Précision numérique

et incohérence géométrique

Olivier Devillers

INRIA SOPHIA ANTIPOLIS

FIABLE

Computational Geometry

Theoretical hypotheses

Algorithms

Analysis

Computational Geometry

Theoretical hypotheses

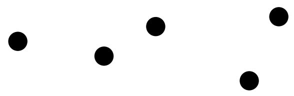
Algorithms

Real RAM
General position

Analysis

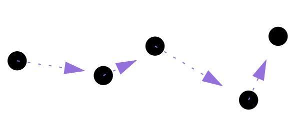
Real RAM
Asymptotic complexity

Combinatorial nature

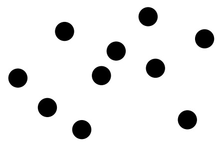


Sorting

Combinatorial nature

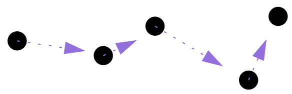


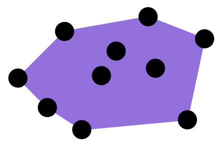
Sorting

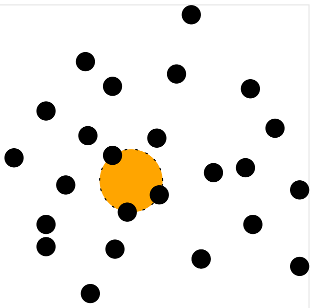


Convex hull

Combinatorial nature

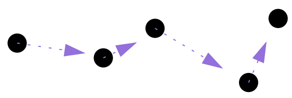


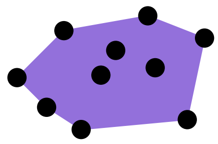


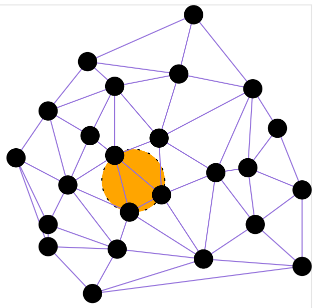


Sorting **Convex hull** **Delaunay**

Combinatorial nature



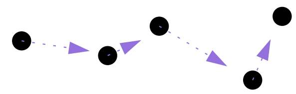


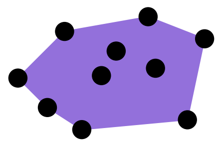


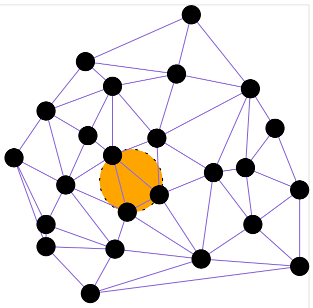
Sorting **Convex hull** **Delaunay**

Numerical nature

Combinatorial nature

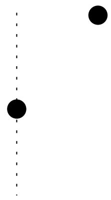






Sorting **Convex hull** **Delaunay**

Comparison

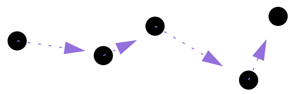


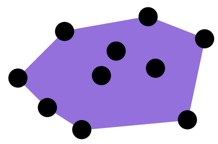
Predicates

