Dessin rapide à haute résolution de courbes et surfaces algébriques

Nuwan Herath Mudiyanselage
avec Guillaume Moroz et Marc Pouget

Guillaume Moroz

Marc Pouget

Inria Nancy-Grand Est / Loria
GAMBLE

- géométrie algorithmique
- approche probabiliste
- combinatoire
- calcul formel
- géométrie hyperbolique

Guillaume Moroz

Marc Pouget

Inria Nancy-Grand Est / Loria
GAMBLE

- **géométrie algorithmique**
- approche probabiliste
- combinatoire
- **calcul formel**
- géométrie hyperbolique

Vue d'ensemble

Dessin de courbes implicites

Problème du dessin de courbe implicite

Problème général

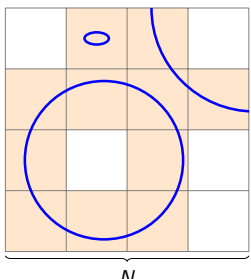
Représentation discrète d'une courbe implicite sur une grille fixe

- **Entrée**:
  - fonction F
  - résolution N
  - fenêtre de visualisation

  Courbe implicite définie comme l'ensemble solution

$$\{(x, y) \in \mathbb{R}^2 \mid F(x, y) = 0\}$$

- **Sortie**: dessin (ensemble de pixels)



Problème du dessin de courbe implicite

Notre objectif

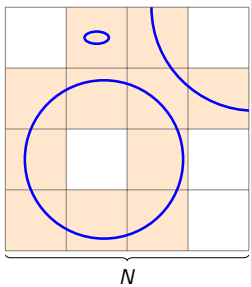
Représentation discrète d'une **courbe algébrique** sur une grille fixe

- **Entrée**:
  - **polynôme bivarié** P de **degré partiel** d
  - résolution N
  - **fenêtre** $[-1, 1] \times [-1, 1]$

  Courbe algébrique définie comme l'ensemble solution

$$\{(x, y) \in \mathbb{R}^2 \mid P(x, y) = 0\}$$

- **Sortie**: dessin (ensemble de pixels)



Objectif : dessin rapide à haute résolution de courbes algébriques de haut degré

- $d \approx 100 \qquad \longrightarrow d^2 \approx 10\ 000$ monômes
- $N \approx 1\ 000$

Pourquoi des courbes algébriques de haut degré ?

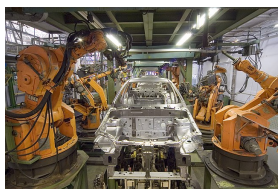
Objectif de la visualisation : avoir une meilleure intuition et une meilleure compréhension de données

En robotique, l'espace de configuration peut être de grande dimension

$$\mathbb{R}^N \to \mathbb{R}^M$$

Opérations sur des variétés algébriques :

- coupe
- projection

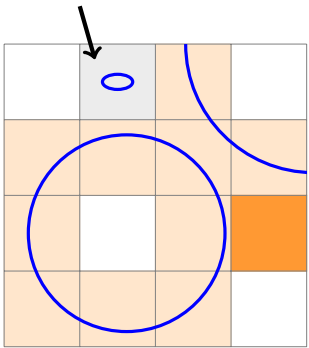


Robots industriels de KUKA par Mixabest
(CC BY-SA 3.0)

Correction du dessin

Pour des raisons numériques, il peut y avoir :

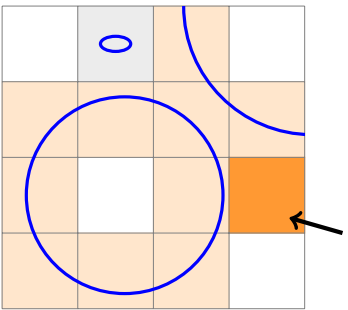
- des pixels **faux négatifs**



Correction du dessin

Pour des raisons numériques, il peut y avoir :

- des pixels **faux négatifs**
- des pixels **faux positifs**

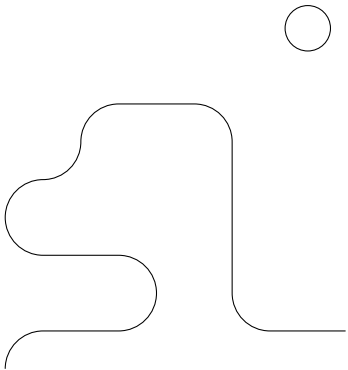


État de l'art

Marching squares

L'idée

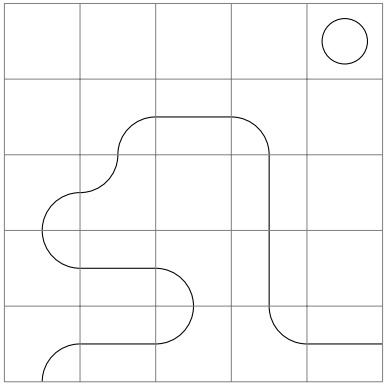
Variante 2D du très utilisé algorithme de marching cubes [Lorensen & Cline, 1987]
Courbe implicite définie par $P(X, Y) = 0$



Marching squares

L'idée

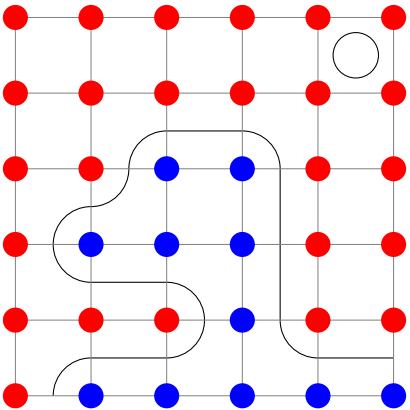
Variante 2D du très utilisé algorithme de marching cubes [Lorensen & Cline, 1987]
Courbe implicite définie par $P(X, Y) = 0$



Marching squares

L'idée

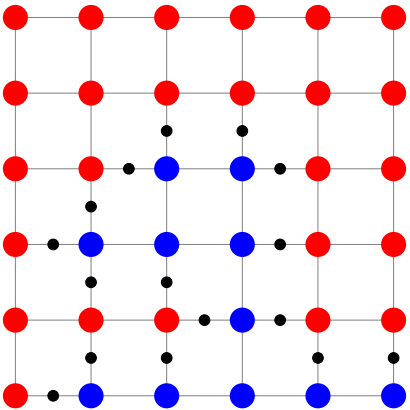
Variante 2D du très utilisé algorithme de marching cubes [Lorensen & Cline, 1987]
Courbe implicite définie par $P(X, Y) = 0$



