Delaunay triangulation:

Implementation

Monique Teillaud

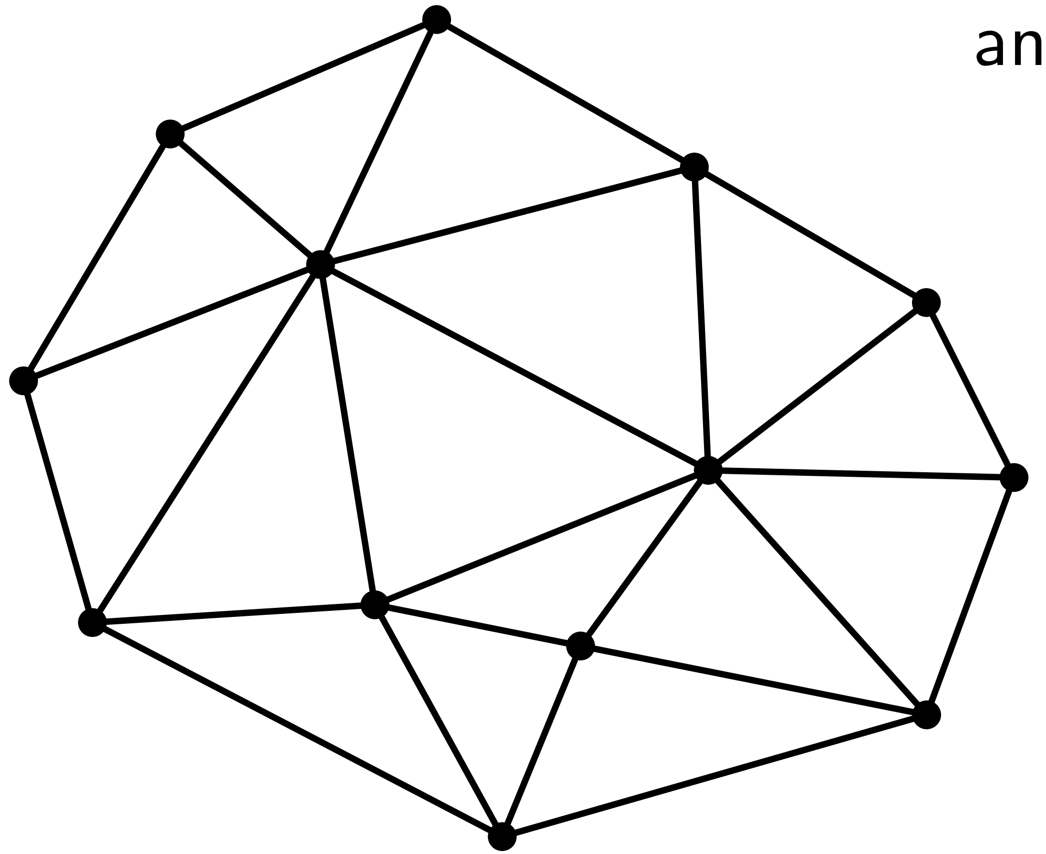
Choosing an algorithm

(not only) laziness

Incremental algorithm

fully dynamic

any dimension



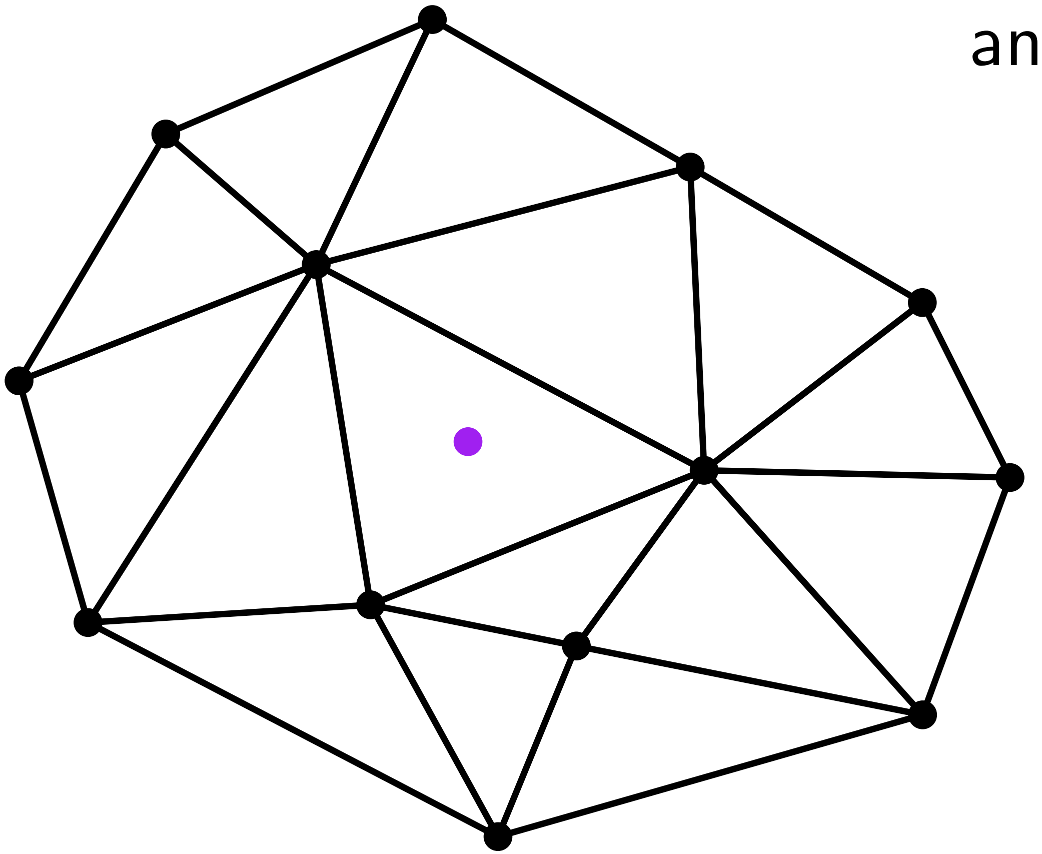
Choosing an algorithm

(not only) laziness

Incremental algorithm

fully dynamic

any dimension



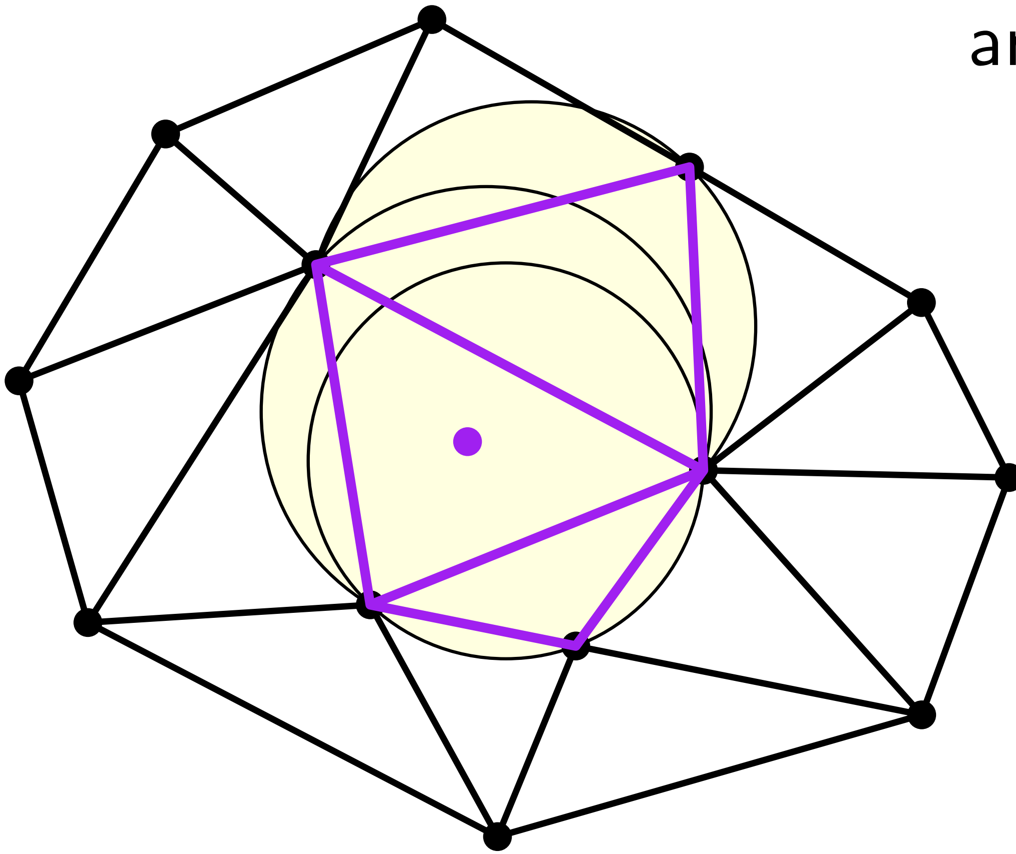
Choosing an algorithm

(not only) laziness

Incremental algorithm

fully dynamic

any dimension



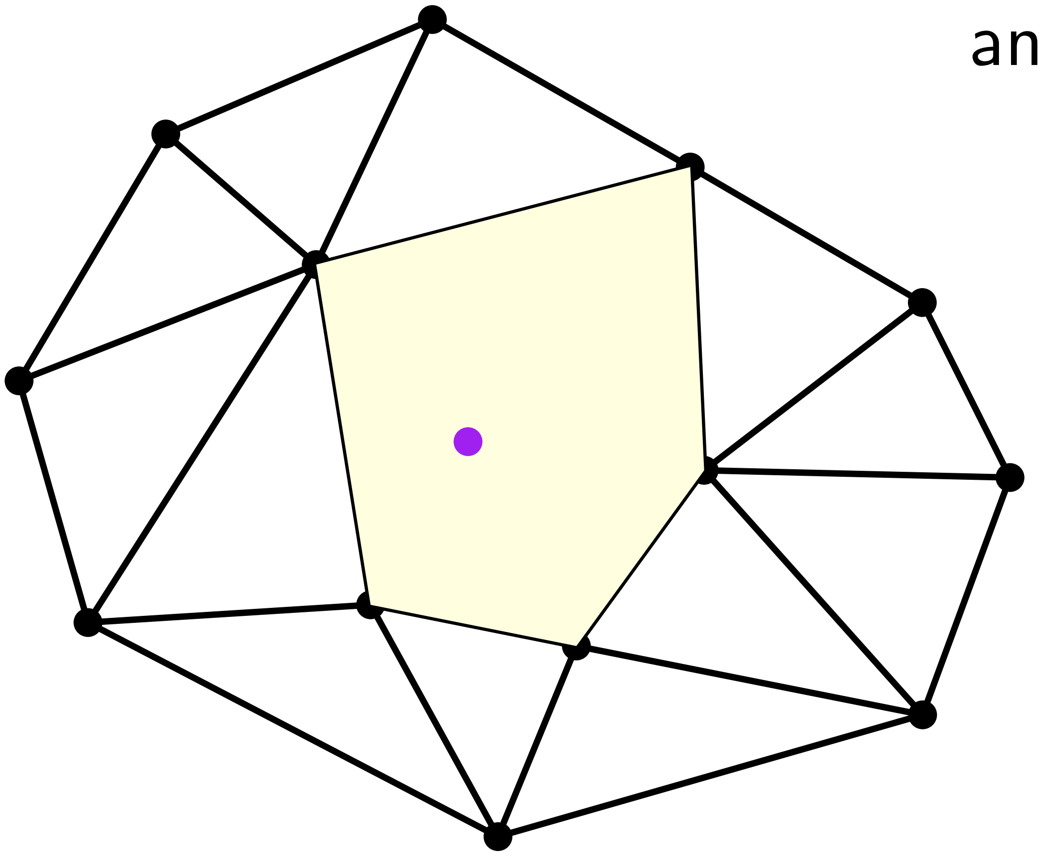
Choosing an algorithm

(not only) laziness

Incremental algorithm

fully dynamic

any dimension



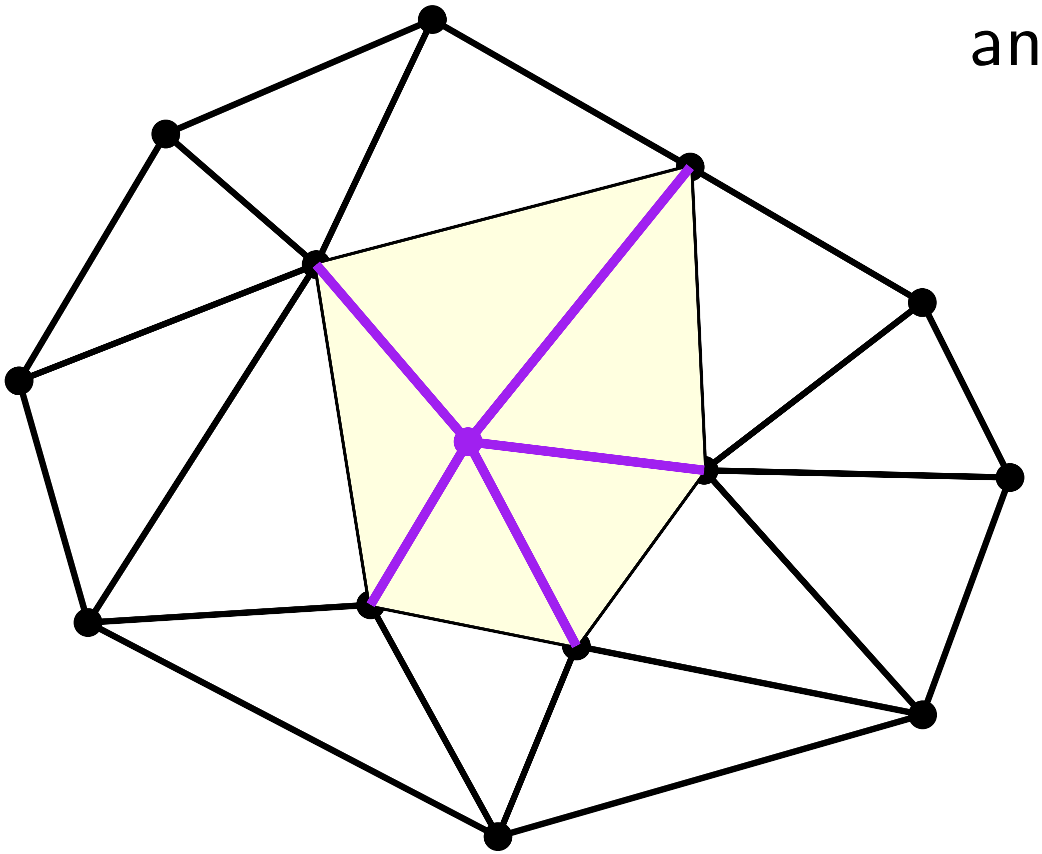
Choosing an algorithm

(not only) laziness

Incremental algorithm

fully dynamic

any dimension

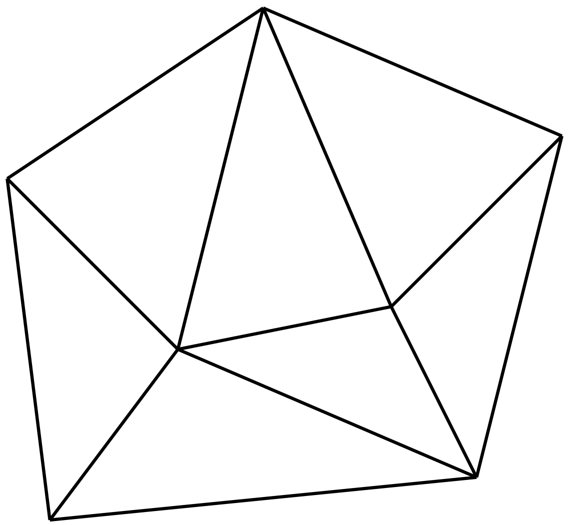


Representation

walk: access to

- vertices of a triangle
- neighbors of a triangle

in constant time

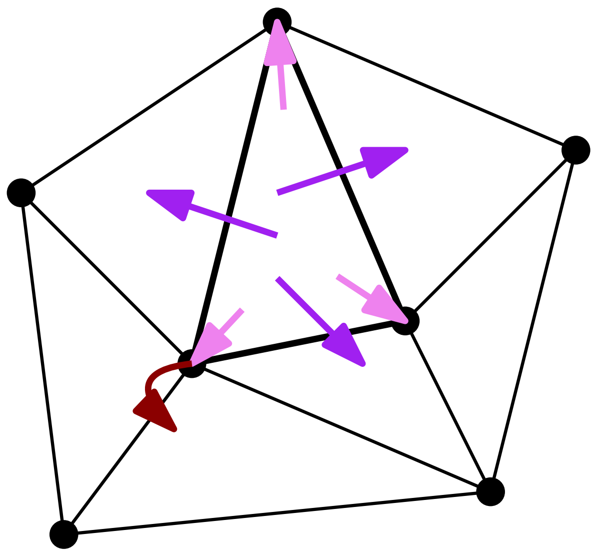


Representation

walk: access to

- vertices of a triangle
- neighbors of a triangle

in constant time



combinatorics:

store

- d-simplices
- vertices

adjacency relations as pointers

geometry

store

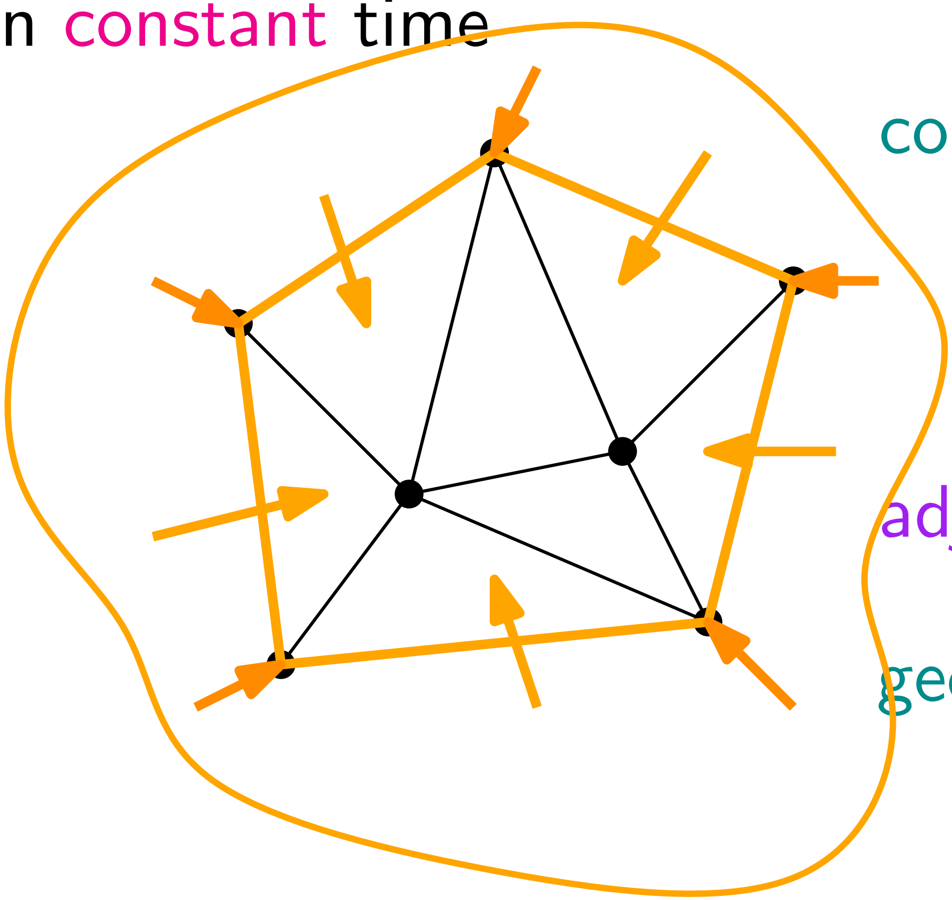
- points in vertices

Representation

walk: access to

- vertices of a triangle
- neighbors of a triangle

in constant time



combinatorics:

store

- d-simplices
- vertices

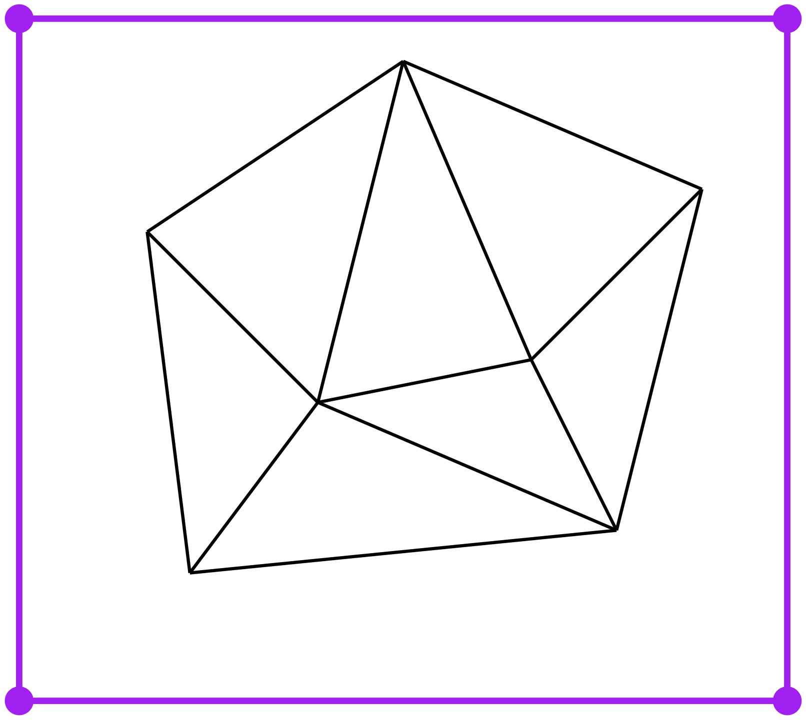
adjacency relations as pointers

geometry

what about the infinite region? unbounded size. . .

Representation

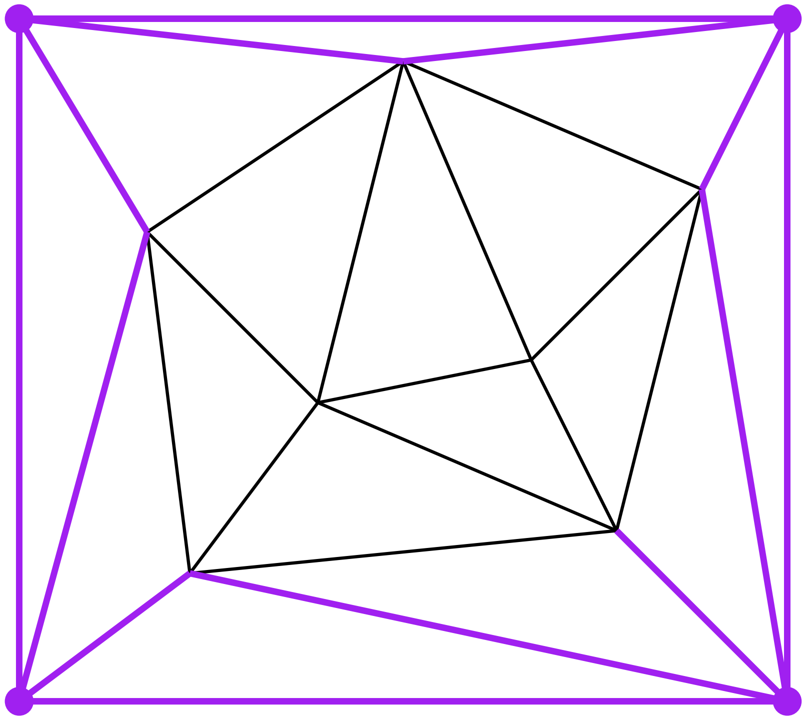
add a bounding box?



requires to know points in advance

Representation

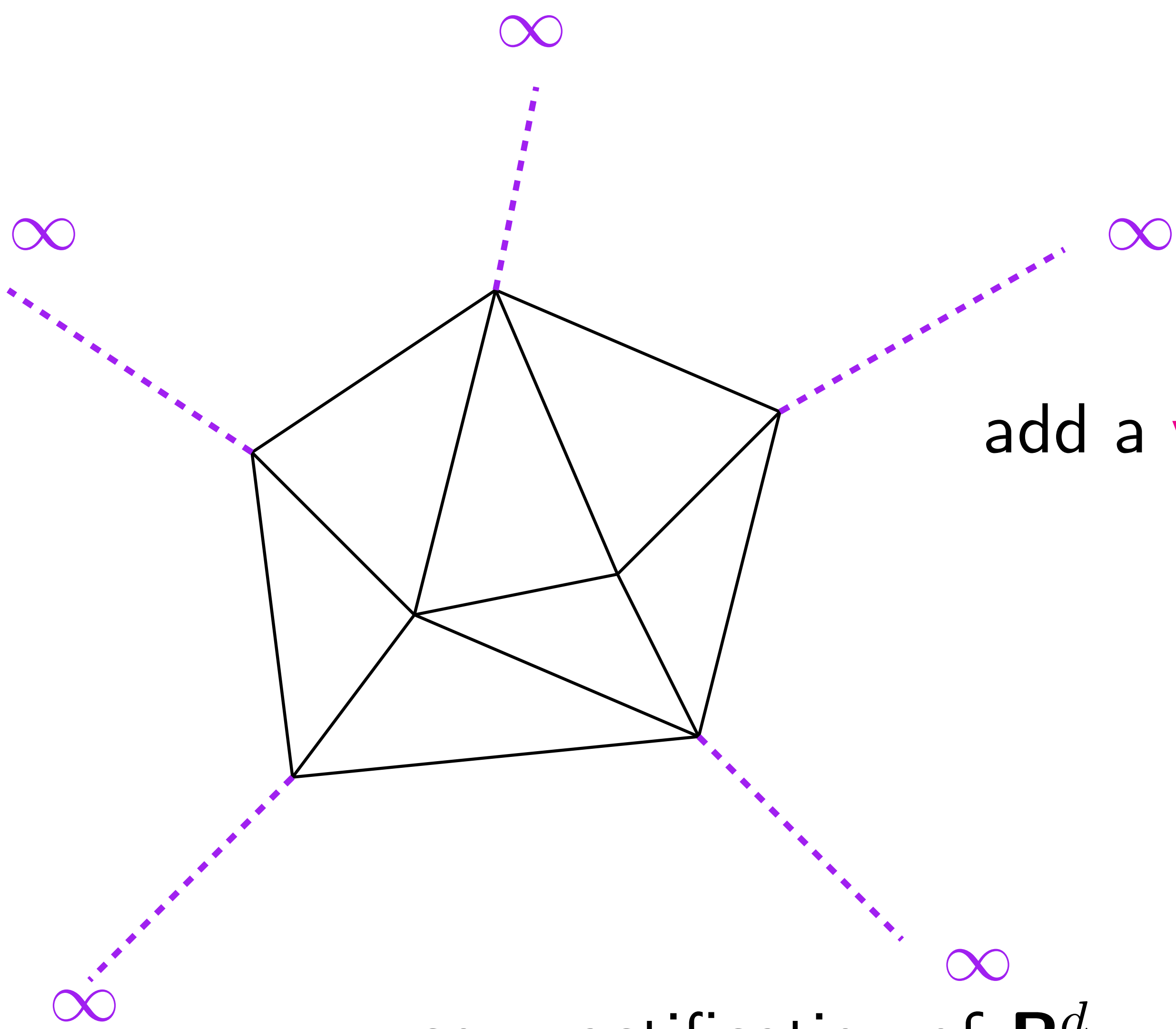
add a bounding box?



requires to know points in advance

creates ugly triangles

Representation

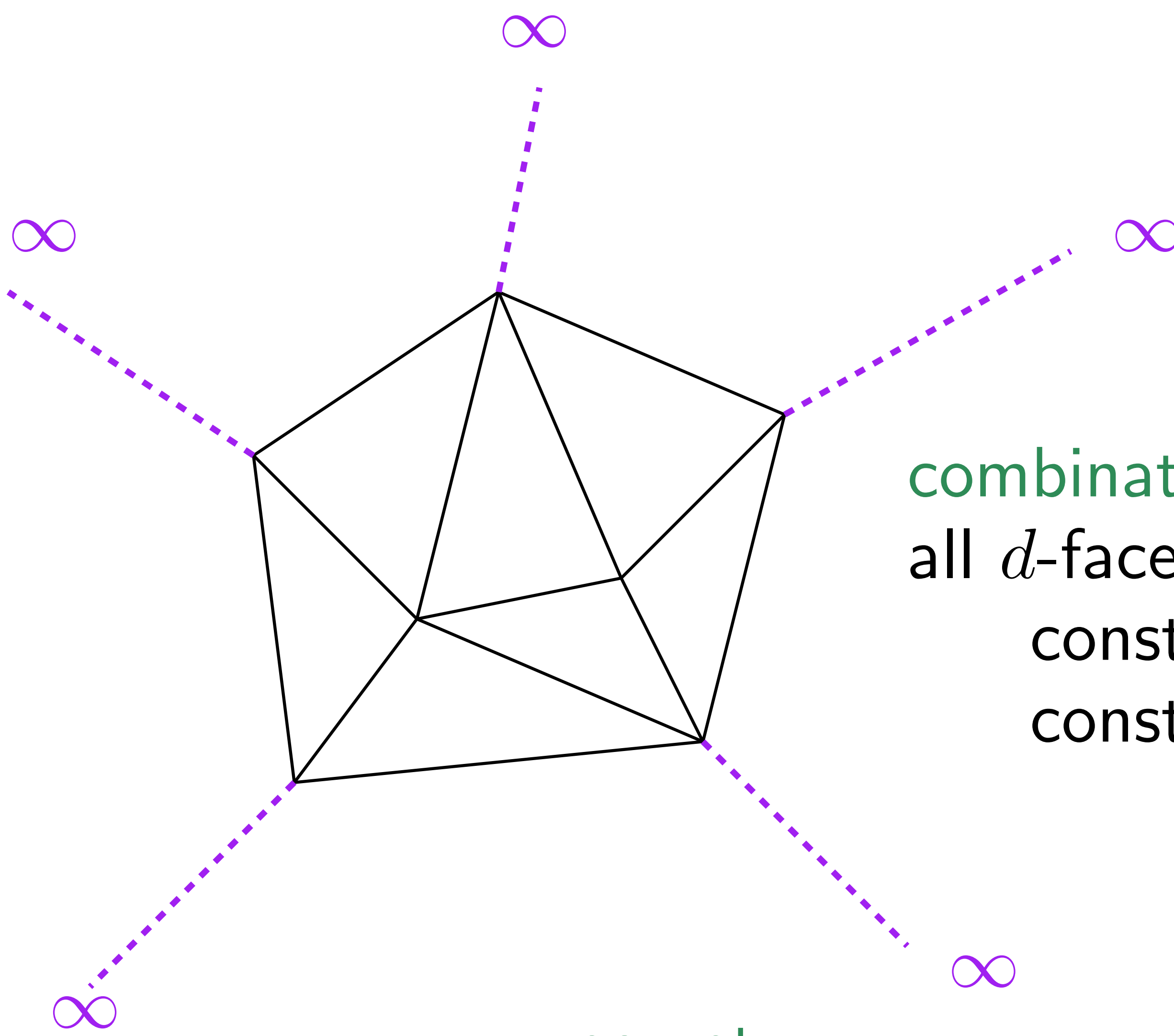


add a vertex at infinity

compactification of $\mathbf{R}^d$

$\Longrightarrow$ triangulation of the sphere $\mathbf{S}^d$

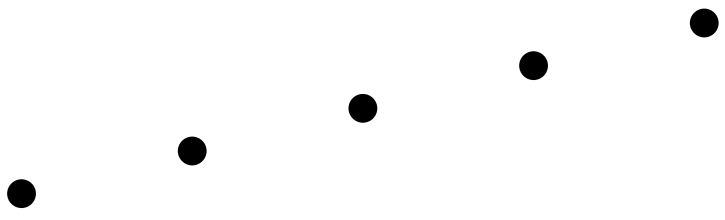
Representation



combinatorics
all d-faces are simplices
constant description
constant-time access...