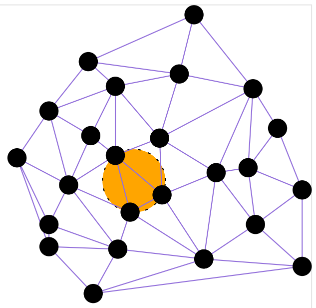
Numerical nature

INRIA

Combinatorial nature

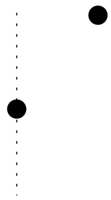


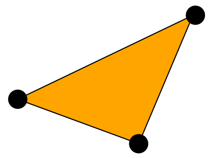




Sorting **Convex hull** **Delaunay**

Comparison *Which side*

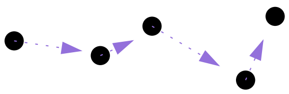


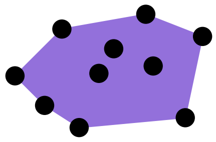


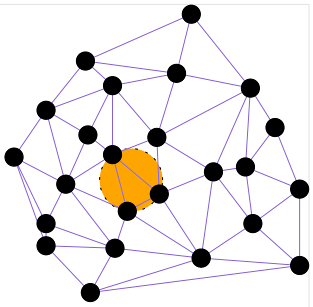
Predicates

Numerical nature

Combinatorial nature

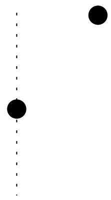


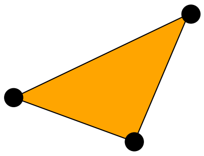


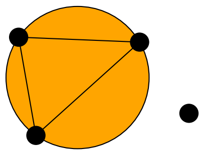


Sorting **Convex hull** **Delaunay**

Comparison *Which side* *In circle*





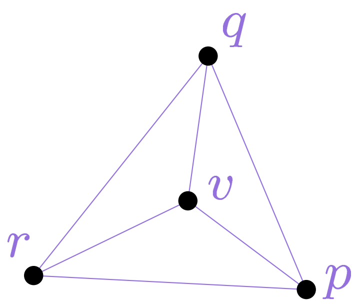


Predicates

Numerical nature

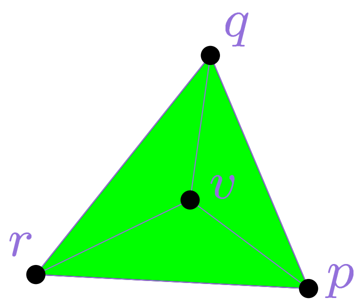
Algorithms rely on
geometric theorems

Algorithms rely on geometric theorems



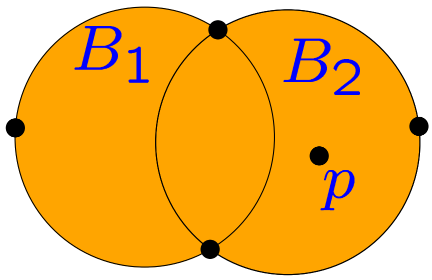
$$\begin{matrix} pqv \\ qrv \\ rpv \end{matrix} \quad \text{CCW} \qquad \Longrightarrow$$

Algorithms rely on geometric theorems



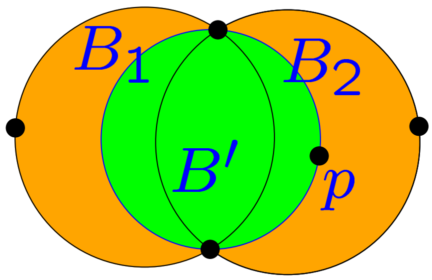
$$\begin{matrix} pqv \\ qrv \\ rpv \end{matrix} \text{ CCW} \quad \Longrightarrow \quad \boxed{pqr \text{ CCW}}$$

Algorithms rely on geometric theorems



$$\begin{array}{l} p \in B_2 \\ p \notin B_1 \end{array} \qquad \Longrightarrow$$