Marching squares

L'idée

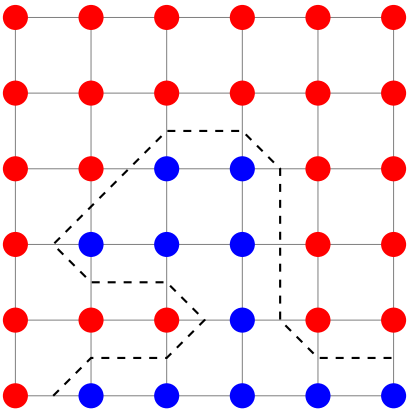
Variante 2D du très utilisé algorithme de marching cubes [Lorensen & Cline, 1987]
Courbe implicite définie par $P(X, Y) = 0$



Marching squares

L'idée

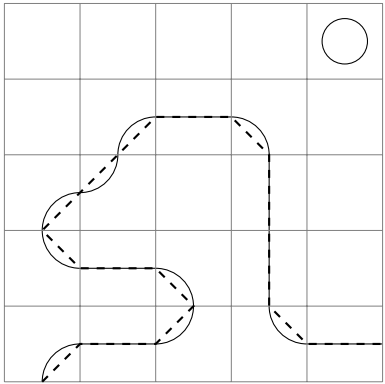
Variante 2D du très utilisé algorithme de marching cubes [Lorensen & Cline, 1987]
Courbe implicite définie par $P(X, Y) = 0$



Marching squares

L'idée

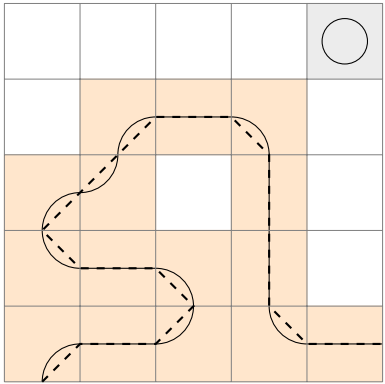
Variante 2D du très utilisé algorithme de marching cubes [Lorensen & Cline, 1987]
Courbe implicite définie par $P(X, Y) = 0$



Marching squares

L'idée

Variante 2D du très utilisé algorithme de marching cubes [Lorensen & Cline, 1987]
Courbe implicite définie par $P(X, Y) = 0$



Marching squares

Complexité

Complexité (nombre d'opérations élémentaires)
Évaluation naïve

$$\theta(d^2 N^2)$$

d degré partiel
N résolution de la grille

Complexité arithmétique de marching squares

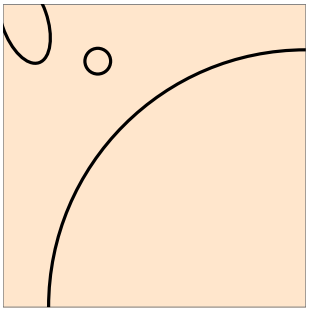
Avec évaluation partielle de $P(x, y)$, en supposant $d < N$

$$\theta(dN^2)$$

Lent pour de hautes résolutions...
Peut-on avoir un algorithme en $O(dN)$?

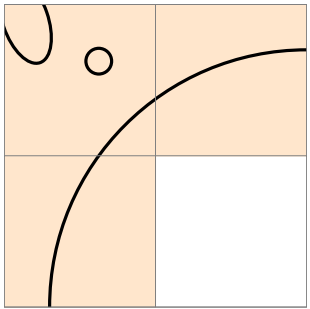
Subdivision adaptative

Raffinement local de la grille



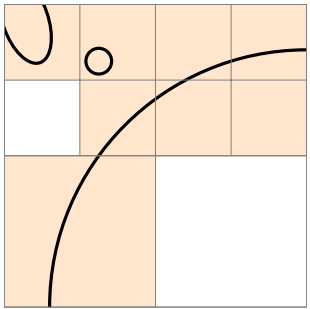
Subdivision adaptative

Raffinement local de la grille



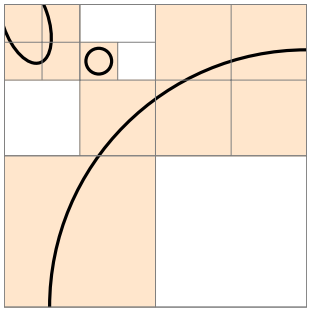
Subdivision adaptative

Raffinement local de la grille



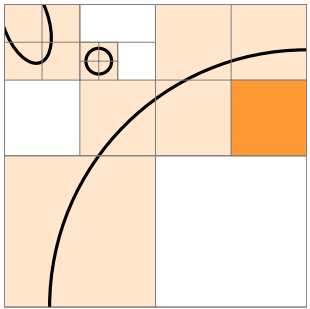
Subdivision adaptative

Raffinement local de la grille



Subdivision adaptative

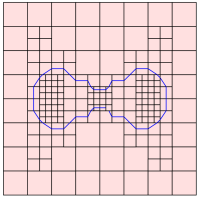
Raffinement local de la grille



Méthodes fournissant une correction topologique

Subdivision 2D adaptative avec arithmétique d'intervalles

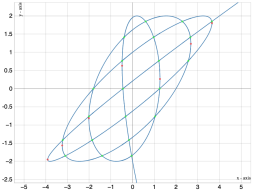
- [Snyder, 1992]
- [Plantinga & Vegter, 2004]
- [Burr et al., 2008]
- [Lin & Yap, 2011]
- ...



[Lin & Yap, 2011]

Décomposition algébrique cylindrique (CAD)

- [Gonzalez-Vega & Necula, 2002]
- [Eigenwillig et al., 2007]
- [Alberti et al., 2008]
- [Cheng et al., 2009]
- [Kobel & Sagraloff, 2015]
- [Diatta et al., 2018]
- ...



https://isotop.gamble.loria.fr/

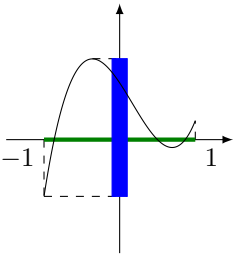
Notre approche

Arithmétique d'intervalles

Propriété d'inclusion

$$P(X) = 2X^3 - X^2 - 1.5X + 0.75$$

How to compute $P(I)$ for $I = [-1, 1]$?



x	-1		$x_1 = \frac{1-\sqrt{10}}{6}$		$x_2 = \frac{1+\sqrt{10}}{6}$		1
$P'(x)$		$+$	0	$-$	0	$+$	
$P(x)$	$P(-1)$	↗	$P(x_1)$	↘	$P(x_2)$	↗	$P(1)$

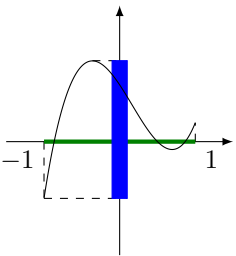
$$P(I) = [-0.75, 1.06\ldots]$$

Arithmétique d'intervalles

Propriété d'inclusion

$$P(X) = 2X^3 - X^2 - 1.5X + 0.75$$

How to compute $P(I)$ for $I = [-1, 1]$?