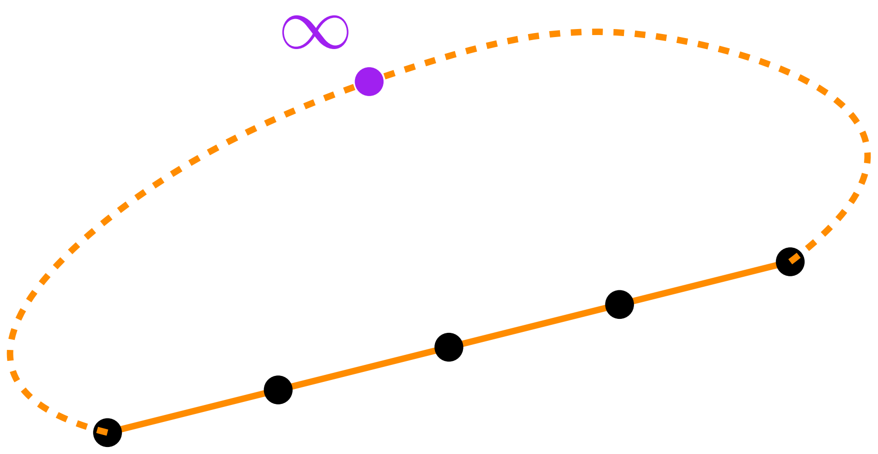
geometry
infinite simplex $=$ half-space
no *point* in the infinite vertex

Representation



what if all points are collinear?

Representation



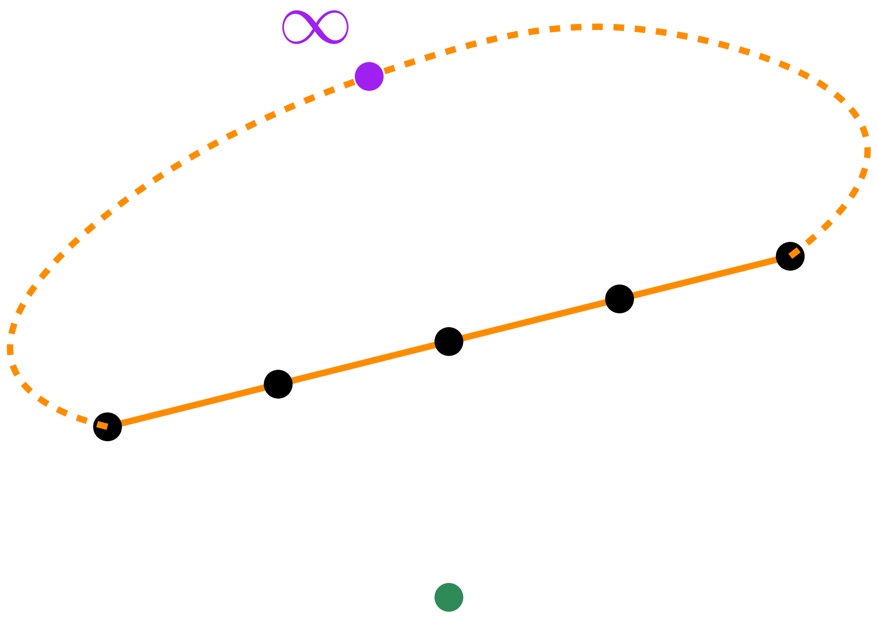
triangulation of $\mathbf{S}^1$

dD triangulation, $d \geq 2$

"incomplete" simplices

what if all points are collinear?

Representation

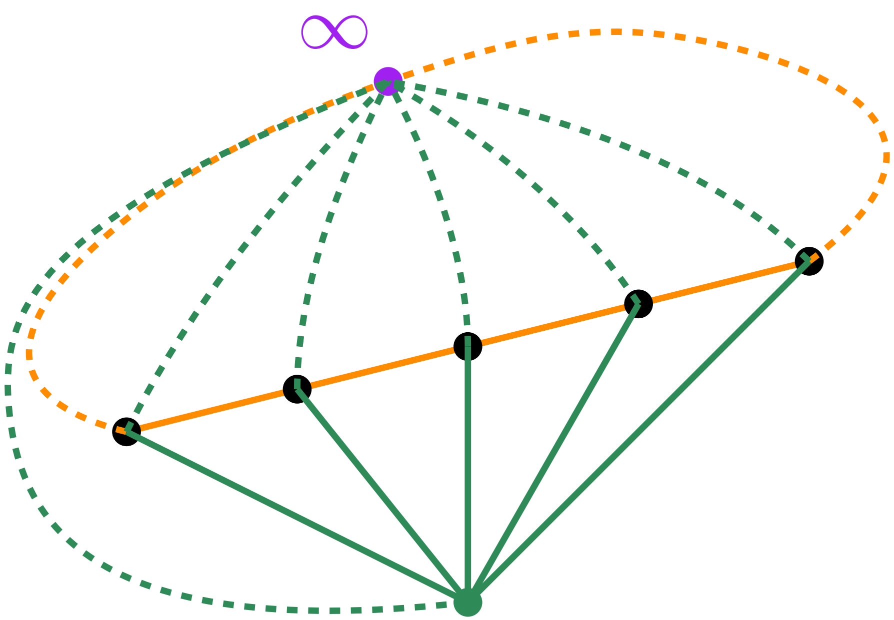


triangulation of $\mathbf{S}^1$

what if all points are collinear?

what if a non-collinear point comes in ?

Representation



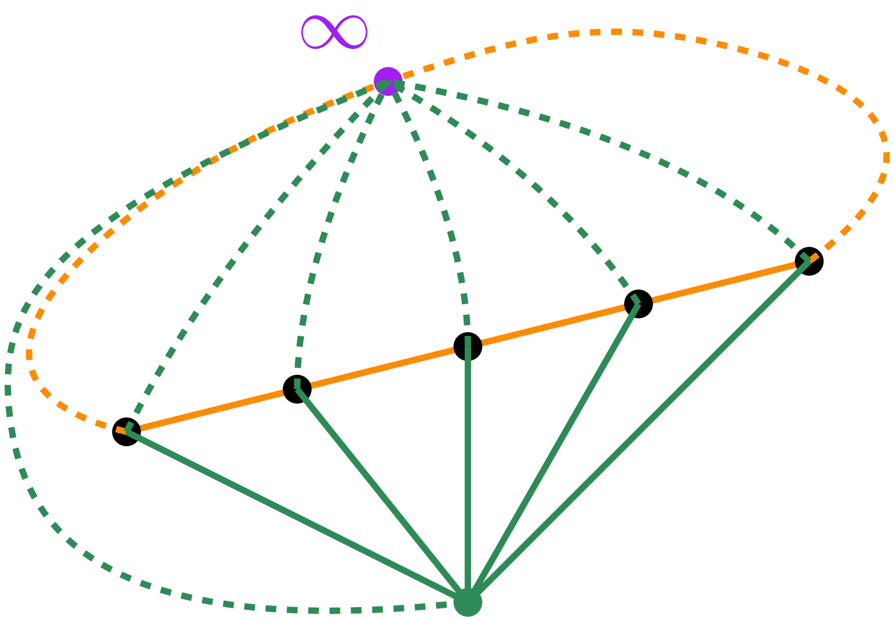
triangulation of $\mathbf{S}^1$

$\longrightarrow$ triangulation of $\mathbf{S}^2$

what if all points are collinear?

what if a non-collinear point comes in ?

Representation



triangulation of $\mathbf{S}^1$

$\longrightarrow$ triangulation of $\mathbf{S}^2$

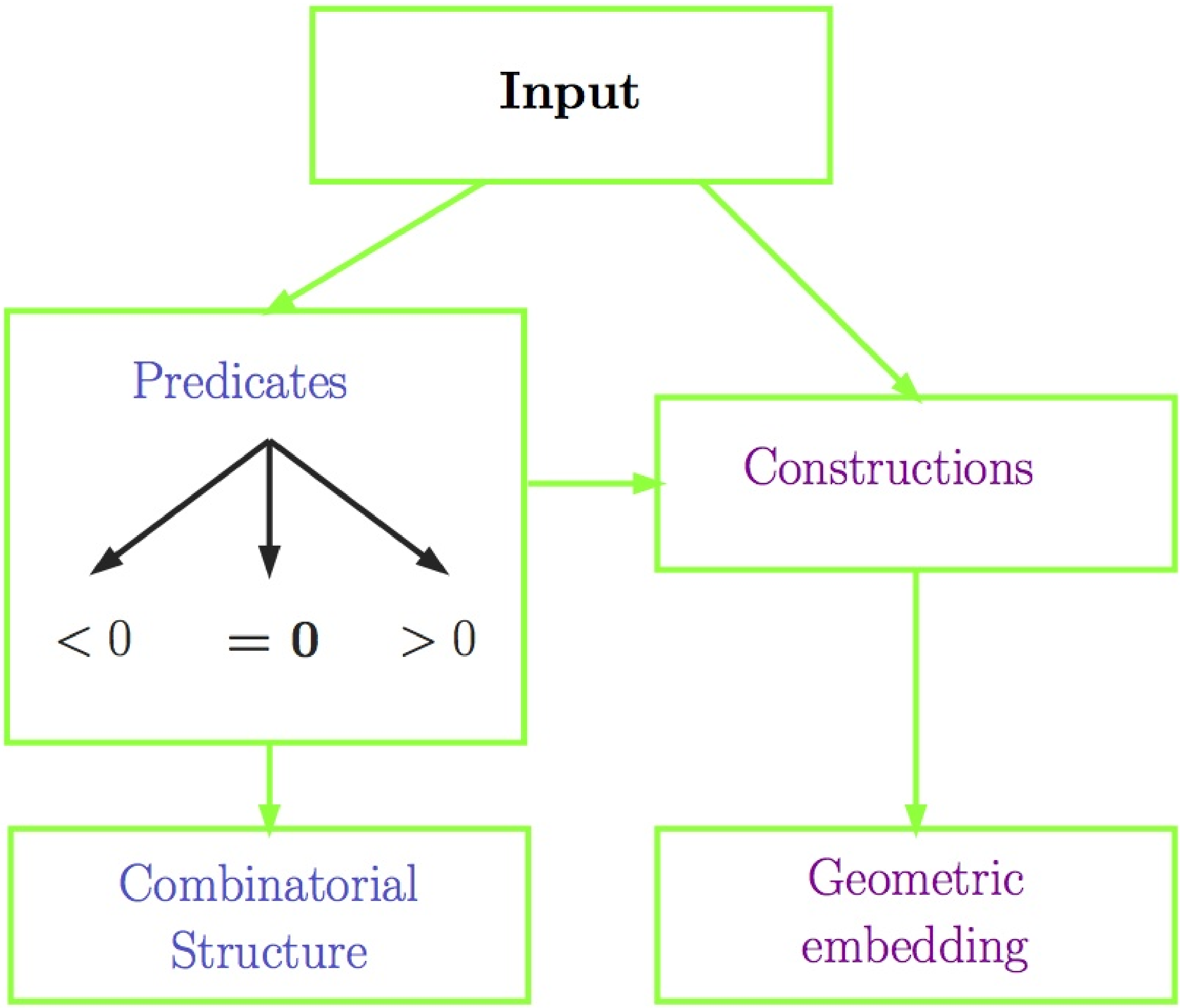
$\longrightarrow$ triangulation of $\mathbf{S}^3$

what if all points are collinear?

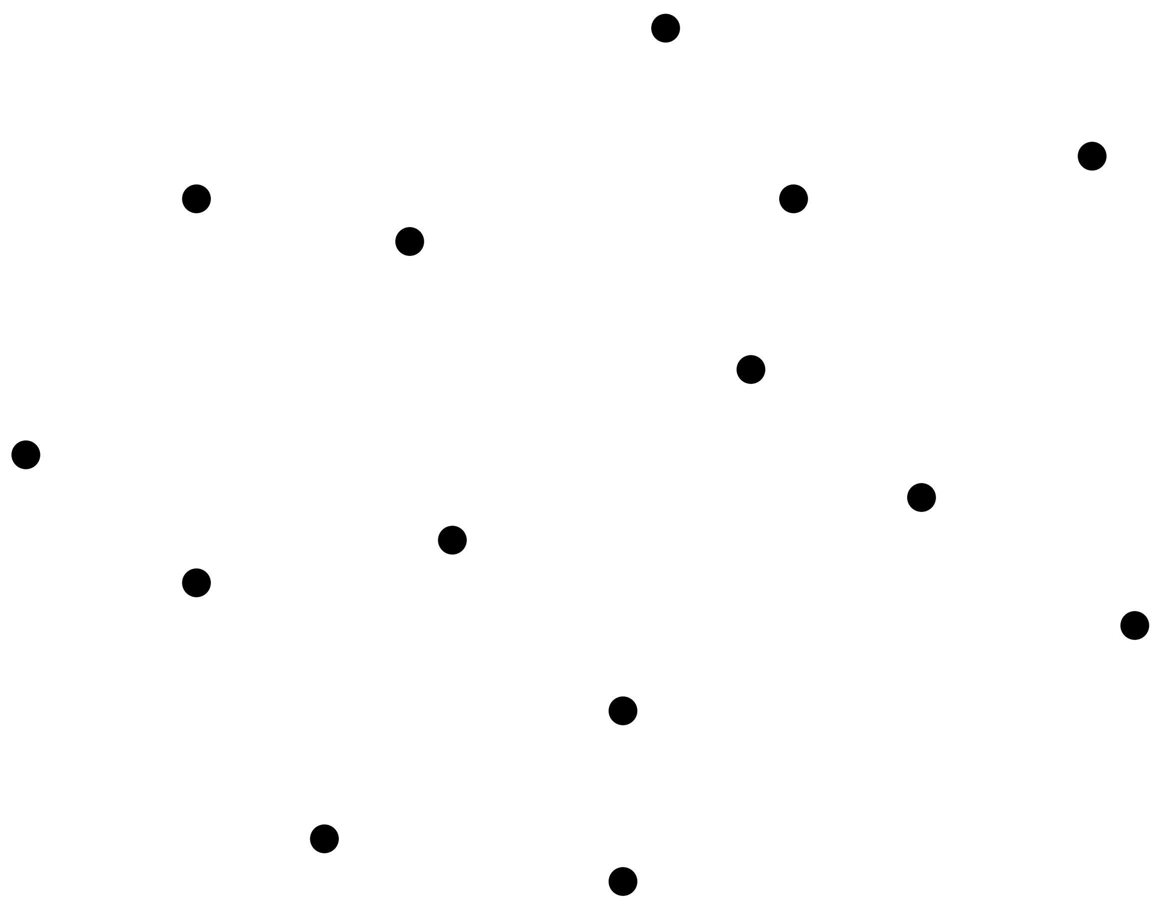
what if a non-collinear point comes in ?

what if a non-coplanar point comes in ?