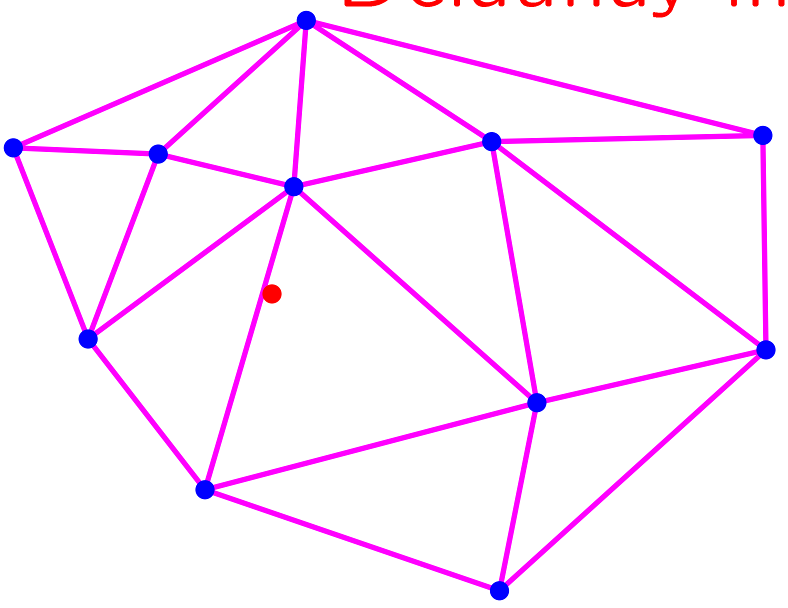
Algorithms rely on geometric theorems



$$p \in B_2 \quad p \notin B_1 \quad \Longrightarrow \quad \boxed{B' \subset B_1 \cup B_2}$$