$$\square P(I) = 2[-1,1]^3 - [-1,1]^2 - 1.5[-1,1] + 0.75$$
$$= [-5.25, 5.25]$$

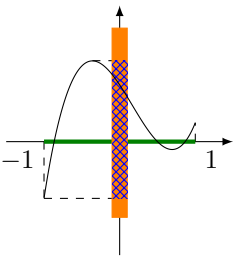
$$P(I) = [-0.75, 1.06\ldots]$$

Arithmétique d'intervalles

Propriété d'inclusion

$$P(X) = 2X^3 - X^2 - 1.5X + 0.75$$

How to compute $P(I)$ for $I = [-1, 1]$?



$$\begin{aligned}\square P(I) &= 2[-1,1]^3 - [-1,1]^2 - 1.5[-1,1] + 0.75 \\ &= [-5.25, 5.25]\end{aligned}$$

Avec la méthode d'Horner :

$$\begin{aligned}\square P(I) &= ((2[-1,1] - 1)[-1,1] - 1.5)[-1,1] + 0.75 \\ &= [-3.75, 5.25]\end{aligned}$$

$$P(I) \subseteq \square P(I)$$

$$P(I) = [-0.75, 1.06\ldots]$$

Arithmétique d'intervalles

Propriété de convergence

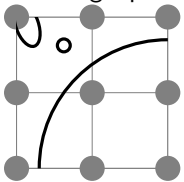
Convergence en un point

With $x \in [a, b]$

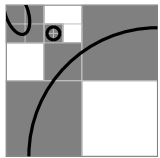
$$\lim_{[a,b] \longrightarrow [x,x]=\{x\}} \Box P([a, b]) = P(x)$$

Notre approche : intersections garanties avec la grille

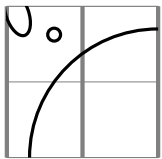
Marching squares



Subdivision adaptative



Nouvelle approche: évaluation le long de fibres



$\Rightarrow$ La rendre rapide et fournir des garanties

Deux algorithmes

Dessin d'arêtes

- *évaluation en X*
  nœuds de Tchebychev
  évaluation multipoint avec IDCT
- *subdivision en Y*
  méthode naïve d'isolation de racines

Garanties
Pixels faux positifs et faux négatifs

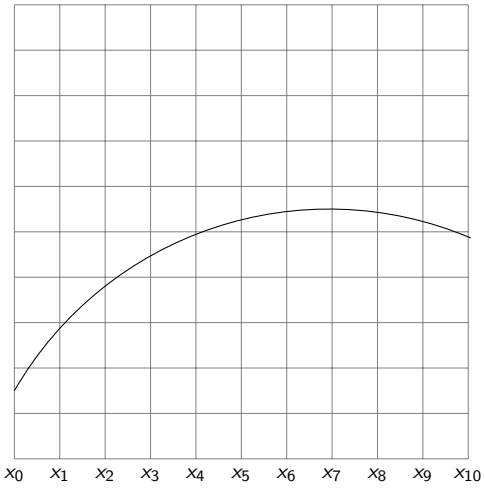
Dessin de pixels

- *évaluation en X*
  nœuds de Tchebychev
  évaluation multipoint avec IDCT
  approximation de Taylor
- *subdivision in Y*
  méthode naïve d'isolation de racines