Arithmetic computations

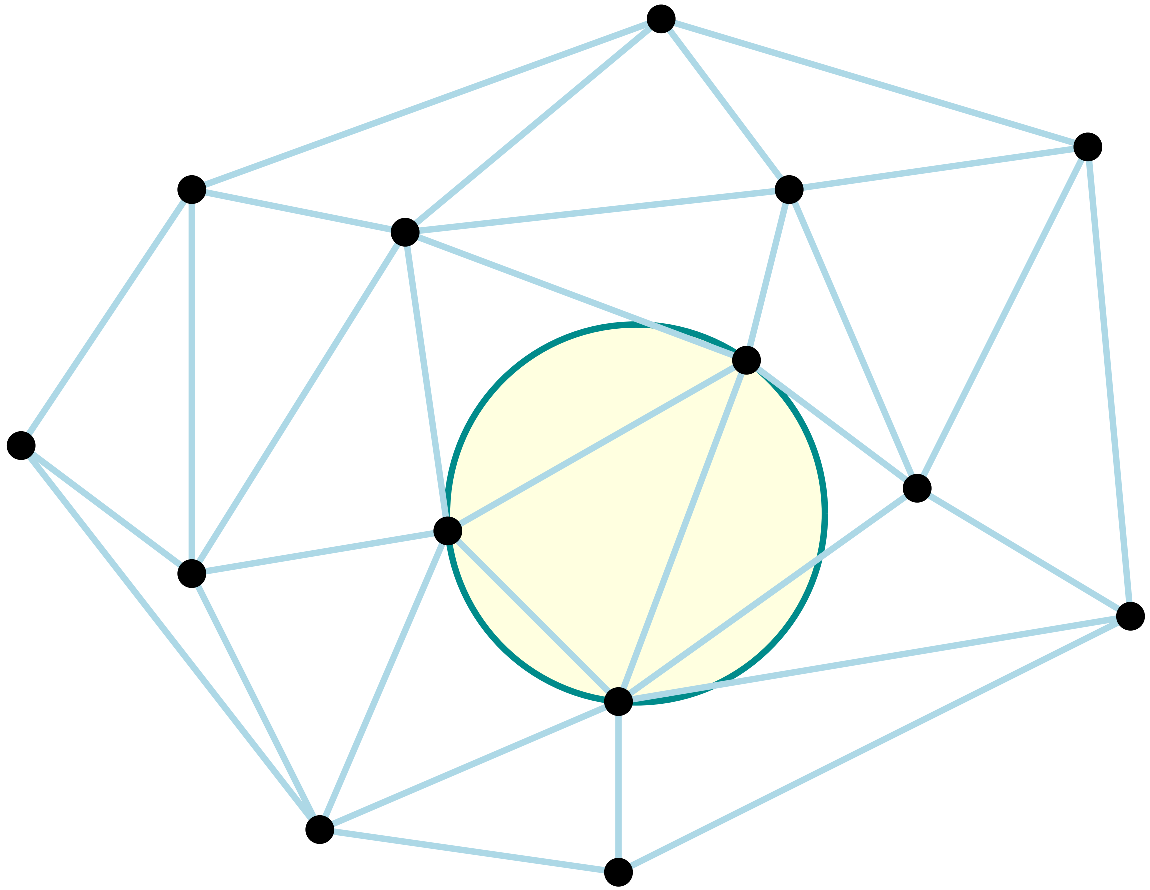


Arithmetic computations



Arithmetic computations

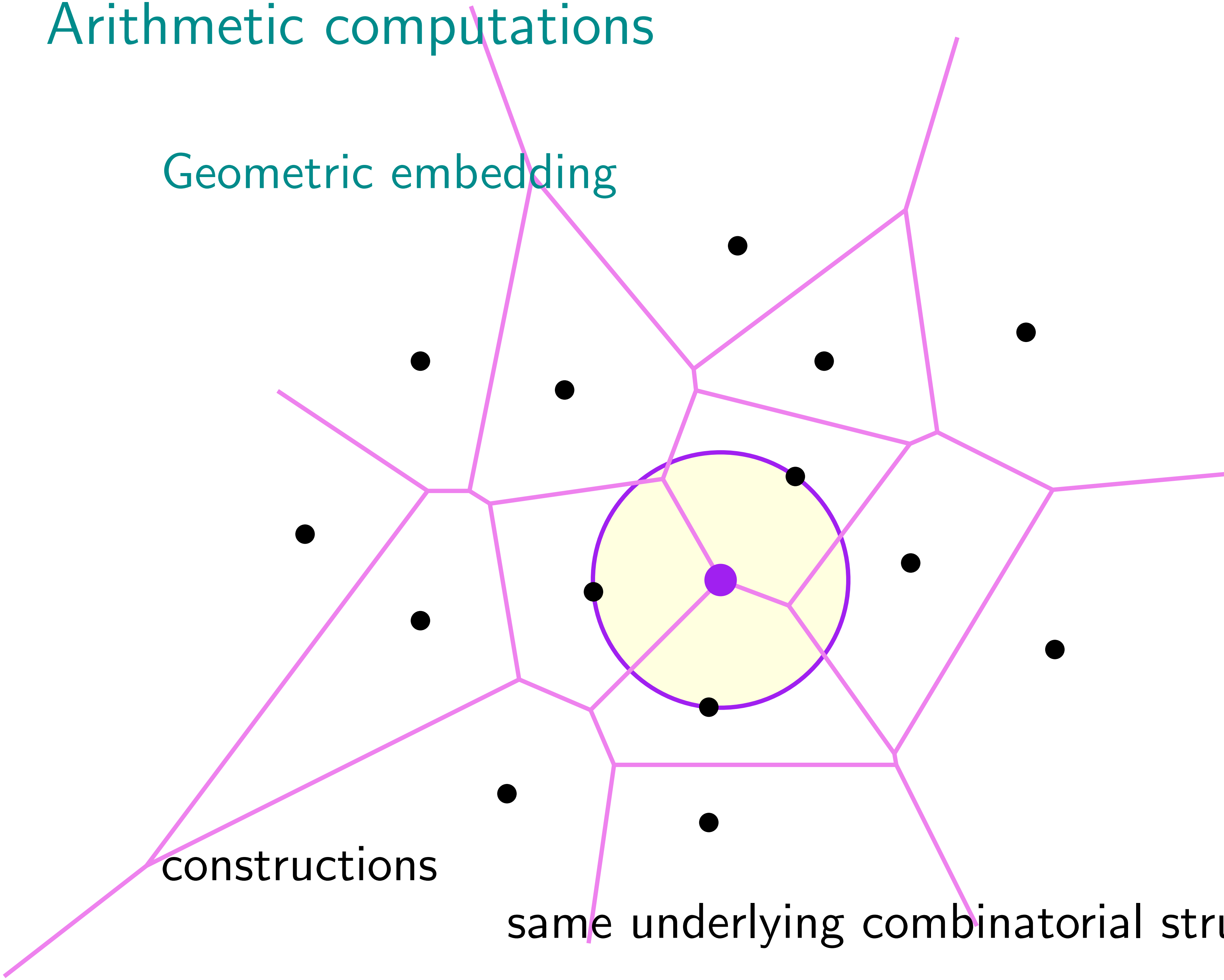
Combinatorial structure



only predicates

Arithmetic computations

Geometric embedding



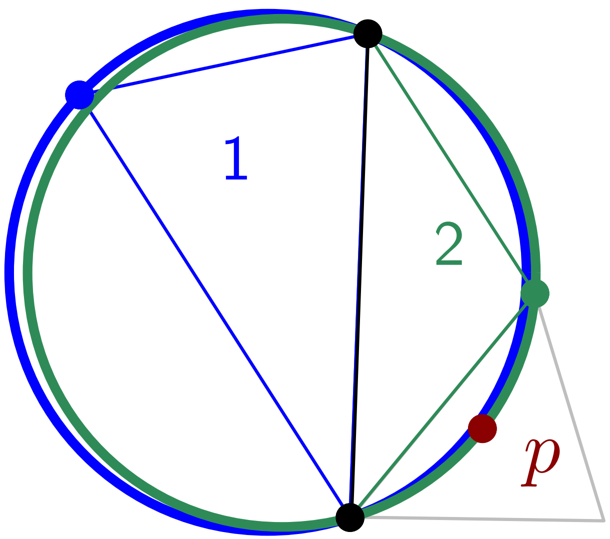
constructions

same underlying combinatorial structure

Arithmetic computations

inexact evaluation of predicates

NOT just an imprecision in the result



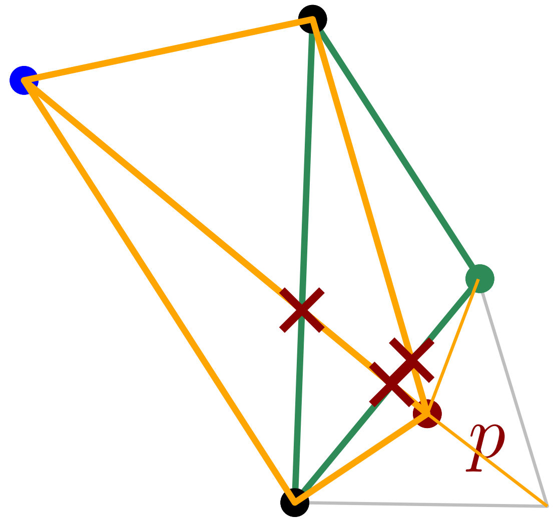
$\text{in\_disk}_1(p) = \text{true}$

$\text{in\_disk}_2(p) = \text{false}$

Arithmetic computations

inexact evaluation of predicates

NOT just an imprecision in the result



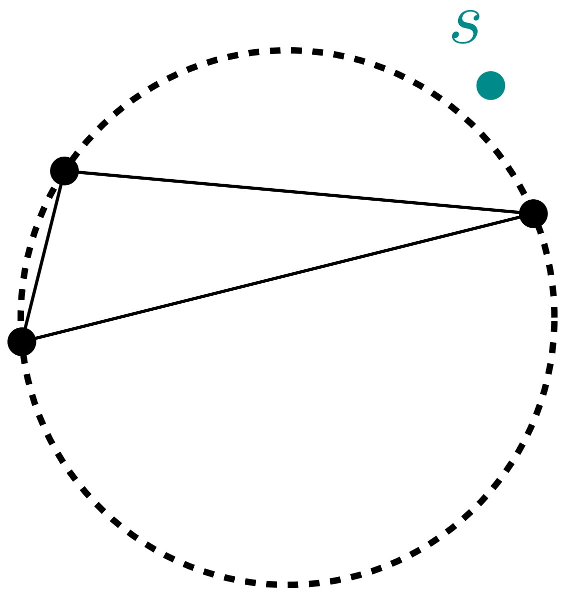
$\text{in\_disk}_1(p) = \text{true}$

$\text{in\_disk}_2(p) = \text{false}$

inconsistencies!

algorithms fail

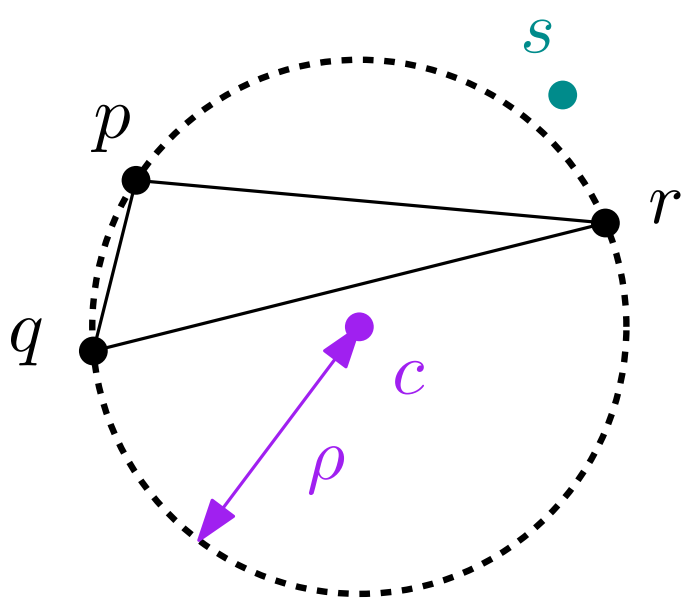
Arithmetic computations



Predicate

Is s inside or outside the disk?

Arithmetic computations



Predicate

Is s inside or outside the disk?

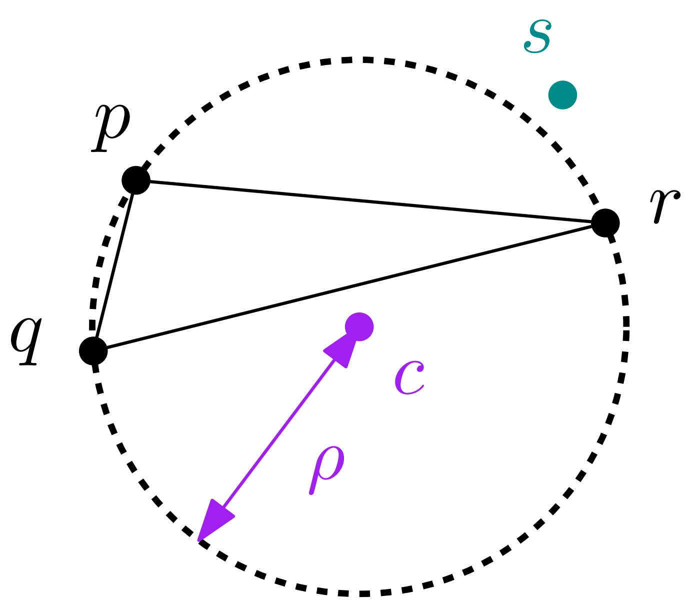
circle $\mathcal{C}$ through p, q, r

unknowns c, ρ

solve $\longrightarrow$

- center c
- radius ρ

Arithmetic computations



Predicate

Is s inside or outside the disk?

circle $\mathcal{C}$ through p, q, r

unknowns c, ρ

solve $\longrightarrow$

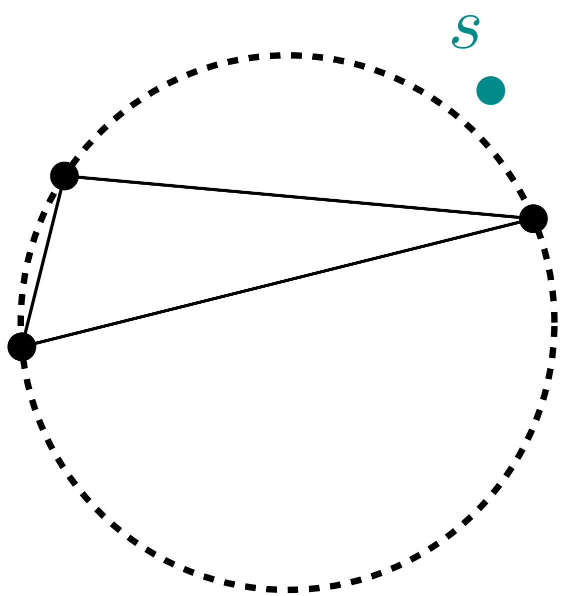
- center c
- radius ρ

Bad idea. . . reals do not exist!

rounding errors $\hookrightarrow p, q, r \notin \mathcal{C}(c, \rho)$

"random" result for s

Arithmetic computations

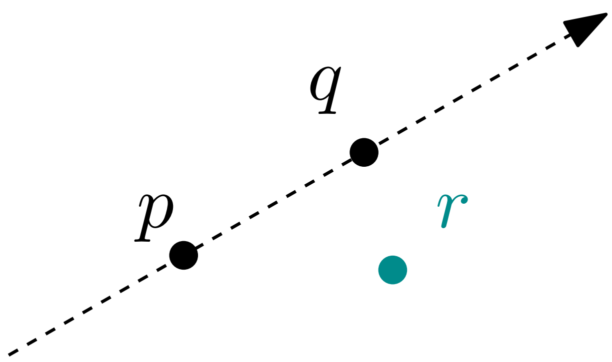


Predicate

Is s inside or outside the disk?

in_disk(p, q, r, s) sign $\begin{vmatrix} 1 & 1 & 1 & 1 \\ p_x & q_x & r_x & s_x \\ p_y & q_y & r_y & s_y \\ p_x^2 + p_y^2 & q_x^2 + q_y^2 & r_x^2 + r_y^2 & s_x^2 + s_y^2 \end{vmatrix}$

Arithmetic computations



A simpler predicate

Is r on the left or right side?