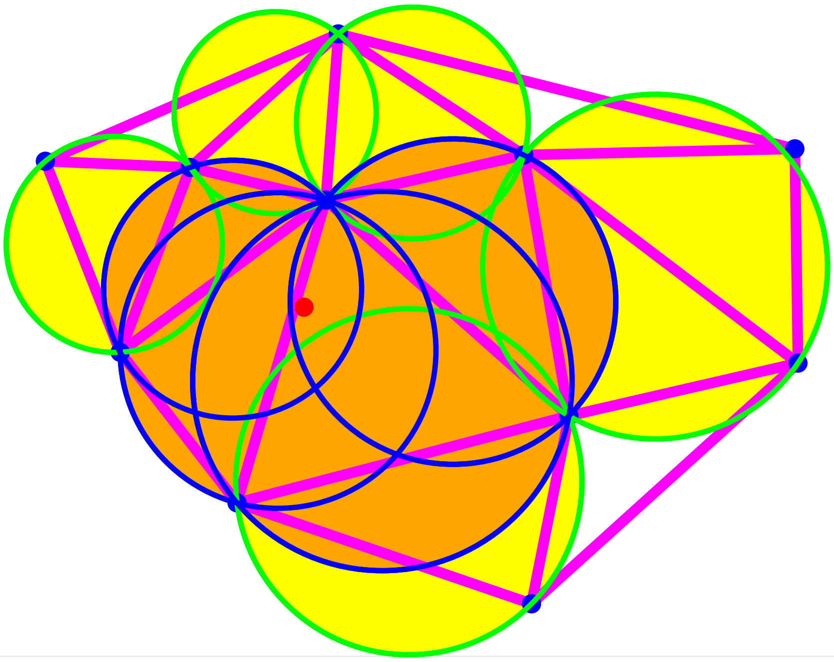
Algorithms rely on geometric theorems

Delaunay insertion

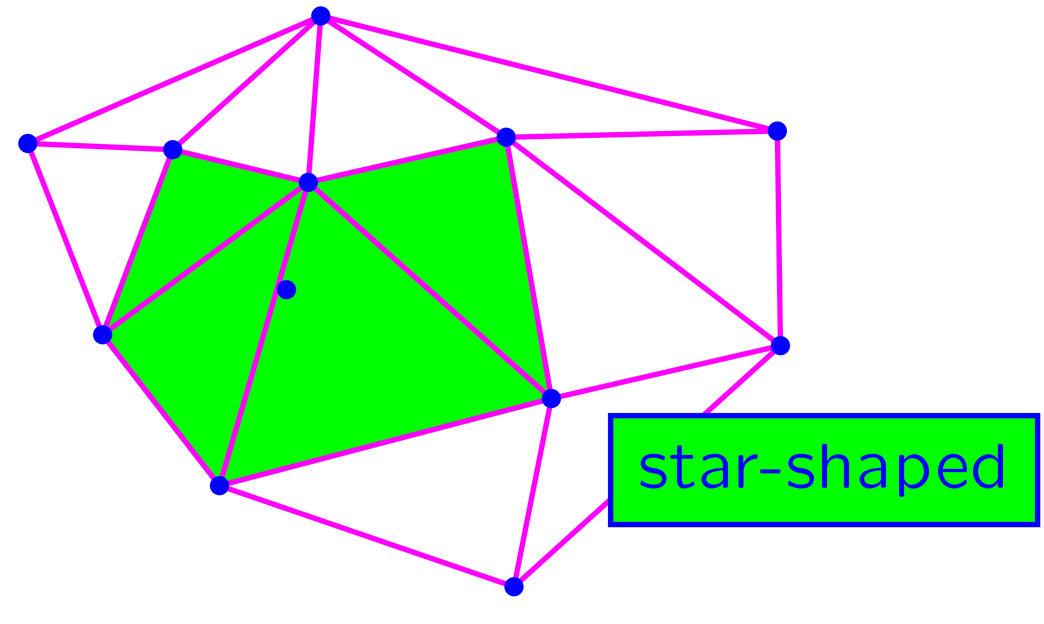


INRIA SOPHIA ANTIPOLIS

Algorithms rely on geometric theorems

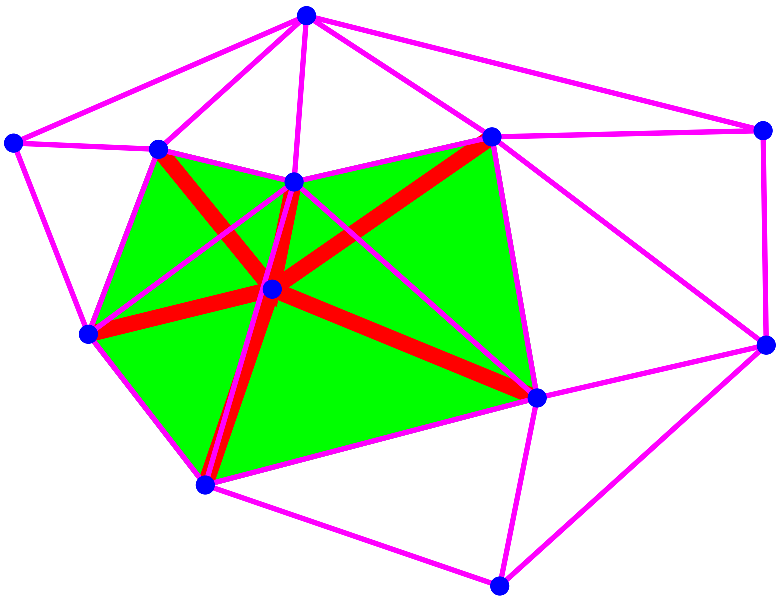


Algorithms rely on
geometric theorems