Garanties
Pixels faux positifs *seulement*

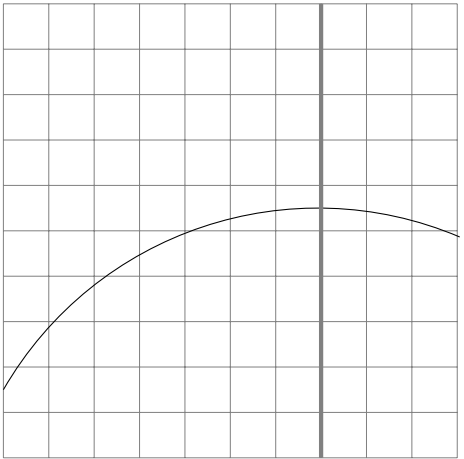
Subdivisions le long d'une fibre

$P(x_k, Y) = \sum a_j Y^j$



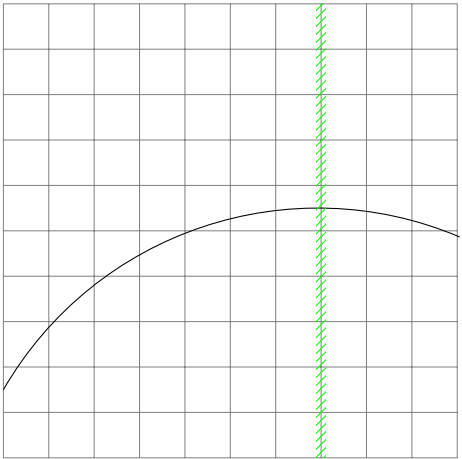
Subdivisions le long d'une fibre

$P(x_k, Y) = \sum a_j Y^j$



Subdivisions le long d'une fibre

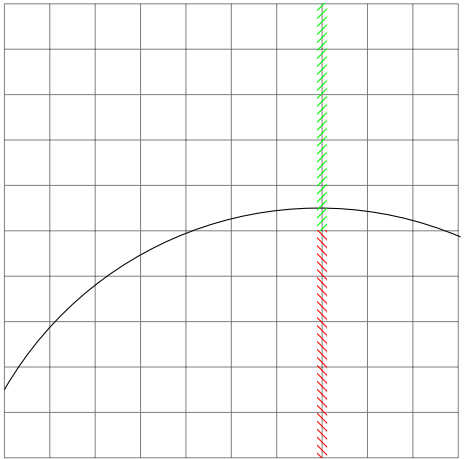
$P(x_k, Y) = \sum a_j Y^j$



✓

Subdivisions le long d'une fibre

$P(x_k, Y) = \sum a_j Y^j$

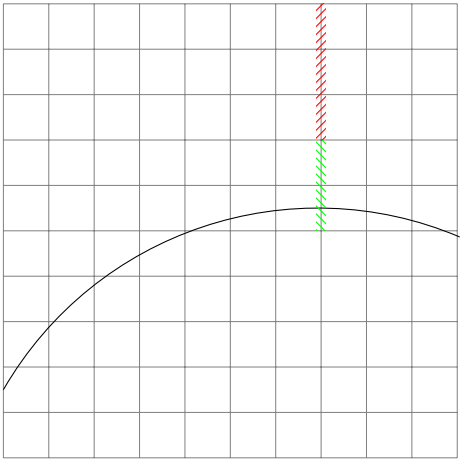


✓

✗

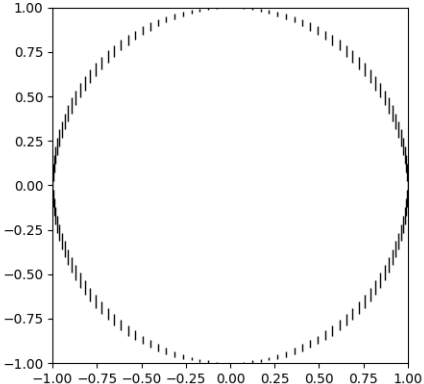
Subdivisions le long d'une fibre

$P(x_k, Y) = \sum a_j Y^j$



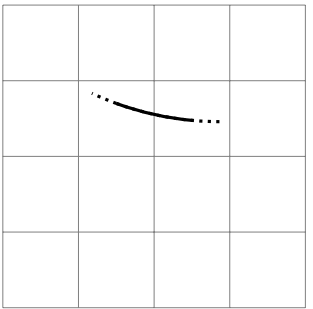
Un exemple

$$X^2 + Y^2 - 1 = 0$$



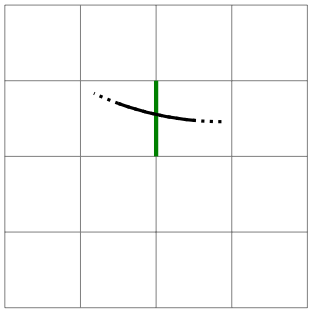
Résolution $N = 64$

Allumage de pixels
Dessin d'arêtes



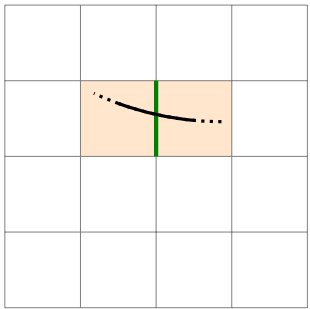
Allumage de pixels
Dessin d'arêtes

- Détecter la traversée entre deux nœuds consécutifs de la grille



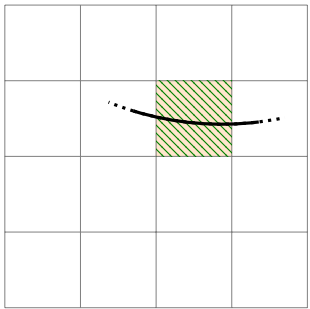
Allumage de pixels
Dessin d'arêtes

- Détecter la traversée entre deux nœuds consécutifs de la grille
- Allumer les pixels adjacents



Allumage de pixels
Dessin de pixels

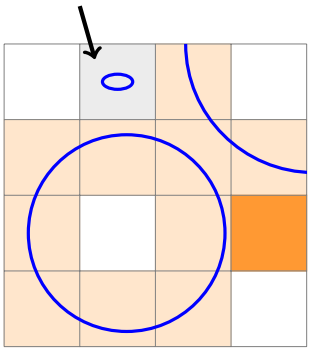
- Détecter la traversée dans un pixel de la grille
- Allumer ce pixel



Pixels faux positifs et faux négatifs
Dessin d'arêtes

Des pixels incorrects :

- **Faux négatif** quand une composante connexe se trouve dans un pixel



Pixels faux positifs et faux négatifs
Dessin d'arêtes

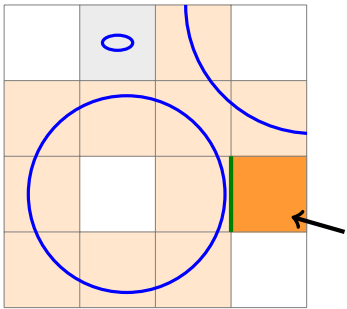
Des pixels incorrects :

- **Faux négatif** quand une composante connexe se trouve dans un pixel
- **Faux positif** quand l'évaluation sur le bord d'un pixel est proche de zéro
  Cela se produit pour un segment S quand

$$0 \in \square P(S) + [-E, E]$$

Certification des segments qui ne sont pas traversés :

$$0 \notin \square P(S) + [-E, E]$$
$$\Downarrow$$
$$0 \notin P(S)$$



Pixels faux positifs et faux négatifs
Dessin de pixels

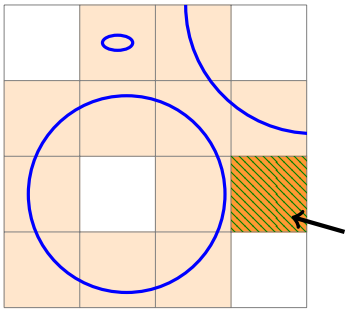
Des pixels incorrects :

- ~~**Faux négatif** quand une composante connexe se trouve dans un pixel~~
- **Faux positif** quand l'évaluation sur le bord d'un pixel est proche de zéro
  Cela se produit pour un segment S quand

$$0 \in \square P(S) + [-E, E]$$

Certification des segments qui ne sont pas traversés :

$$0 \notin \square P(S) + [-E, E]$$
$$\Downarrow$$
$$0 \notin P(S)$$



Évaluation multipoint rapide

Un prérequis pour l'évaluation multipoint rapide

Polynômes de Tchebychev

Definition

Les polynômes de Tchebychev (T_k) vérifient $\forall k \in \mathbb{N}, T_k(\cos\theta) = \cos(k\theta)$

Les trois premiers polynômes de Tchebychev

$$\cos(0 \cdot \theta) = 1 \qquad T_0 = 1$$
$$\cos(1 \cdot \theta) = \cos(\theta) \qquad T_1 = X$$
$$\cos(2 \cdot \theta) = 2\cos(\theta)^2 - 1 \qquad T_2 = 2X^2 - 1$$

Un prérequis pour l'évaluation multipoint rapide

Polynômes de Tchebychev

Definition

Les polynômes de Tchebychev (T_k) vérifient $\forall k \in \mathbb{N}, T_k(\cos\theta) = \cos(k\theta)$

Lemma

Un polynôme arbitraire p de degré d peut être écrit en fonction des polynômes de Tchebychev :

$$p(X) = \sum_{k=0}^{d} \alpha_k T_k(X)$$

Un prérequis pour l'évaluation multipoint rapide

Polynômes de Tchebychev

Definition

Les polynômes de Tchebychev (T_k) vérifient $\forall k \in \mathbb{N}$, $T_k(\cos\theta) = \cos(k\theta)$

