth edges, such as χ_Ω ?

A more tamed version of this problem: for such functions, is there a certain splitting scenario which generates triangulations such that $D \leq CN^{-1}$? (in other words, is the class of triangulation generated by splitting rich enough to approximate general smooth edges).

Negative answer in the case where the split is limited to the mid-point, positive answer with more choices.

The case of a step function

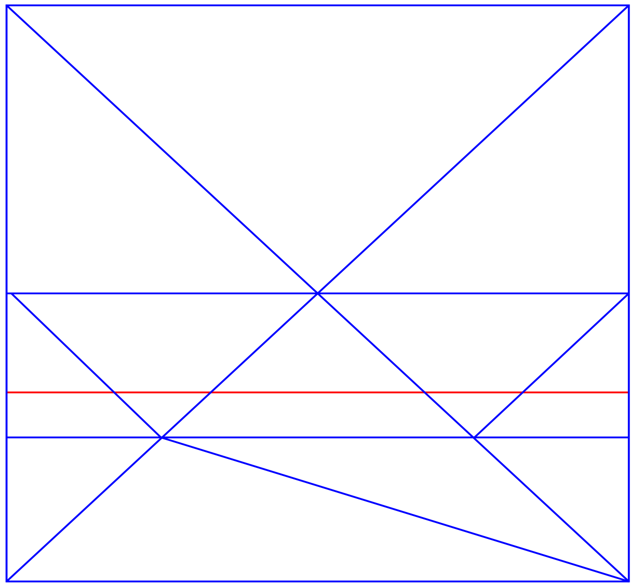
Consider the step function $f(x,y) = \chi_{\{y>1/3\}}$ on $[0,1]^2$.



Supervised split : refine the edge at resolution $\Delta y = 1, \frac{1}{2}$

The case of a step function

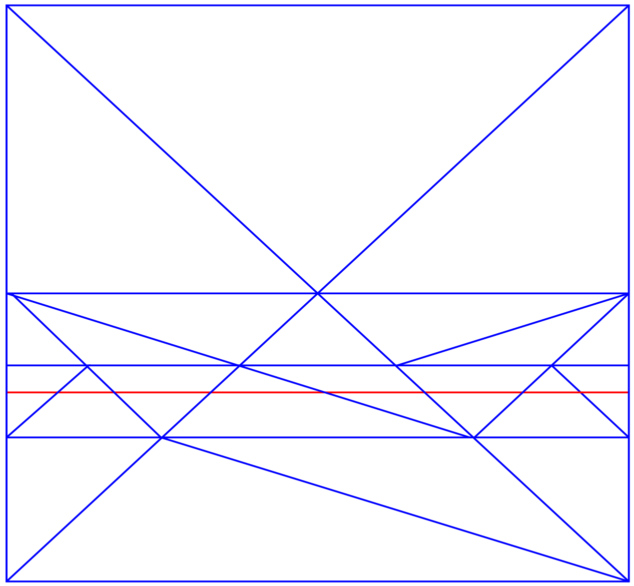
Consider the step function $f(x,y) = \chi_{\{y>1/3\}}$ on $[0,1]^2$.



Supervised split : refine the edge at resolution $\Delta y = 1, \frac{1}{2}, \frac{1}{4}$

The case of a step function

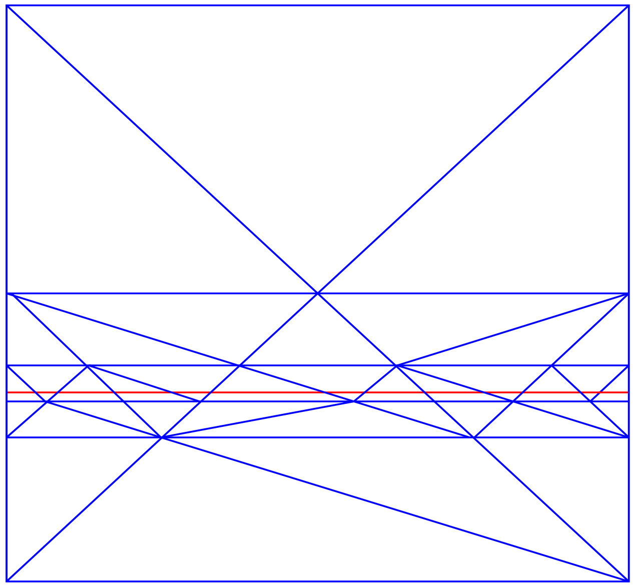
Consider the step function $f(x, y) = \chi_{\{y>1/3\}}$ on $[0, 1]^2$.