INRIA SOPHIA ANTIPOLIS

Algorithms rely on geometric theorems



Floating point arithmetic

rounded computations

IEEE 754

24 binary digits
53 binary digits

Toy model

2 decimal digits

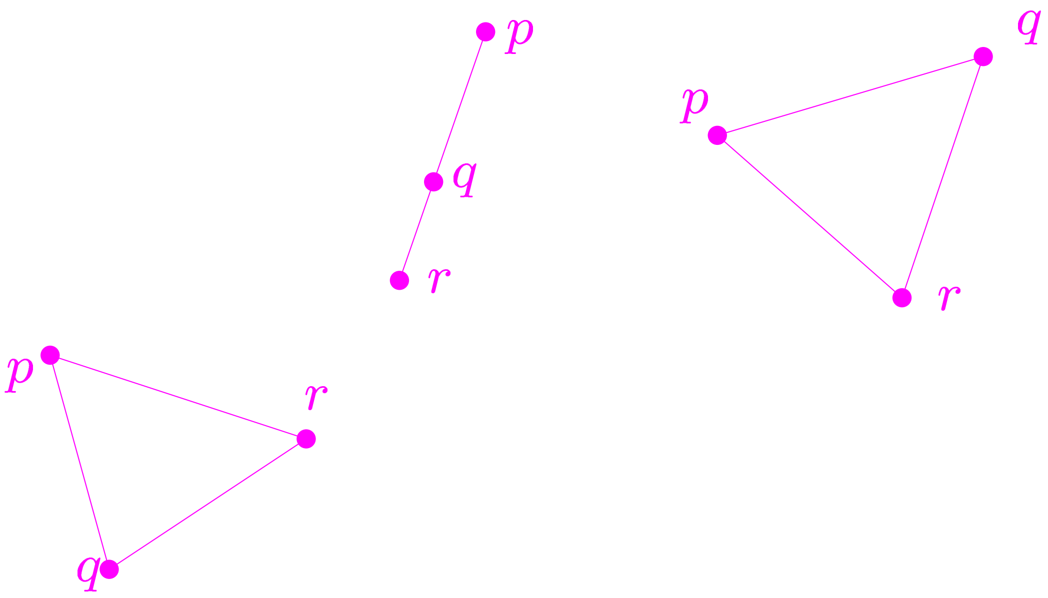
$$97 + 8.2 = 105.2$$
$$97 + 8.2 \to 110$$