Lemma

Un polynôme arbitraire p de degré d peut être écrit en fonction des polynômes de Tchebychev :

$$p(X) = \sum_{k=0}^{d} \alpha_k T_k(X)$$

Lemma

*Pour $N \in \mathbb{N}$, un polynôme p de degré d peut être évalué aux nœuds de Tchebychev $(c_n)_{0 \leq n \leq N-1}$ en utilisant l'*IDCT*:*

$$(p(c_n))_{0 \leq n \leq N-1} = \frac{1}{2}(\alpha_0, \ldots, \alpha_0) + \mathrm{IDCT}((\alpha_k)_{0 \leq k \leq N-1})$$

Un prérequis pour l'évaluation multipoint rapide

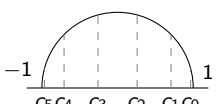
Nœuds de Tchebychev

Definition

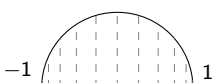
Pour $N \in \mathbb{N}$, les nœuds de Tchebychev sont

$$c_n = \cos\left(\frac{2n+1}{2N}\pi\right), \ n = 0, \ldots, N-1$$

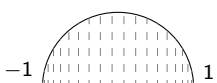
Ce sont les racines de T_N



$N = 6$



$N = 11$

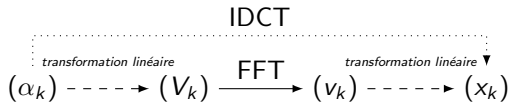


$N = 20$

Transformée en Cosinus Discrète Inverse

Transformée en Cosinus Discrète Inverse (IDCT): $\alpha_k \rightarrow x_n$

$$x_n = \frac{1}{2}\alpha_0 + \sum_{k=1}^{N-1} \alpha_k \cos\left[\frac{\pi k(2n+1)}{2N}\right]$$



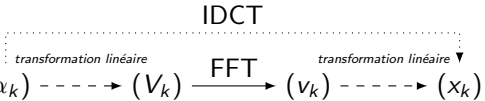
$\Rightarrow$ Rapide grâce à l'algorithme de Transformée de Fourier Rapide (FFT) en $O(N \log_2 N)$

[Makhoul, 1980]

Transformée en Cosinus Discrète Inverse

Transformée en Cosinus Discrète Inverse (IDCT): $\alpha_k \rightarrow x_n$

$$x_n = \frac{1}{2}\alpha_0 + \sum_{k=1}^{N-1} \alpha_k \cos\left[\frac{\pi k(2n+1)}{2N}\right]$$



$\Rightarrow$ Rapide grâce à l'algorithme de Transformée de Fourier Rapide (FFT) en $O(N \log_2 N)$

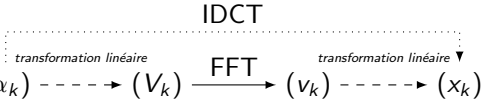
[Makhoul, 1980]

$$p(c_n) = \sum_{k=0}^{N-1} \alpha_k T_k \left(\cos\left(\frac{2n+1}{2N}\pi\right)\right)$$

Transformée en Cosinus Discrète Inverse

Transformée en Cosinus Discrète Inverse (IDCT): $\alpha_k \rightarrow x_n$

$$x_n = \frac{1}{2}\alpha_0 + \sum_{k=1}^{N-1} \alpha_k \cos\left[\frac{\pi k(2n+1)}{2N}\right]$$



$\Rightarrow$ Rapide grâce à l'algorithme de Transformée de Fourier Rapide (FFT) en $O(N \log_2 N)$

[Makhoul, 1980]

$$p(c_n) = \sum_{k=0}^{N-1} \alpha_k T_k \left(\cos\left(\frac{2n+1}{2N}\pi\right)\right) = \sum_{k=0}^{N-1} \alpha_k \cos\left[\frac{\pi k(2n+1)}{2N}\right]$$

Transformée en Cosinus Discrète Inverse

Transformée en Cosinus Discrète Inverse (IDCT): $\alpha_k \rightarrow x_n$

$$x_n = \frac{1}{2}\alpha_0 + \sum_{k=1}^{N-1} \alpha_k \cos\left[\frac{\pi k(2n+1)}{2N}\right]$$

$\Rightarrow$ Rapide grâce à l'algorithme de Transformée de Fourier Rapide (FFT) en $O(N \log_2 N)$

[Makhoul, 1980]

$$p(c_n) = \frac{1}{2}\alpha_0 + \frac{1}{2}\alpha_0 + \sum_{k=1}^{N-1} \alpha_k \cos\left[\frac{\pi k(2n+1)}{2N}\right]$$

$$(p(c_n))_{0 \leq n \leq N-1} = \frac{1}{2}(\alpha_0, \ldots, \alpha_0) + \mathrm{IDCT}((\alpha_k)_{0 \leq k \leq N-1})$$

Erreur de l'IDCT

[Makhoul, 1980] et [Brisebarre et al., 2020, Theorem 3.4] conduisent à

Theorem (H., Moroz, Pouget, 2022)

Assume radix-2, precision-p arithmetic, with rounding unit $u = 2^{-p}$. Let $\widehat{x}$ be the computed 2^n-point IDCT of $\alpha \in \mathbb{C}^{2^n}$, and let x be the exact value. Then

$$\|\widehat{x} - x\|_\infty = n\|\alpha\|_\infty O(u).$$

Table: IDCT error bounds for $p = 53$ (double precision)

$N = 2^n$	$1,024$	$2,048$	$4,096$	$8,192$	$16,384$	$32,768$
$\|\widehat{x} - x\|_\infty / \|\alpha\|_\infty$	7.97e-15	8.84e-15	9.72e-15	1.06e-14	1.15e-14	1.23e-14

Algorithmes

Idée générale : arêtes englobantes

Illustration

$$P(X,Y) = \sum \left(\sum a_{i,j} X^i \right) Y^j = \sum p_j(X) Y^j$$

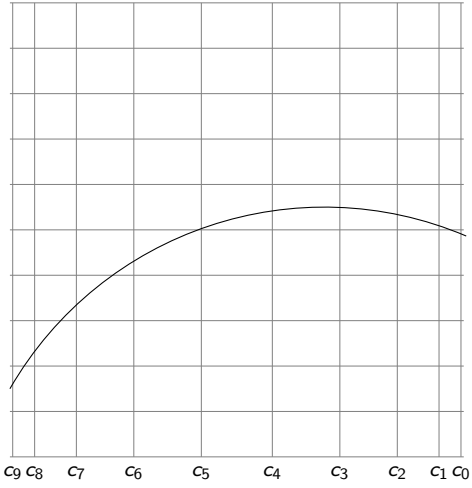
$$p_j(X) = \sum a_{i,j} X^i = \sum \alpha_{i,j} T_i(X)$$

$$(p_j(c_n))_{0 \leq n \leq N-1} = \frac{1}{2}(\alpha_{0,j}, \ldots, \alpha_{0,j}) + \mathrm{IDCT}((\alpha_{k,j})_{0 \leq k \leq N-1})$$