Supervised split : refine the edge at resolution $\Delta y = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$

The case of a step function

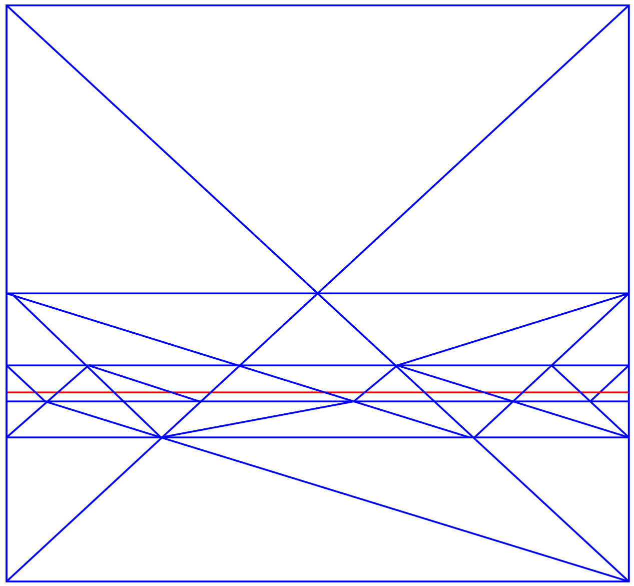
Consider the step function $f(x, y) = \chi_{\{y>1/3\}}$ on $[0, 1]^2$.



Supervised split : refine the edge at resolution $\Delta y = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16} \ldots$

The case of a step function

Consider the step function $f(x,y) = \chi_{\{y>1/3\}}$ on $[0,1]^2$.



Supervised split : refine the edge at resolution $\Delta y = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16} \cdots$

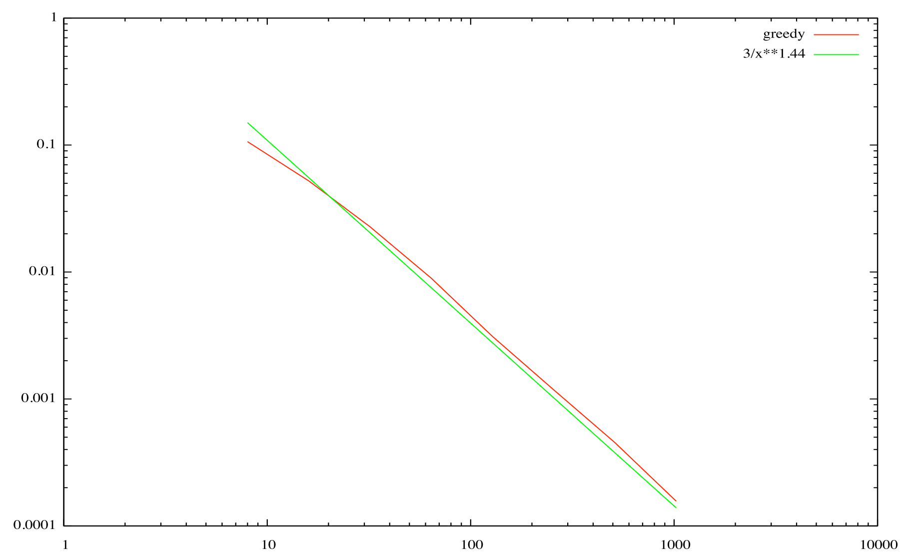
Number of triangles $N = N(j)$ generated at resolution 2^{-j} :

$$N(j) = N(j-1) + N(j-2) \sim \mathrm{G}^j$$

Convergence rate

Supervised split : at resolution 2^{-j}, the L^2 square error is at best controlled by

$$E \leq 2^{-j} \leq CN^{-r}, \quad r = \frac{\log 2}{\log G} \approx 1.44$$



Greedy split exhibits the same convergence rate. General result ?

Comparision between the two algorithms

- **Triangulations**:
  First – Delaunay triangulation of a set of significant points (pixels).
  Second – A triangulation with a binary tree structure.

- **Approximating spaces**:
  First – Continuous piecewise linear polynomials over the triangulation.
  Second – Discontinuous piecewise linear polynomials over the triangulation.
  Nested approximating spaces during the performance of the algorithm.

- **Computation of the optimal approximant**:
  First – A global minimization over the values attached to the significant points (pixels).

  Second – Local computation on each triangle; local correction after a split.

Conclusion:
The approximant obtained by the second algorithm can be encoded more efficiently.