errors invalidate
geometric theorems

INRIA

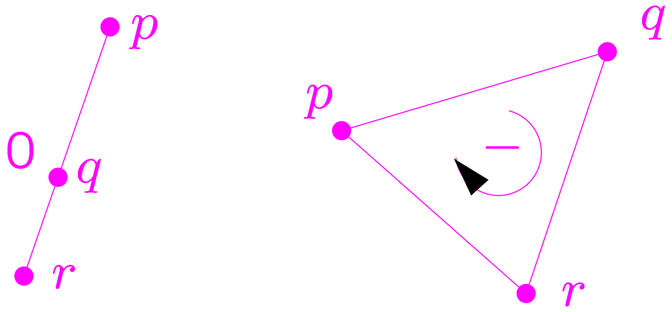
Floating point arithmetic

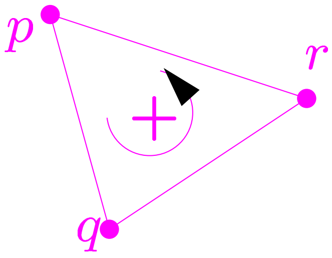
Orientation test



Floating point arithmetic

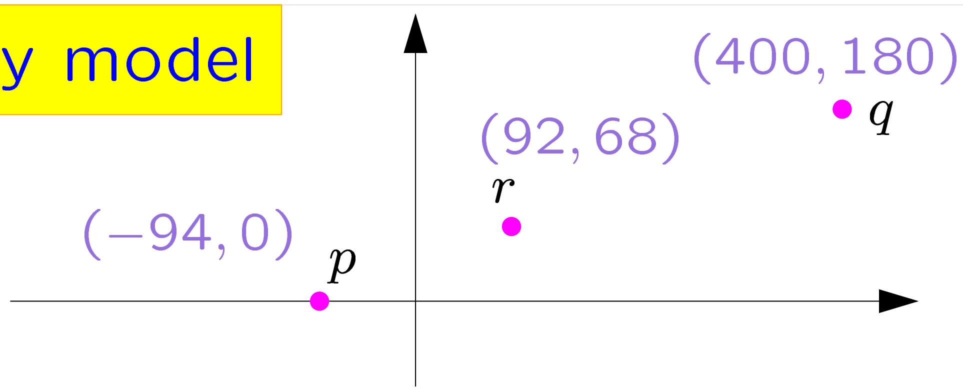
Orientation test





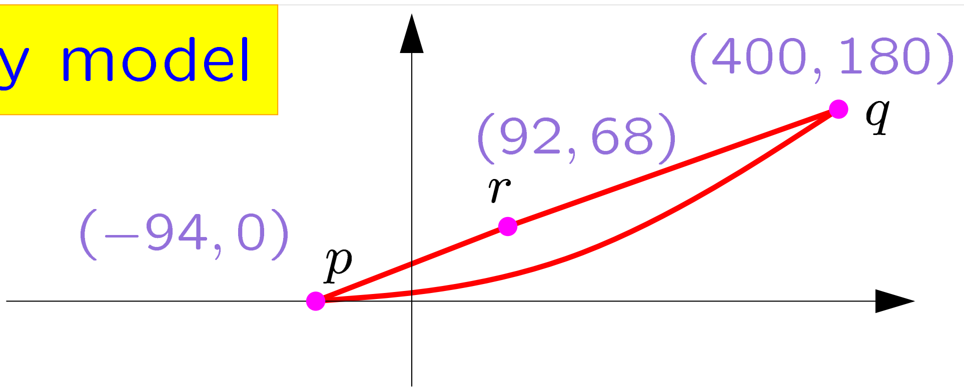
$$\begin{vmatrix} x_q - x_p & x_r - x_p \\ y_q - y_p & y_r - y_p \end{vmatrix}$$

Toy model



Exact computation

Toy model



$$(400 + 94) * 68 - (92 + 94) * 180 =$$

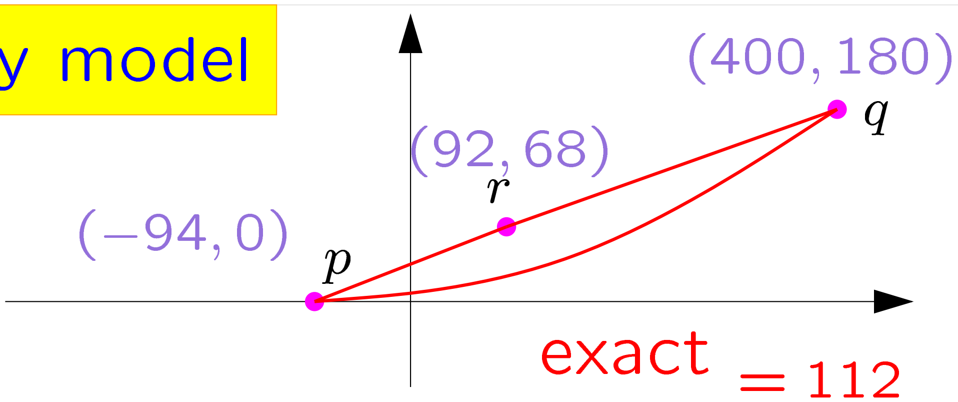
$$494 * 68 - 186 * 180 \ = 33592 - 33480$$