$\mathrm{orient}(p, q, r)$

$$\text{sign}\begin{vmatrix} 1 & 1 & 1 \\ p_x & q_x & r_x \\ p_y & q_y & r_y \end{vmatrix}$$

Arithmetic computations

`double` numbers are not reals
53 binary digits

$p = (0.5{+}x.u,\ 0.5{+}y.u)$

$\quad 0 \leq\ x, y\ < 256,$

$\quad u = 2^{-53}$

$q = (12, 12)$

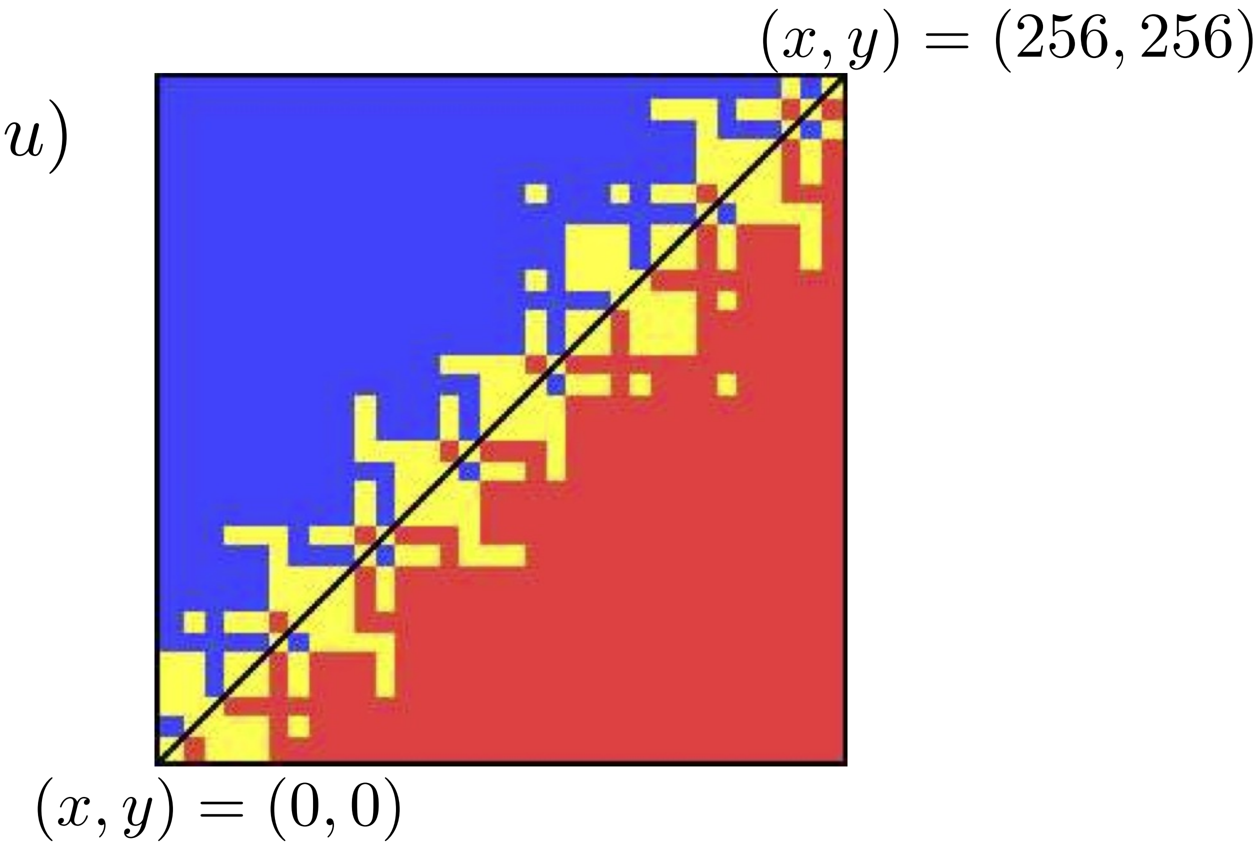
$r = (24, 24)$

orient(p, q, r)

> 0

$= 0$

< 0



fast, but wrong

Arithmetic computations

a solution:

rely on an exact arithmetic package (multiprecision, etc)

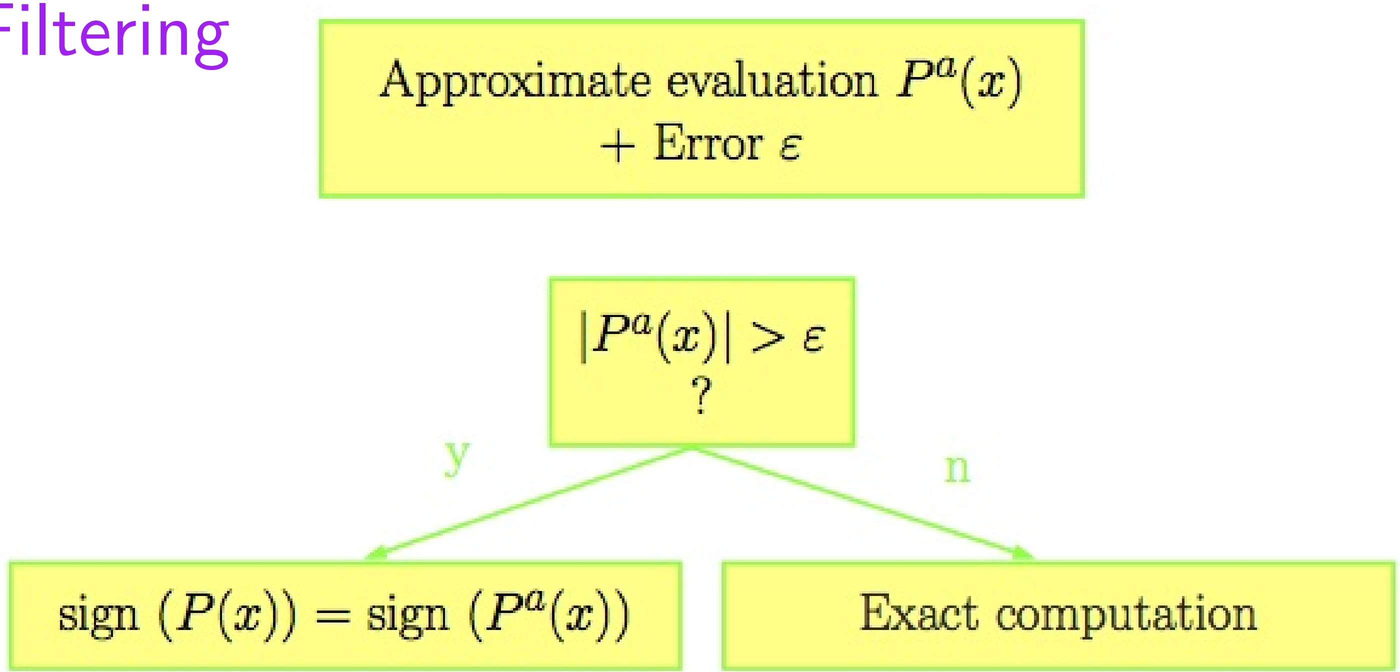
powerful, but slow

Arithmetic computations

Exact Geometric Computing paradigm

$=$ exact predicates, $\neq$ exact arithmetics

Filtering



easy cases are more frequent
$\Longrightarrow$ cost $\simeq$ cost of approximate (double) computation

Arithmetic computations

Dynamic filtering   interval arithmetic

error on $+, -, *, /, \sqrt{}$ known (IEEE 754)

$$\left[\, \underline{a}, \overline{a} \,\right] + \left[\, \underline{b}, \overline{b} \,\right] = \left[\, \underline{a} \;\underline{+}\; \underline{a} \;,\; \overline{a} \;\overline{+}\; \overline{b} \,\right]$$

and propagate. . .

Arithmetic computations

Dynamic filtering   interval arithmetic

error on $+, -, *, /, \sqrt{}$ known (IEEE 754)

$$[\,\underline{a}, \overline{a}\,] + [\,\underline{b}, \overline{b}\,] = [\,\underline{a} \,\underline{+}\, \underline{a}\;,\; \overline{a} \,\overline{+}\, \overline{b}\,]$$

and propagate. . .

$0 \notin [\,\underline{\textbf{result}}\;,\; \overline{\textbf{result}}\,]$ **?**

y → sign certified

n → exact arithmetic

Choosing an algorithm

Degree of predicates & number of operations

$\longrightarrow$ constant in $O()$

$\longrightarrow$ size of errors

$\longrightarrow$ length of integers for exact arithmetic

Choosing an algorithm

Degree of predicates & number of operations

$\longrightarrow$ constant in $O()$

$\longrightarrow$ size of errors

$\longrightarrow$ length of integers for exact arithmetic

Incremental algorithm

only uses intrinsic predicates   orient, in_disk

any algorithm computing Delaunay triangulation
is able to answer them

Sweep

uses ad hoc higher degree predicates

Choosing an algorithm

Degree of predicates & number of operations

$\longrightarrow$ constant in $O()$

$\longrightarrow$ size of errors

$\longrightarrow$ length of integers for exact arithmetic

Incremental algorithm

only uses intrinsic predicates   orient, in_disk

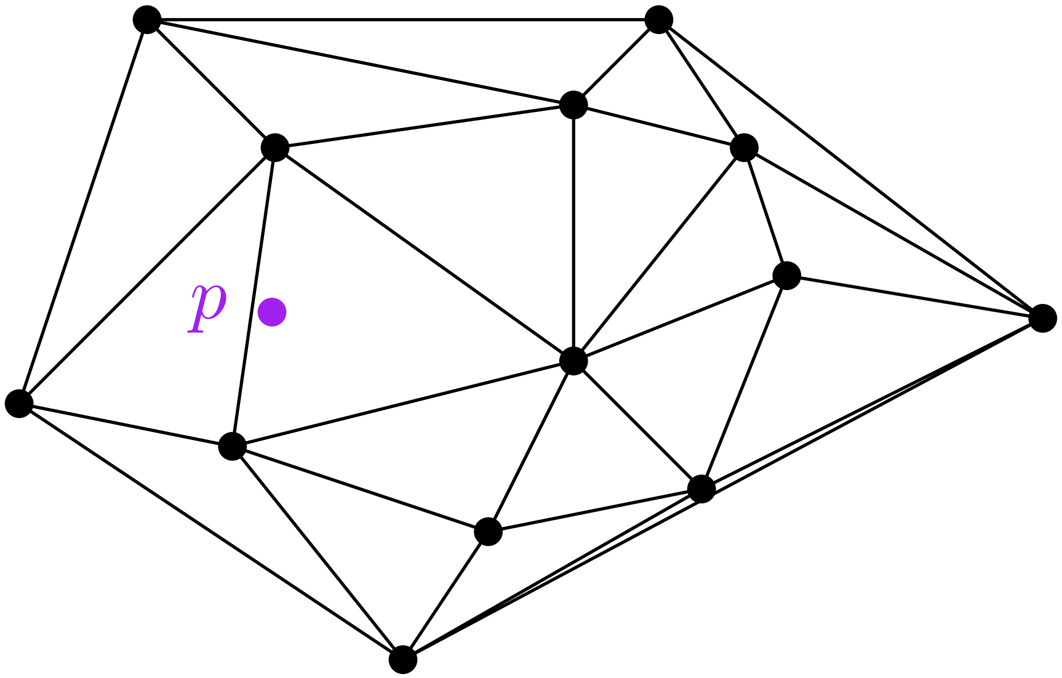
any algorithm computing Delaunay triangulation
is able to answer them

Sweep

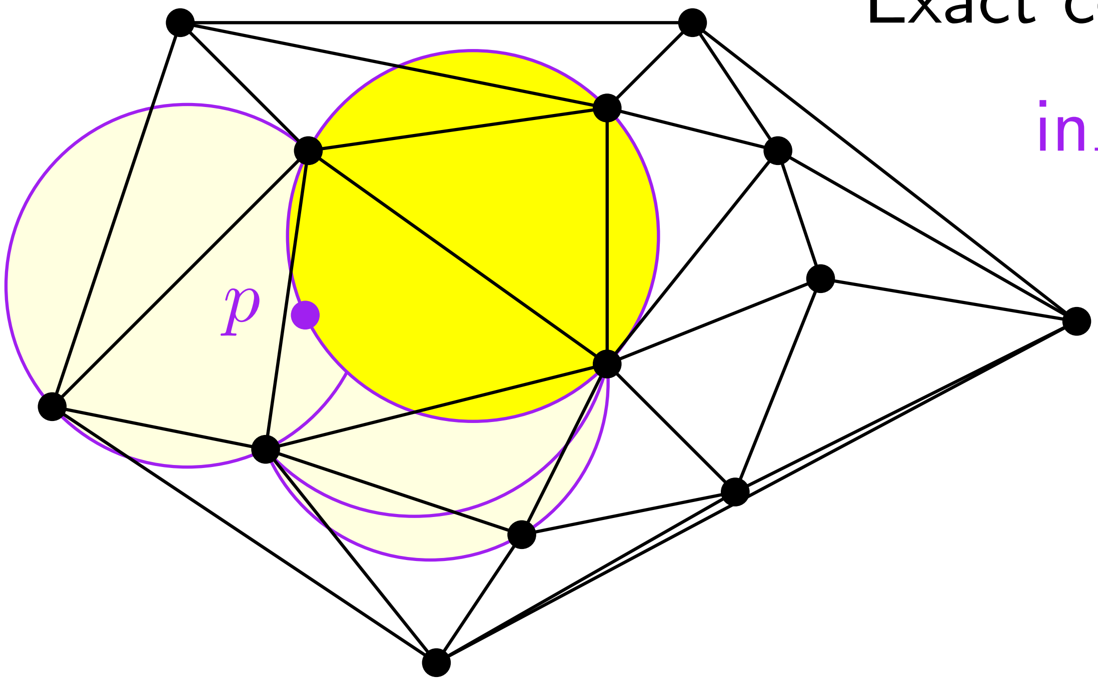
uses ad hoc higher degree predicates

laziness is not the only criterion

Degeneracies



Degeneracies



Exact computation

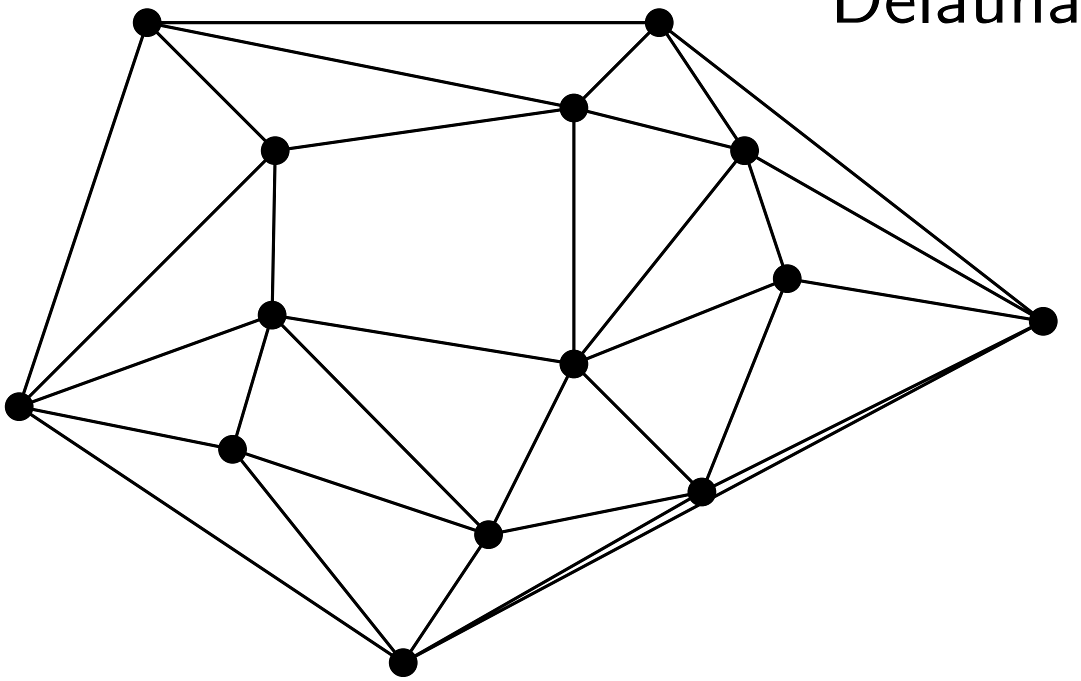
$\text{in\_disk}(\cdot, \cdot, \cdot, p) = 0$

what if p lies on a circle?