Idée générale : arêtes englobantes

Illustration

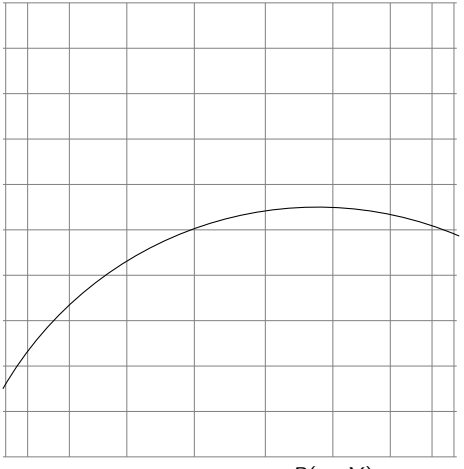
$P(c_n, Y) = \sum p_j(c_n) Y^j$



Idée générale : arêtes englobantes

Illustration

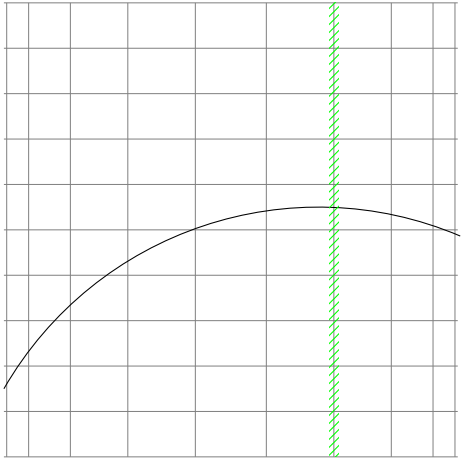
$P(c_3, Y) = \sum p_j(c_3) Y^j$



Idée générale : arêtes englobantes

Illustration

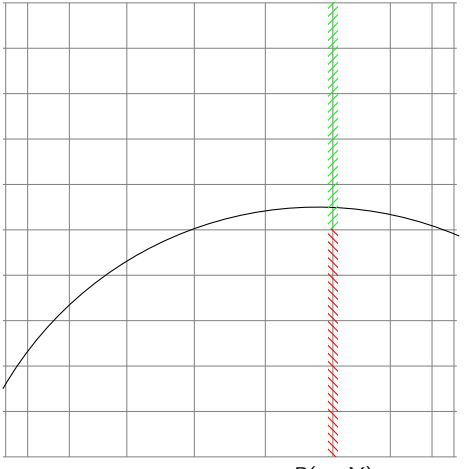
$P(c_3, Y) = \sum p_j(c_3) Y^j$



Idée générale : arêtes englobantes

Illustration

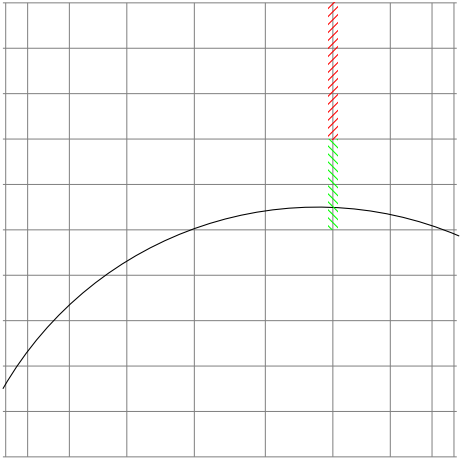
$P(c_3, Y) = \sum p_j(c_3) Y^j$



Idée générale : arêtes englobantes

Illustration

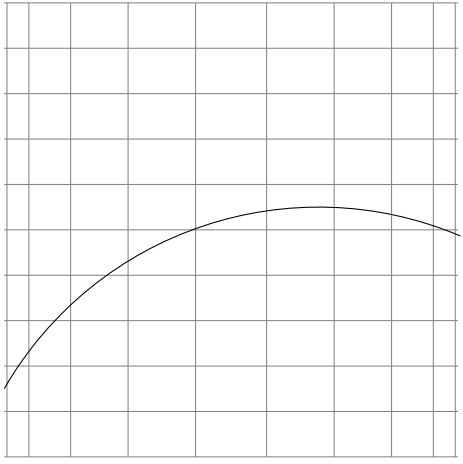
$P(c_3, Y) = \sum p_j(c_3) Y^j$



Idée générale : pixels englobants

Illustration

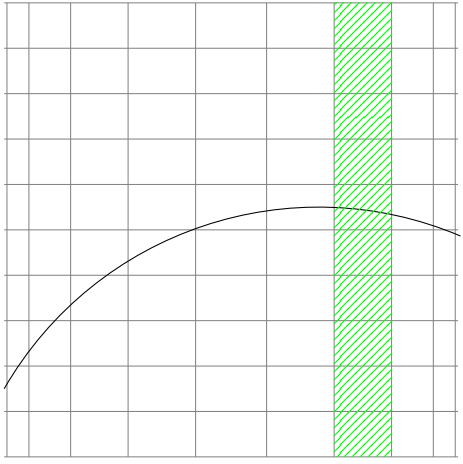
$P(I, Y) = \sum p_j(I) Y^j$



Idée générale : pixels englobants

Illustration

$P(I, Y) = \sum p_j(I) Y^j$

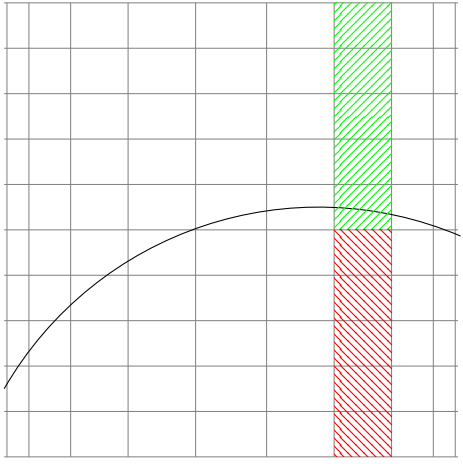


✓

Idée générale : pixels englobants

Illustration

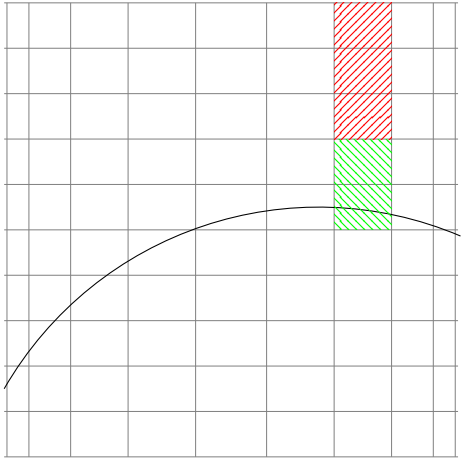
$P(I, Y) = \sum p_j(I) Y^j$



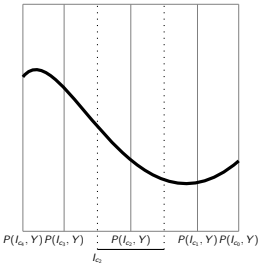
Idée générale : pixels englobants

Illustration

$P(I, Y) = \sum p_j(I) Y^j$



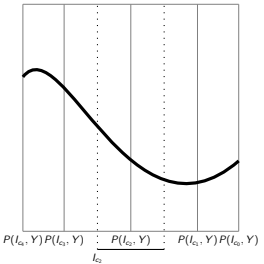
Un algorithme de pixels englobants



évaluation multipoint avec IDCT en X
autour de $c_0, c_1 \ldots$

subdivision en Y

Un algorithme de pixels englobants



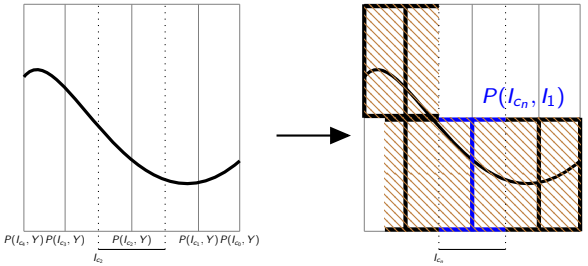
évaluation multipoint avec IDCT + approximation de Taylor en X

subdivision in Y

Développement de Taylor des polynômes partiels de $P(X, Y) = \sum p_j(X) Y^j$

$$\left| p(c_n + r) - \left(p(c_n) + rp'(c_n) + \cdots + \frac{r^m}{m!} p^{(m)}(c_n) \right) \right| \leq \max_{I_{c_n}} \left| p^{(m+1)} \right| \frac{|r|^{(m+1)}}{(m+1)!}$$