$$= 112$$

Exact computation

INRIA

Toy model

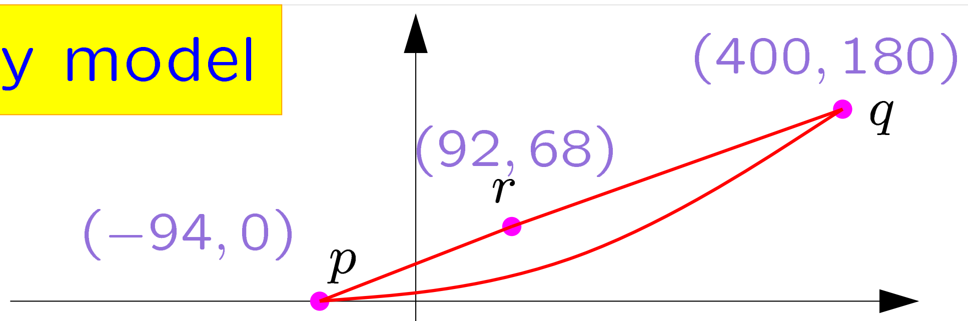


$$\text{exact} = 112$$

$$(400 + 94) * 68 - (92 + 94) * 180 =$$

Floating point computation

Toy model

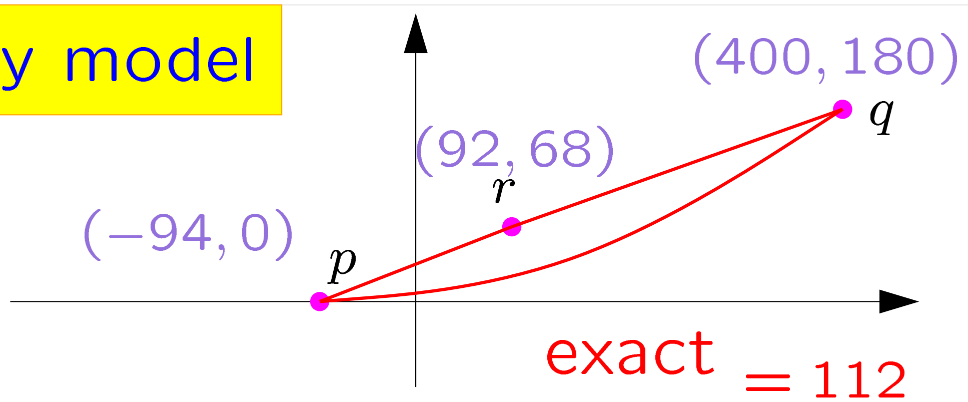


exact $= 112$

$$(400+94)*68-(92+94)*180=$$
$$494*68-186*180$$

Floating point computation

Toy model



exact $= 112$

$(400 + 94) * 68 - (92 + 94) * 180 =$

$494 * 68 - 186 * 180$

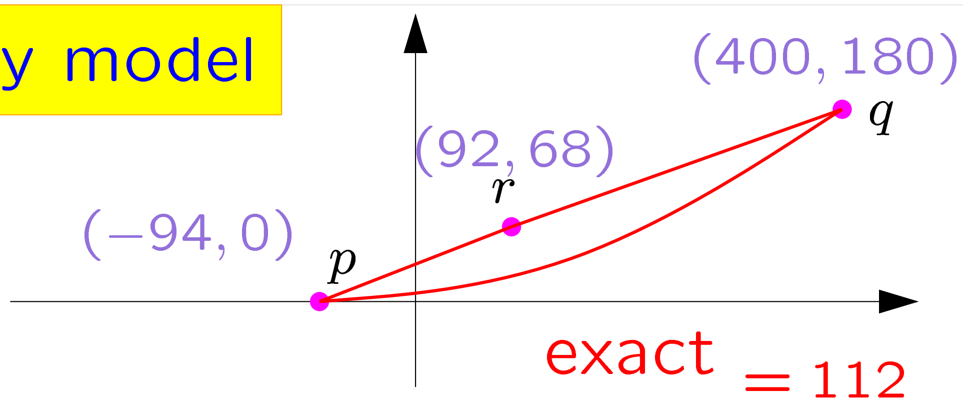
$490 * 68 - 190 * 180$

$= 33320 - 34200$

Floating point computation

INRIA

Toy model



exact $= 112$

$$(400+94)*68-(92+94)*180=$$

$$494*68-186*180$$

$$490*68-190*180$$

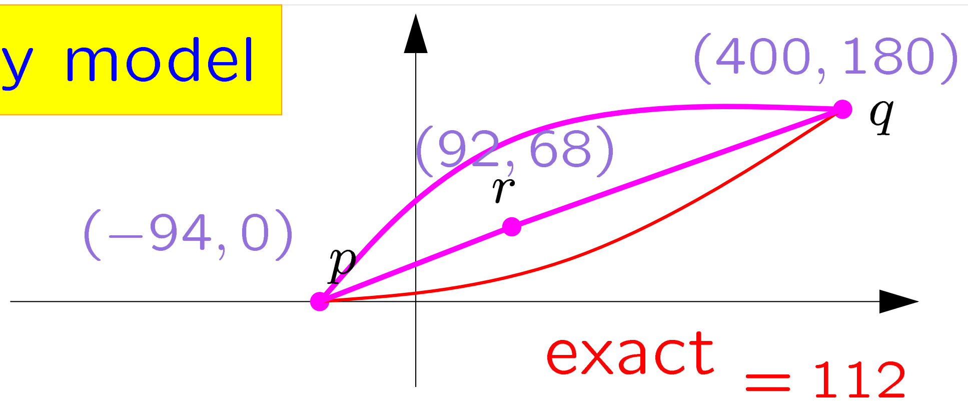
$$=33320-34200$$

$$=33000-34000$$

Floating point computation

INRIA

Toy model



exact $= 112$