yes, it does happen!

input data are rounded

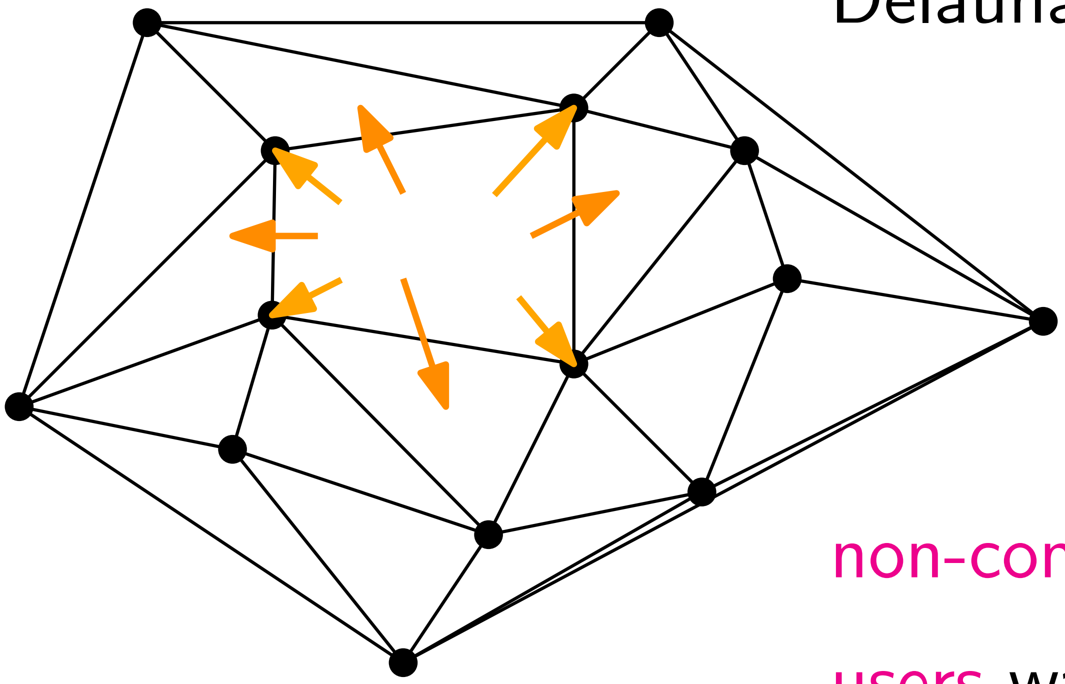
Degeneracies



Delaunay complex

non-simplicial faces

Degeneracies



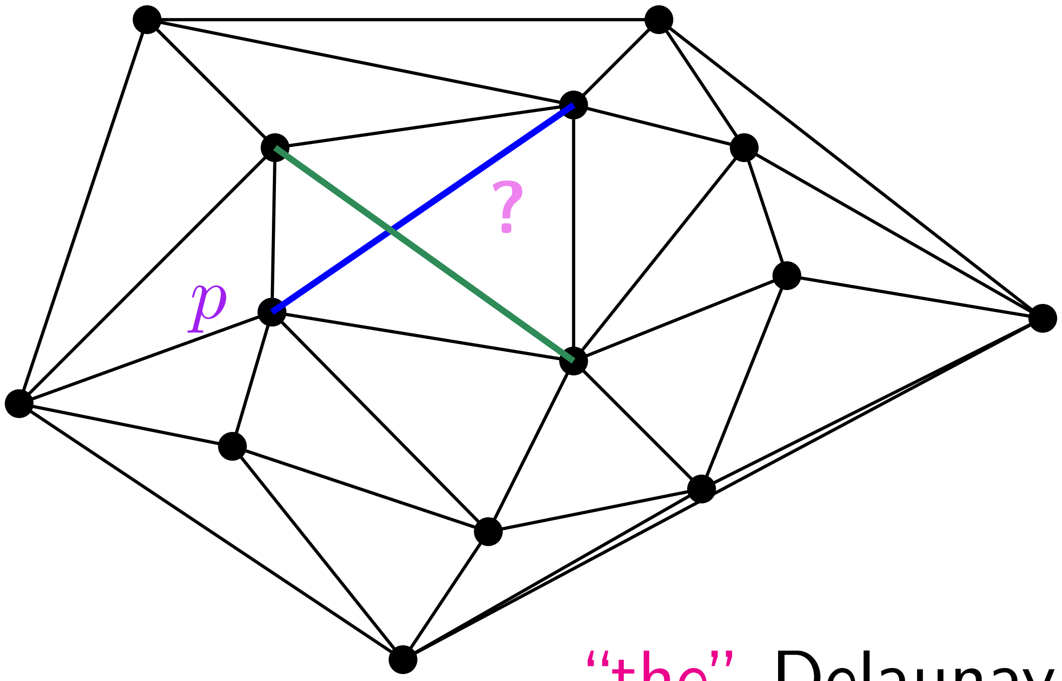
Delaunay complex

non-simplicial faces

non-constant storage and access

users want triangles

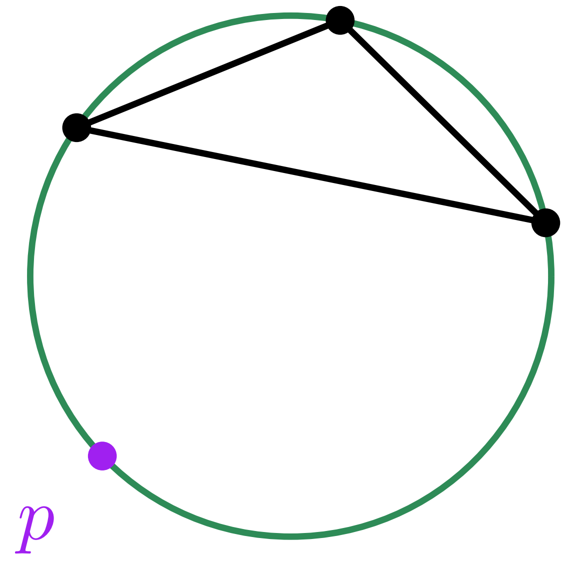
Degeneracies



"the" Delaunay triangulation is not unique

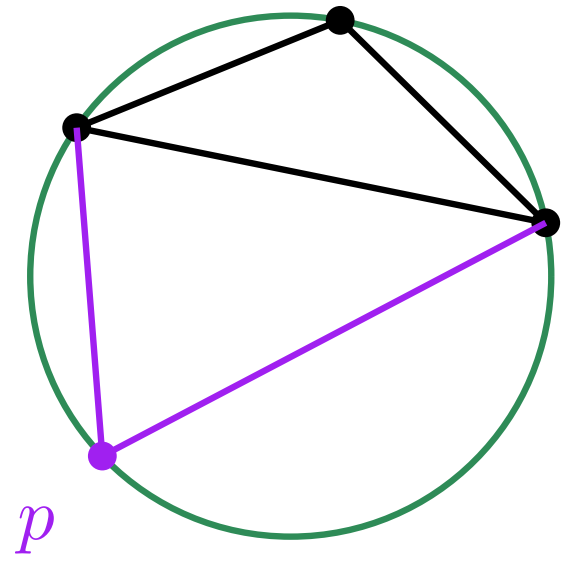
Degeneracies

Simulating the absence of degeneracies



Degeneracies

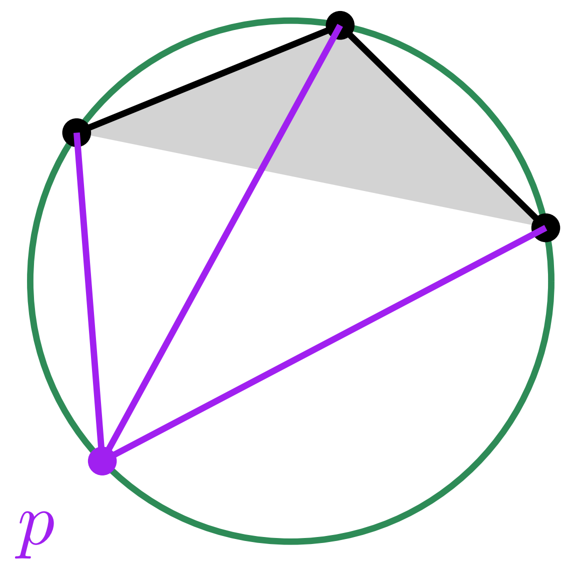
Simulating the absence of degeneracies



as if p outside

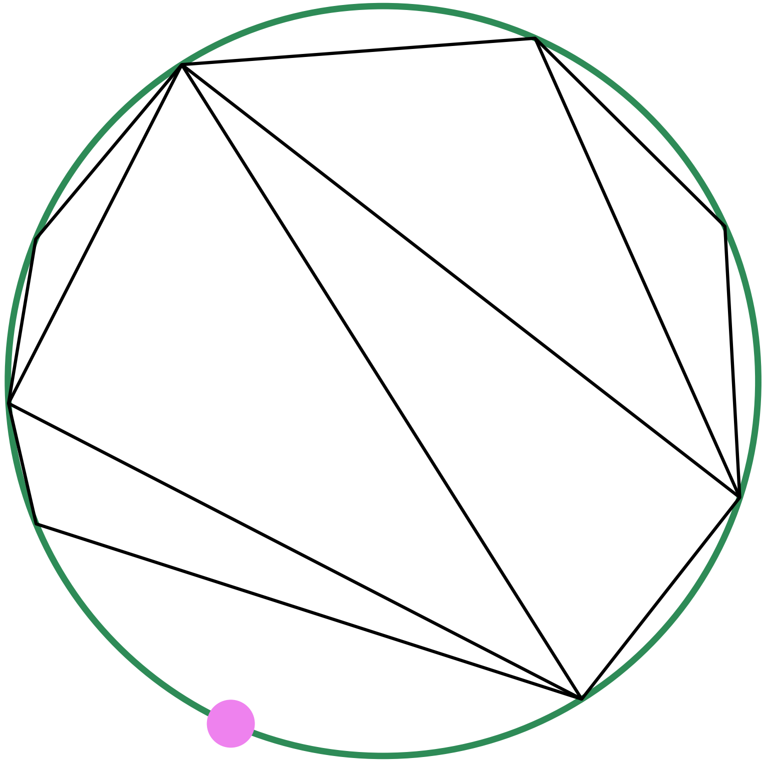
Degeneracies

Simulating the absence of degeneracies

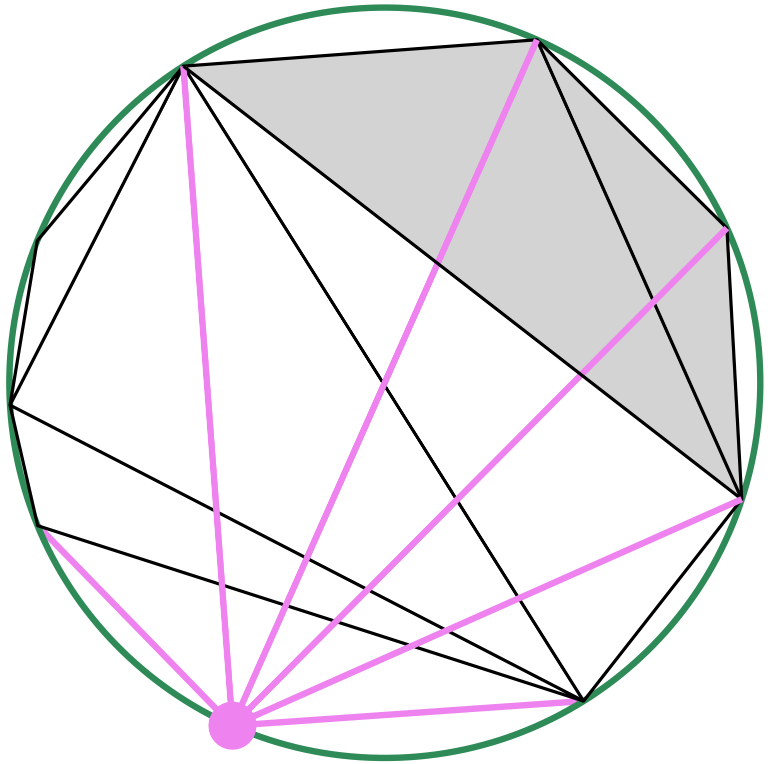


as if p inside

Degeneracies



Degeneracies



decisions must be made in a consistent way

Degeneracies

Symbolic perturbation

Input data $\mapsto$ data depending on a symbolic parameter ε

- $\varepsilon = 0$: (maybe) degenerate problem
- $\varepsilon \neq 0$: non-degenerate problem $\mapsto$ Result(ε)

Final result $= \lim_{\varepsilon \to 0^+}$ Result(ε)

Degeneracies

SoS: simulation of simplicity

Input: n points $p_i = (x_i, y_i), i = 1, \ldots, n$

$\forall i, (x_i, y_i) \ \mapsto \ (x_i, y_i) + \varepsilon^{2^i}(i, i^2)$

Degeneracies

SoS: simulation of simplicity

Input: n points $p_i = (x_i, y_i), i = 1, \ldots, n$

$\forall i, (x_i, y_i) \;\mapsto\; (x_i, y_i) + \varepsilon^{2^i}(i, i^2)$

$$\text{orient}(O, p_i, p_i) \;=\; \text{sign}\begin{vmatrix} x_i & x_j \\ y_i & y_j \end{vmatrix}$$

$$\begin{vmatrix} x_3 & x_1 \\ y_3 & y_1 \end{vmatrix} \mapsto \begin{vmatrix} x_3 + 3\varepsilon^8 & x_1 + \varepsilon^2 \\ y_3 + 9\varepsilon^8 & y_1 + \varepsilon^2 \end{vmatrix} =$$

$$\begin{vmatrix} x_3 & x_1 \\ y_3 & y_1 \end{vmatrix} + \varepsilon^2 \begin{vmatrix} x_3 & 1 \\ y_3 & 1 \end{vmatrix} + \varepsilon^8 \begin{vmatrix} 3 & x_1 \\ 9 & y_1 \end{vmatrix} + \varepsilon^{10} \begin{vmatrix} 3 & 1 \\ 9 & 1 \end{vmatrix}$$