Un algorithme de pixels englobants



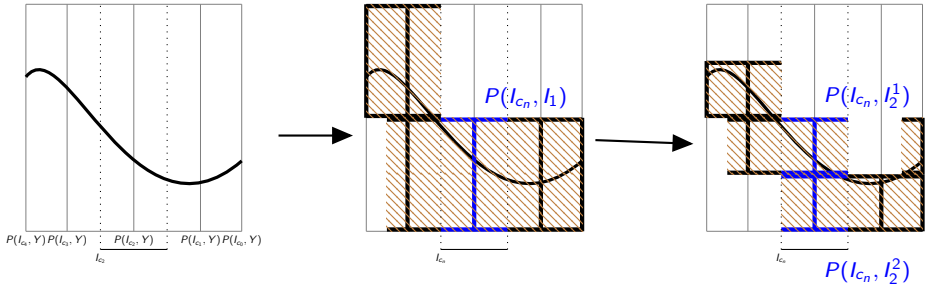
évaluation multipoint avec IDCT + approximation de Taylor en X

subdivision in Y

Développement de Taylor des polynômes partiels de $P(X, Y) = \sum p_j(X) Y^j$

$$\left| p(c_n + r) - \left(p(c_n) + rp'(c_n) + \cdots + \frac{r^m}{m!} p^{(m)}(c_n) \right) \right| \leq \max_{I_{c_n}} \left| p^{(m+1)} \right| \frac{|r|^{(m+1)}}{(m+1)!}$$

Un algorithme de pixels englobants



évaluation multipoint avec IDCT + approximation de Taylor en X

subdivision in Y

Développement de Taylor des polynômes partiels de $P(X, Y) = \sum p_j(X) Y^j$

$$\left| p(c_n + r) - \left(p(c_n) + rp'(c_n) + \cdots + \frac{r^m}{m!} p^{(m)}(c_n) \right) \right| \leq \max_{I_{c_n}} \left| p^{(m+1)} \right| \frac{|r|^{(m+1)}}{(m+1)!}$$

Complexité

Complexité arithmétique

approximation de Taylor multipoint et subdivision $O(\ md^3 + mdN\log_2(N) + dNT)$

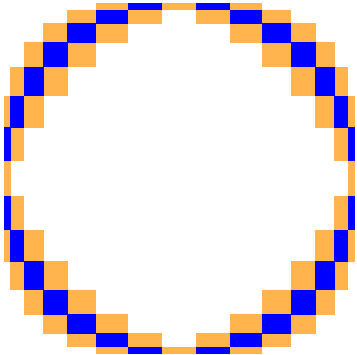
d degré partiel
N résolution
T nombre maximal de nœuds des arbres de subdivision sur toutes les bandes verticales

Avec un nombre constant de branches dans la fenêtre, on attend $T = O(\log_2(N))$

Expériences

Classification des pixels

- traversé : bleu
- non traversé : blanc
- indécidable : jaune

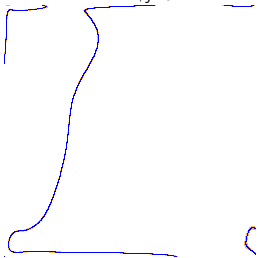


Dessin pour deux familles de polynômes

Expériences sur des courbes lisses $\longrightarrow$ polynômes aléatoires
$\xi_{i,j}$: coefficients aléatoires dans $[-100, 100]$

Polynôme Kac

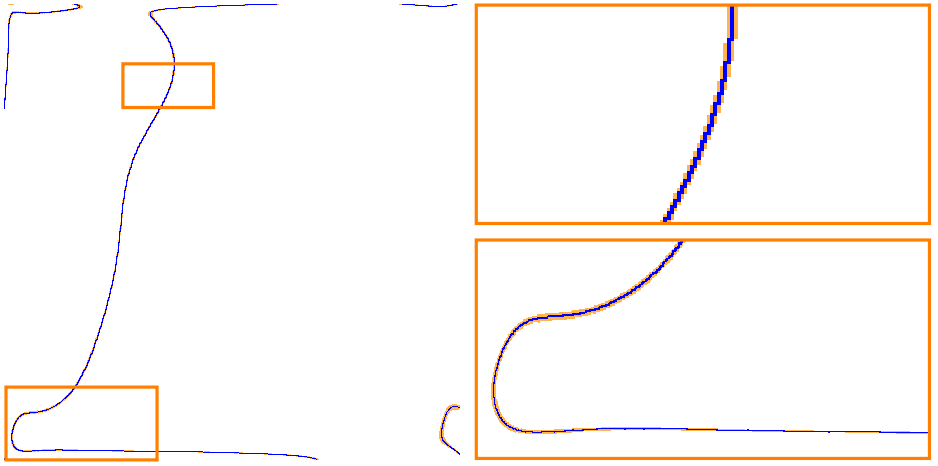
$P(X,Y) = \sum_{i+j=0}^{d} \xi_{i,j} X^i Y^j$