Degeneracies

SoS: simulation of simplicity

Input: n points $p_i = (x_i, y_i), i = 1, \ldots, n$

$\forall i, (x_i, y_i) \ \mapsto \ (x_i, y_i) + \varepsilon^{2^i}(i, i^2)$

$$\text{orient}(O, p_i, p_i) \ = \ \text{sign} \begin{vmatrix} x_i & x_j \\ y_i & y_j \end{vmatrix}$$

non-null polynomial

$$\begin{vmatrix} x_3 & x_1 \\ y_3 & y_1 \end{vmatrix} \mapsto \begin{vmatrix} x_3 + 3\varepsilon^8 & x_1 + \varepsilon^2 \\ y_3 + 9\varepsilon^8 & y_1 + \varepsilon^2 \end{vmatrix} =$$

$$\begin{vmatrix} x_3 & x_1 \\ y_3 & y_1 \end{vmatrix} + \ \varepsilon^2 \begin{vmatrix} x_3 & 1 \\ y_3 & 1 \end{vmatrix} + \ \varepsilon^8 \begin{vmatrix} 3 & x_1 \\ 9 & y_1 \end{vmatrix} + \ \varepsilon^{10} \begin{vmatrix} 3 & 1 \\ 9 & 1 \end{vmatrix}$$

sign $=$ sign of first non-null coefficient

Degeneracies

SoS: simulation of simplicity

Input: n points $p_i = (x_i, y_i), i = 1, \ldots, n$

$\forall i, (x_i, y_i) \;\mapsto\; (x_i, y_i) + \varepsilon^{2^i}(i, i^2)$

$$\text{orient}(O, p_i, p_i) \;=\; \text{sign}\begin{vmatrix} x_i & x_j \\ y_i & y_j \end{vmatrix}$$

$\longrightarrow$ always > 0 or < 0

same for in_disk

Degeneracies

SoS: simulation of simplicity

Input: n points $p_i = (x_i, y_i), i = 1, \ldots, n$

$\forall i, (x_i, y_i) \; \mapsto \; (x_i, y_i) + \varepsilon^{2^i}(i, i^2)$

$$\text{orient}(O, p_i, p_i) \; = \; \text{sign}\begin{vmatrix} x_i & x_j \\ y_i & y_j \end{vmatrix}$$

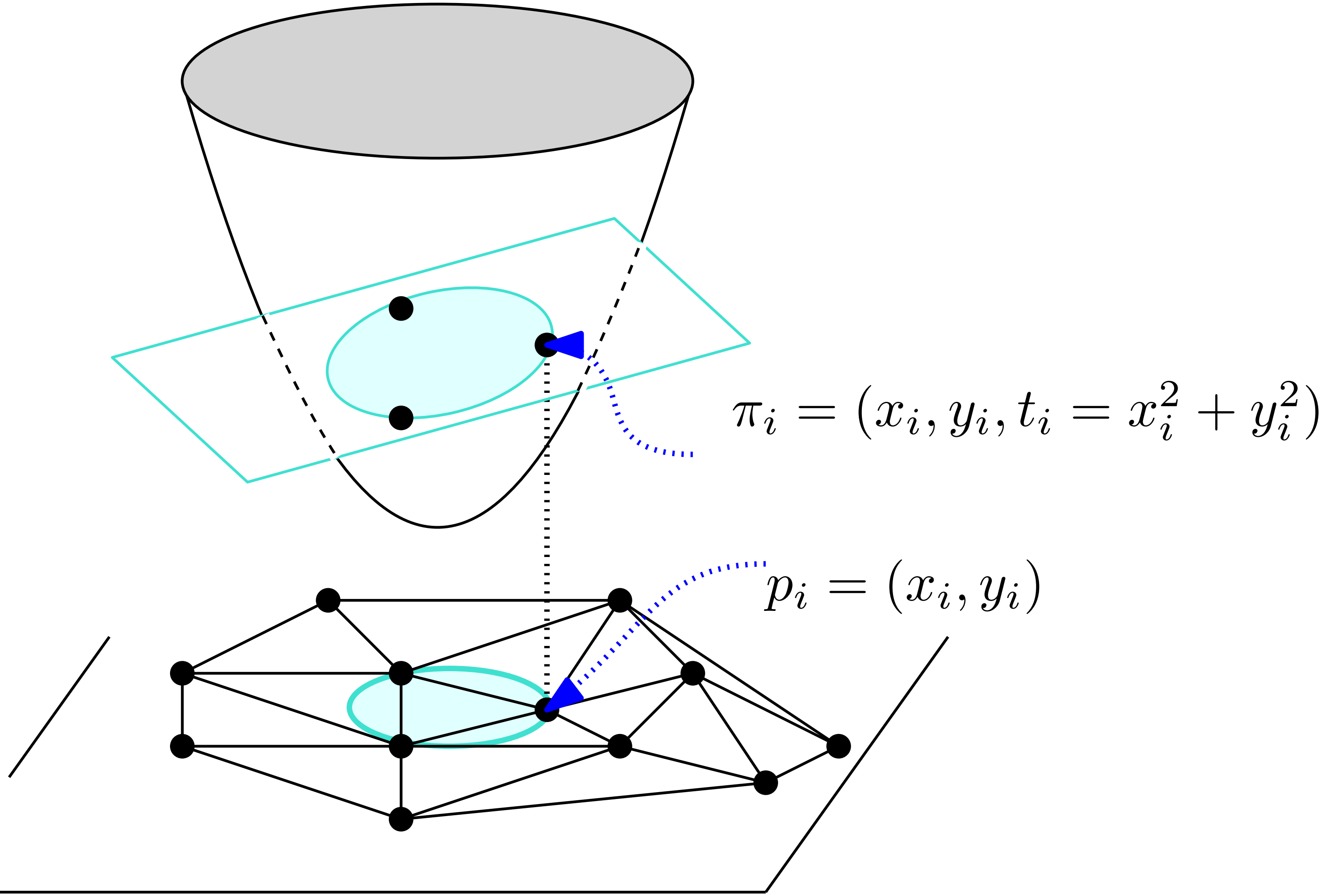
$\longrightarrow$ always > 0 or < 0

same for in_disk

may create FLAT simplices

Degeneracies

Perturbing points in $d + 1^{\mathrm{th}}$ dimension

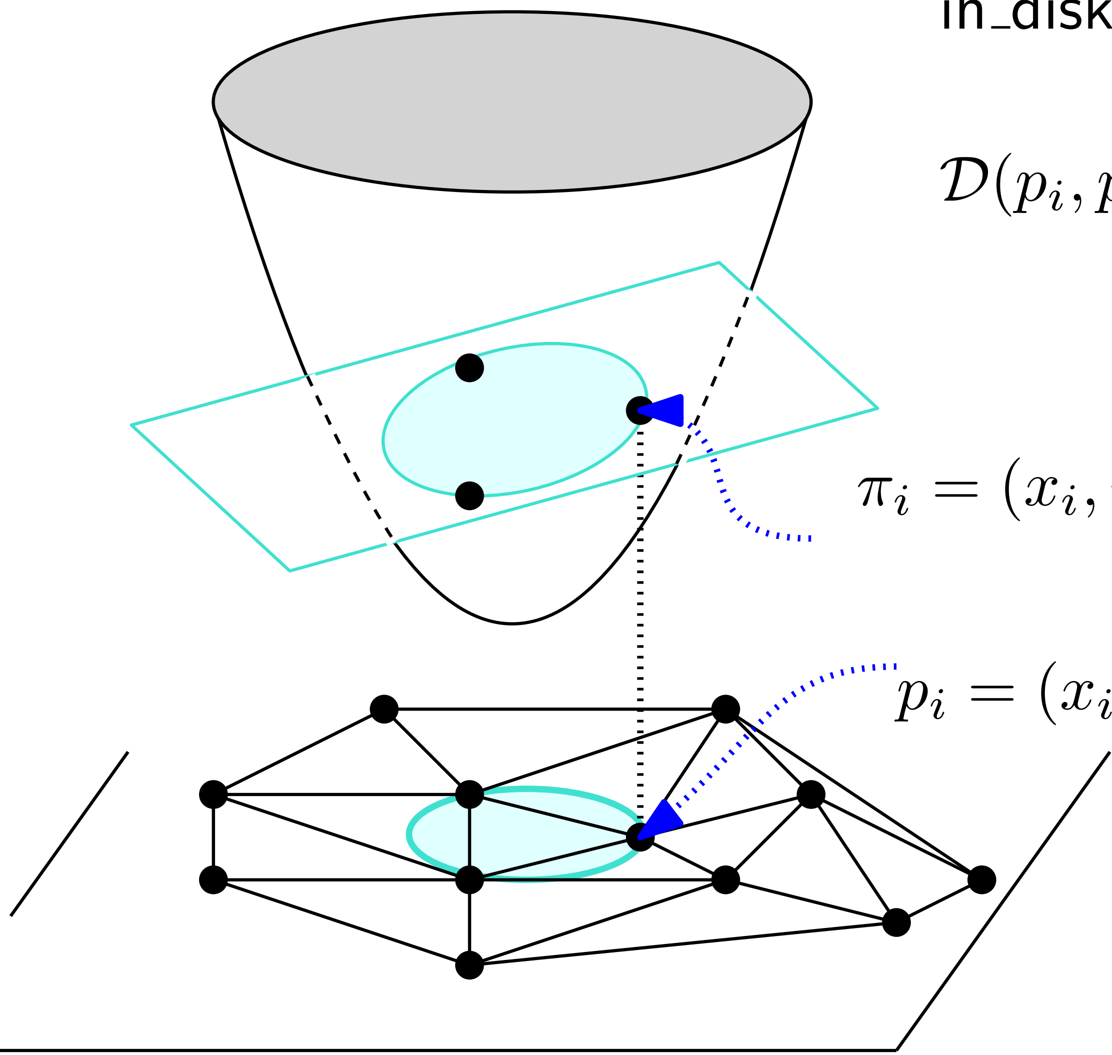


Degeneracies

Perturbing points in $d+1^{\text{th}}$ dimension

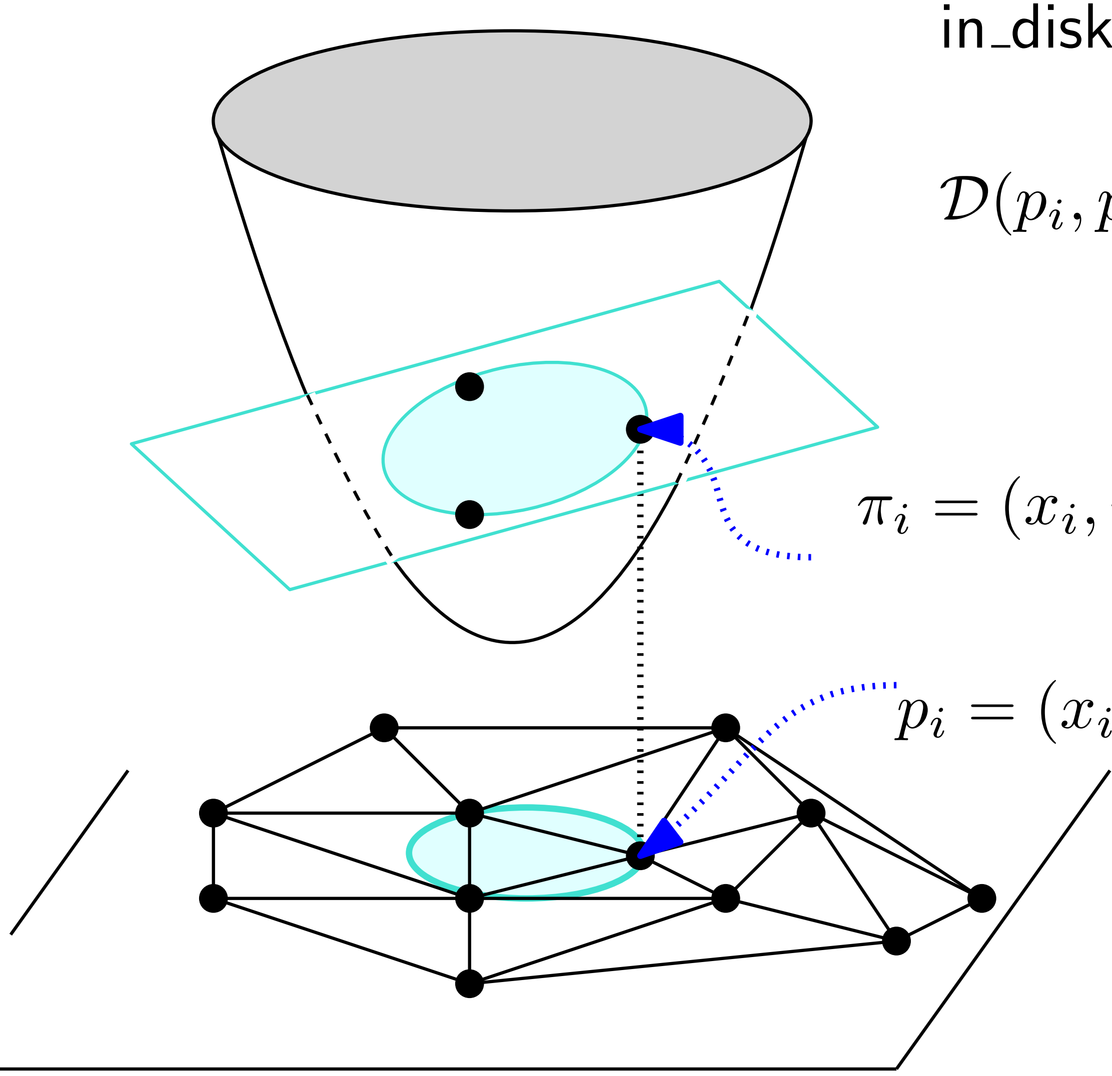
$$\text{in\_disk}(p_i, p_j, p_k, p_l) = \frac{\mathcal{D}(p_i, p_j, p_k, p_l)}{\text{orient}(p_i, p_j, p_k)}$$

$$\mathcal{D}(p_i, p_j, p_k, p_l) = \text{orient}(\pi_i, \pi_j, \pi_k, \pi_l)$$



Degeneracies

Perturbing points in $d+1^{\text{th}}$ dimension



$$\mathsf{in\_disk}(p_i, p_j, p_k, p_l) = \frac{\mathcal{D}(p_i, p_j, p_k, p_l)}{\mathsf{orient}(p_i, p_j, p_k)}$$

$$\mathcal{D}(p_i, p_j, p_k, p_l) = \mathsf{orient}(\pi_i, \pi_j, \pi_k, \pi_l)$$

$$\mapsto \mathsf{orient}(\pi_i^\varepsilon, \pi_j^\varepsilon, \pi_k^\varepsilon, \pi_l^\varepsilon)$$

$$\pi_i = (x_i, y_i, t_i = x_i^2 + y_i^2)$$

$$\mapsto \pi_i^\varepsilon = (x_i, y_i, t_i + \varepsilon^{n-i})$$

$$p_i = (x_i, y_i)$$

Degeneracies

Perturbing points in $d+1^{\text{th}}$ dimension

$$\text{orient}(\pi_i^\varepsilon,\pi_j^\varepsilon,\pi_k^\varepsilon,\pi_l^\varepsilon) = \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_i & x_j & x_k & x_l \\ y_i & y_j & y_k & y_l \\ z_i & z_j & z_k & z_l \\ t_i+\varepsilon^{n-i} & t_j+\varepsilon^{n-j} & t_k+\varepsilon^{n-k} & t_l+\varepsilon^{n-l} \end{vmatrix}$$

$$\begin{aligned} &= \mathcal{D}(p_i,p_j,p_k,p_l) \\ &\qquad -\text{orient}(p_i,p_j,p_k)\varepsilon^{n-l} \\ &\qquad +\text{orient}(p_i,p_j,p_l)\varepsilon^{n-k} \\ &\qquad -\text{orient}(p_i,p_k,p_l)\varepsilon^{n-j} \\ &\qquad +\text{orient}(p_j,p_k,p_l)\varepsilon^{n-i} \end{aligned}$$

4 cocircular points $\longrightarrow$ non-null polynomial in ε

point with highest index
in the disk of the other 3

Degeneracies

Perturbing points in $d+1^{\mathrm{th}}$ dimension

orientation predicate not perturbed

$\Longrightarrow$ NO flat simplex created

global indexing $=$ lexicographic order

$\longrightarrow$ Delaunay triangulation uniquely defined

easy to implement

CGAL

Computational Geometry Algorithms Library

www.cgal.org

open source

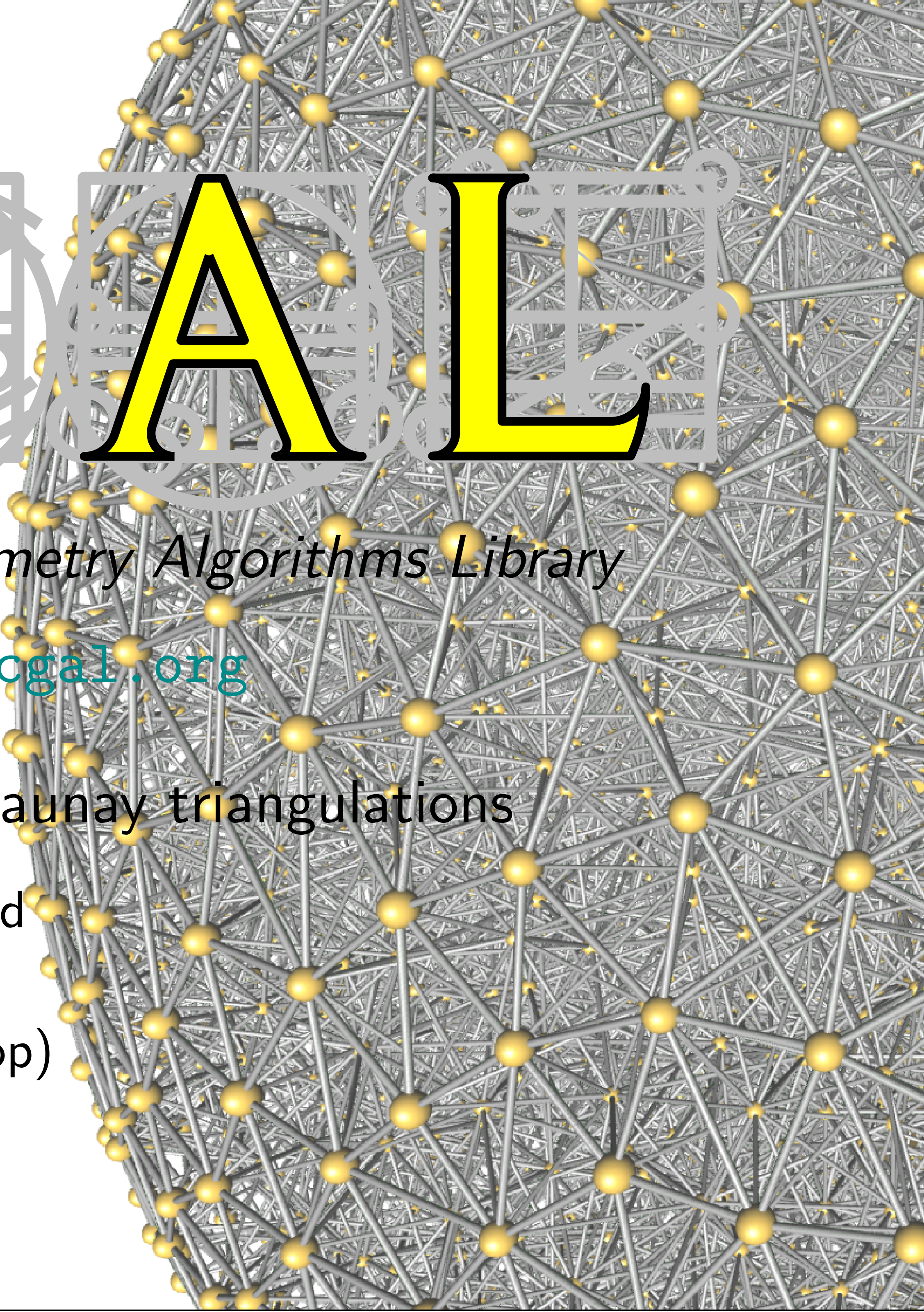
distributed under GPL

commercial licences distributed by GeometryFactory

CGAL

Computational Geometry Algorithms Library

www.cgal.org



2D, 3D, dD [weighted] Delaunay triangulations

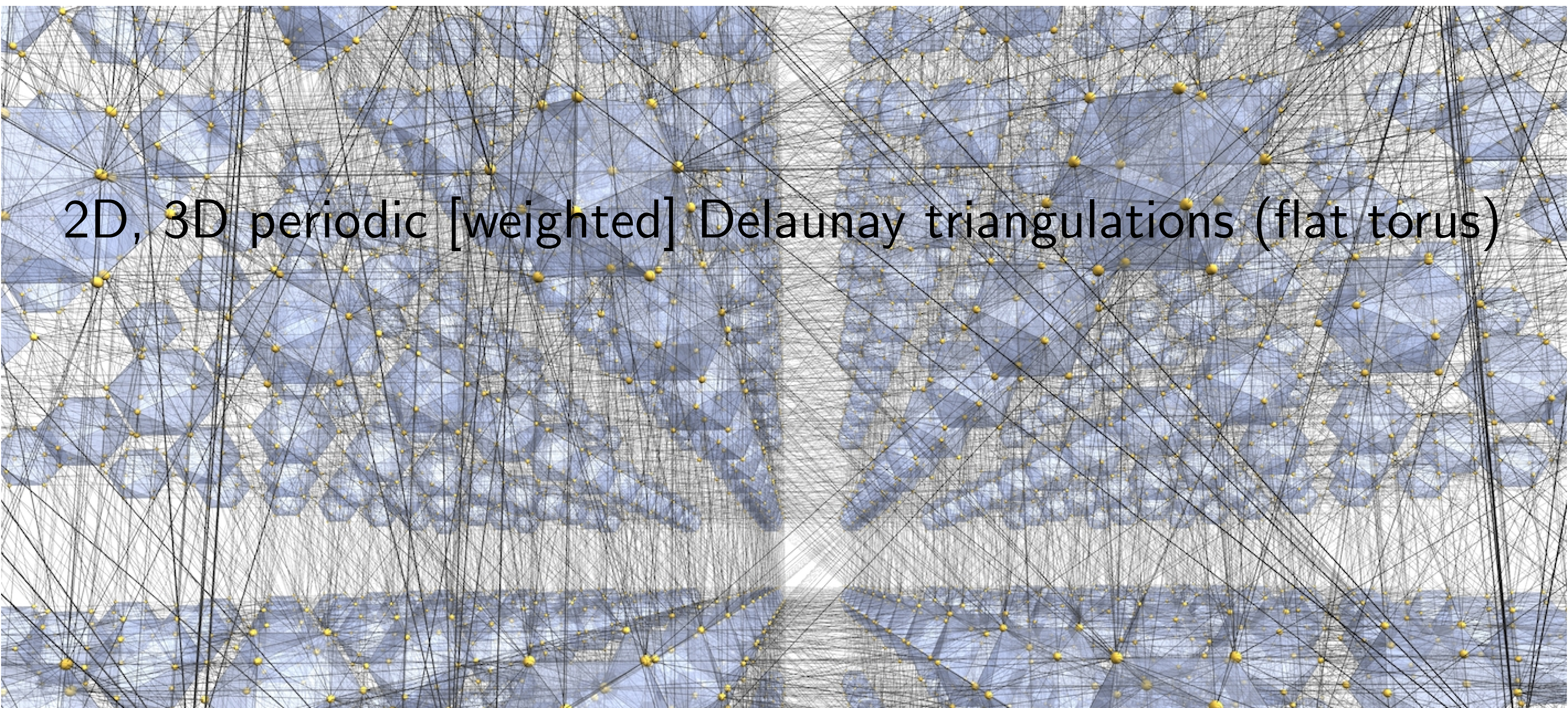
2D $\simeq$ 10 million points / second
3D $\simeq$ 1 million points / second
(on a standard laptop)

fully dynamic

fully robust

CGAL

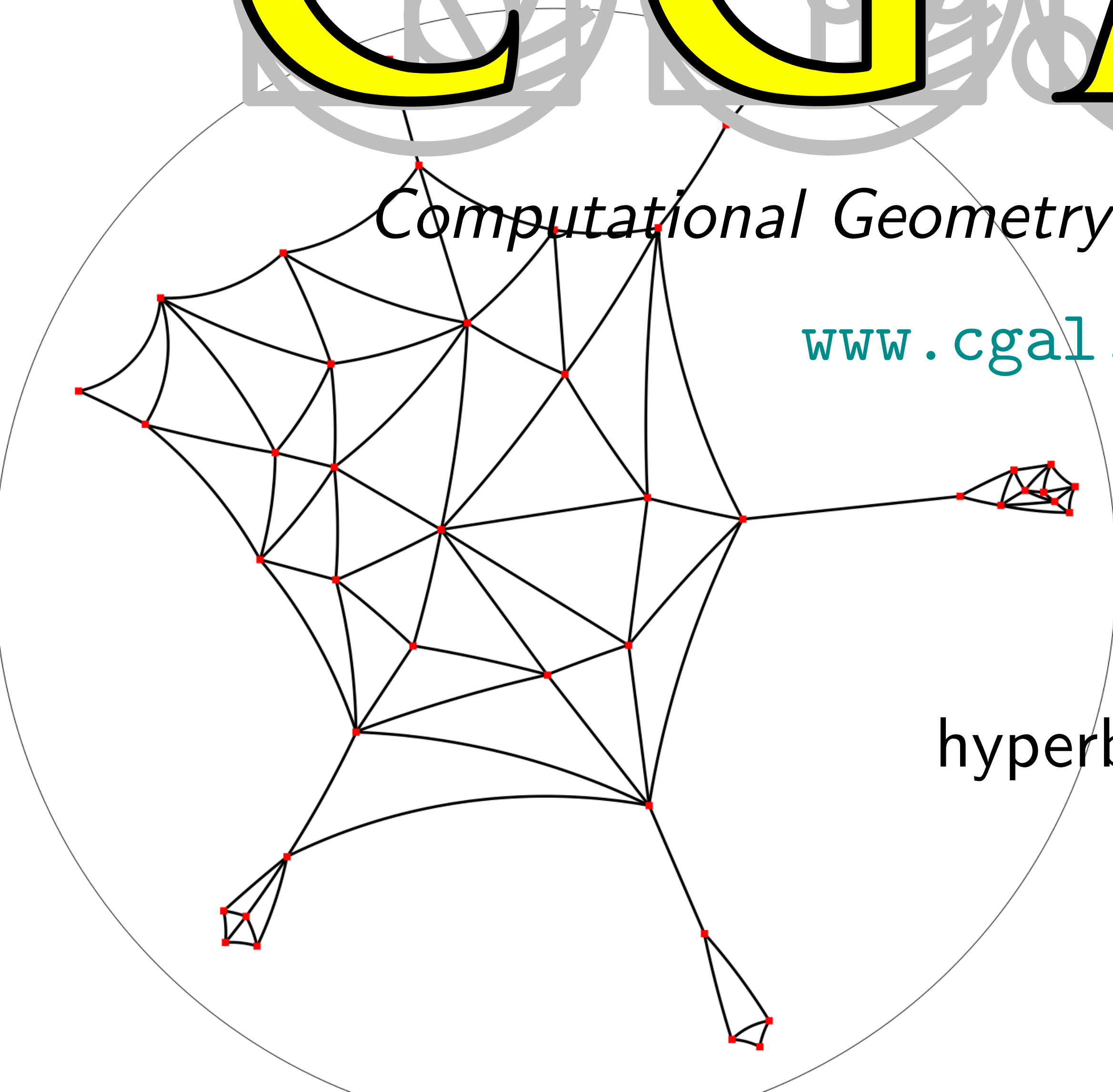
2D, 3D periodic [weighted] Delaunay triangulations (flat torus)



CGAL

Computational Geometry Algorithms Library

www.cgal.org



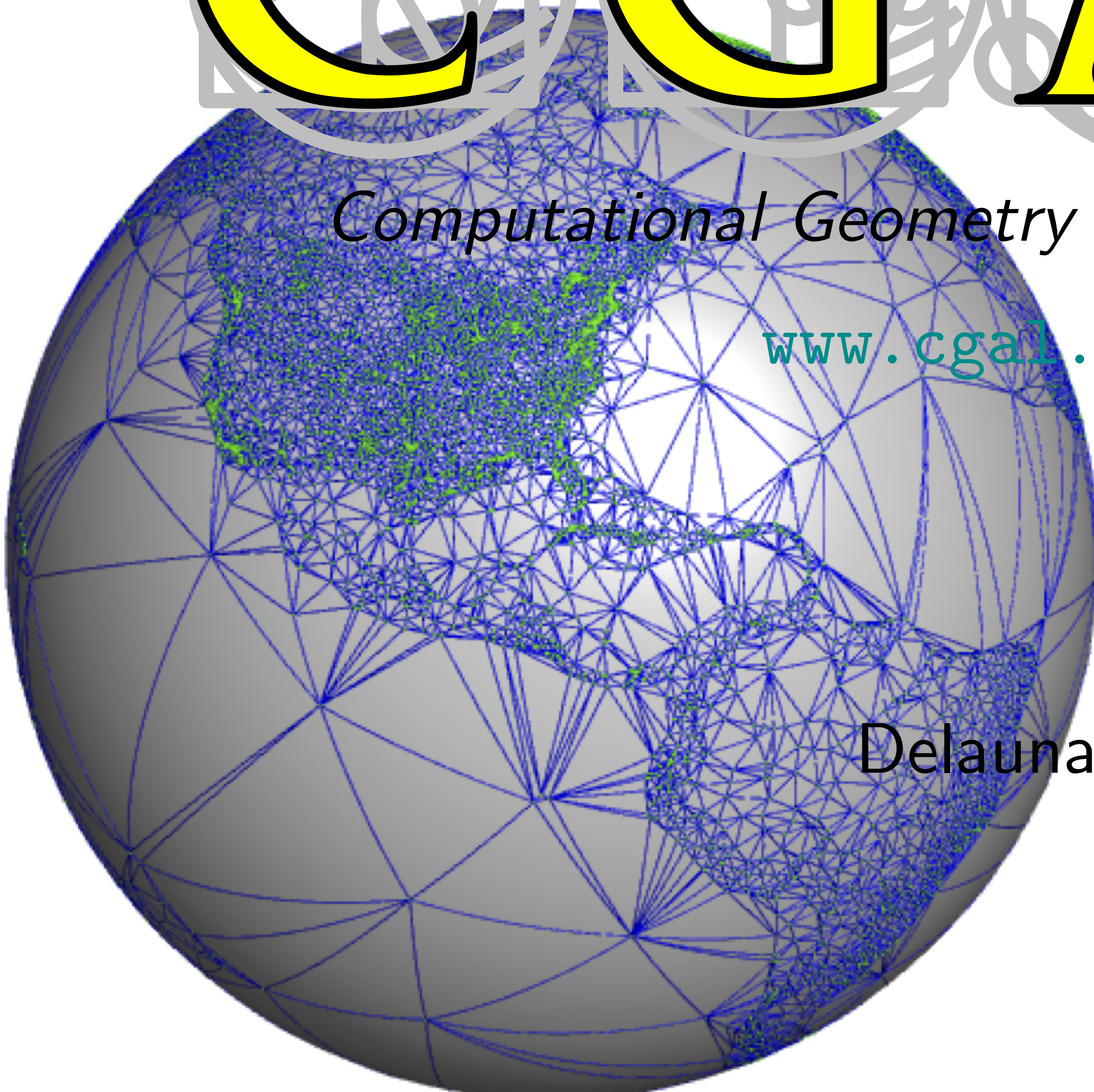
soon (?)

hyperbolic Delaunay triangulations

CGAL

Computational Geometry Algorithms Library

www.cgal.org



soon (?)

Delaunay triangulations on the sphere

CGAL

Computational Geometry Algorithms Library

www.cgal.org

used by astrophysicists, biologists, . . .

CGAL

Computational Geometry Algorithms Library

`www.cgal.org`

used by astrophysicists, biologists, mathematician(s?) ...

Take home (?)

There is a long way from the algorithm to the software

Needed

clean mathematical models

good algorithms

knowledge of computers

union makes strength