Polynôme Kostlan-Shub-Smale (KSS)

$P(X,Y) = \sum_{i+j=0}^{d} \sqrt{\frac{d!}{i!j!(d-i-j)!}} \xi_{i,j} X^i Y^j$

Dessin pour deux familles de polynômes



(a) (b)

Figure: Polynôme Kac de degré $d = 110$ à une résolution $N = 1\ 024$, $\frac{b}{b+j} = 24\%$

Dessin pour deux familles de polynômes

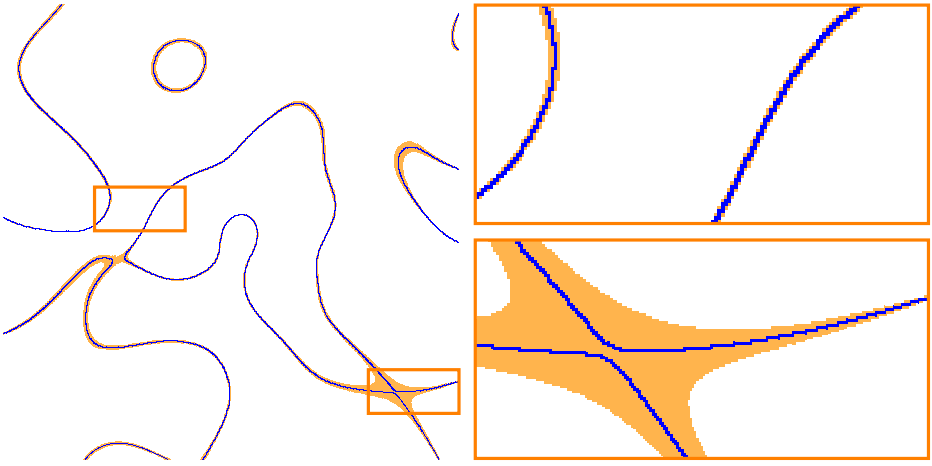


Figure: Polynôme KSS de degré $d = 40$ à une résolution de $N = 1\ 024$, $\frac{b}{b+j} = 19\%$

Comparaison aux logiciels de l'état de l'art

Nos méthodes

- dessin d'arêtes → arêtes englobant la courbe — faux positifs et faux négatifs
- dessin de pixels → pixels englobant la courbe — faux positifs

Des méthodes similaires

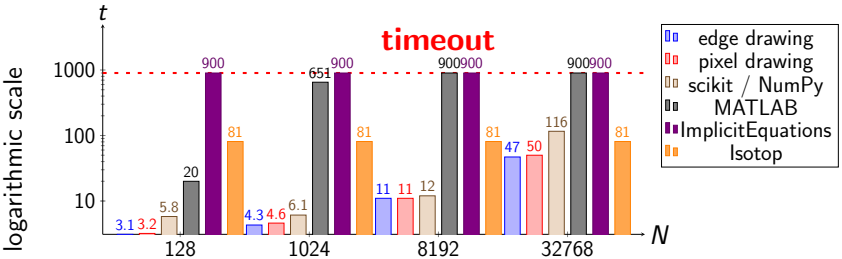
- scikit / NumPy → marching squares — faux négatifs
- MATLAB → méthode utilisée non trouvée — faux négatifs ?
- ImplicitEquations → subdivision 2D adaptative — faux positifs

Une méthode topologiquement correcte

- Isotop → décomposition algébrique cylindrique

Mesures de temps

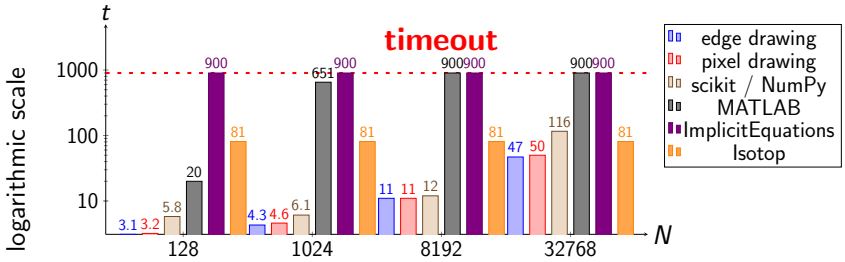
Comparaison pour un polynomial



Temps d'exécution pour un polynôme **Kac** de degré 40 (en secondes)

Mesures de temps

Comparaison pour un polynomial



Temps d'exécution pour un polynôme **Kac** de degré 40 (en secondes)

scikit: $O(dN^2)$

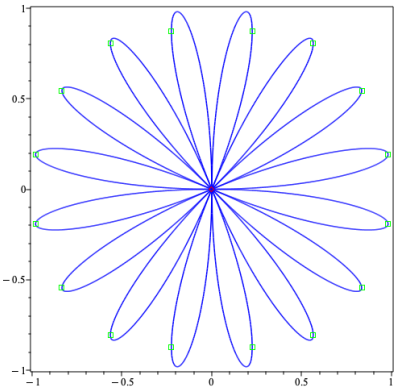
pas de garantie
lent quand d et N sont grands

Nos méthodes : $O(dNT)$
comme attendu $T = O(\log_2(N))$

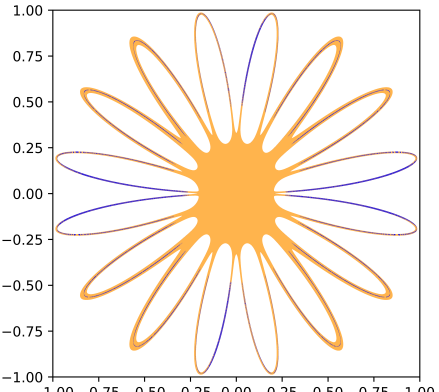
garanties
rapide quand d et N sont grands

Résultat pour une courbe singulière

Courbe : $\mathrm{dfold}_{8,1}$ du Challenge 14 d'Oliver Labs[13][37] ($d = 18$)



Isotop



Dessin de pixels

Conclusion

Contributions

- Deux algorithmes
  - arêtes englobantes
  - pixels englobants
- Algorithmes rapides pour les courbes et surfaces implicites en hautes résolutions : plus rapide que marching squares et marching cubes
- Meilleures garanties sur le dessin que marching squares
- Capables de gérer les hauts degrés ($d > 20$) et les hautes résolutions ($N > 1\ 000$)