$$(400 + 94) * 68 - (92 + 94) * 180 =$$

$$494 * 68 - 186 * 180$$

$$490 * 68 - 190 * 180$$

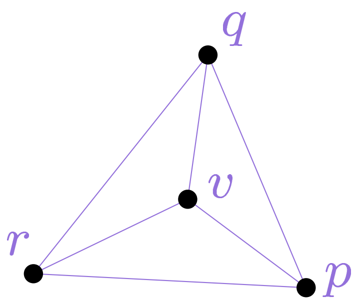
$$= 33320 - 34200$$

$$= 33000 - 34000$$

$$= -1000$$

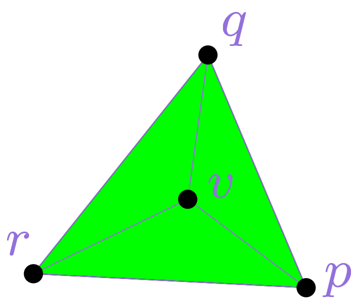
Floating point computation

Inexact arithmetic violates theorems



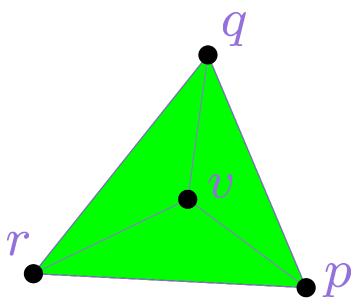
pqv
qrv CCW $\Longrightarrow$
rpv

Inexact arithmetic violates theorems



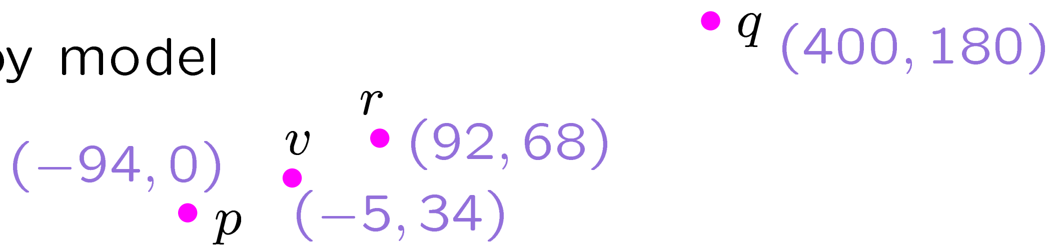
pqv
qrv CCW $\Longrightarrow$ pqr CCW
rpv

Inexact arithmetic violates theorems



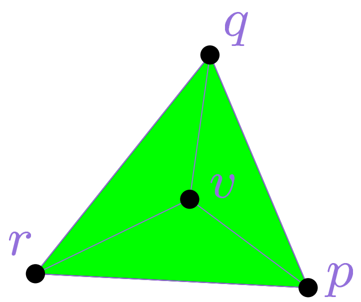
$\begin{matrix} pqv \\ qrv \\ rpv \end{matrix}$ CCW $\Longrightarrow$ pqr CCW

Toy model



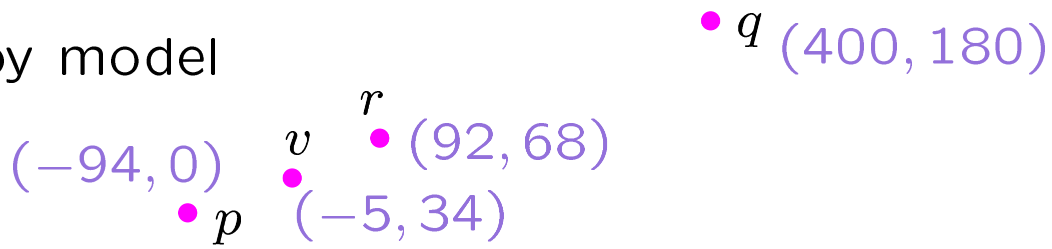
$\begin{matrix} pqv \\ qrv \\ rpv \end{matrix}$ CCW

Inexact arithmetic violates theorems



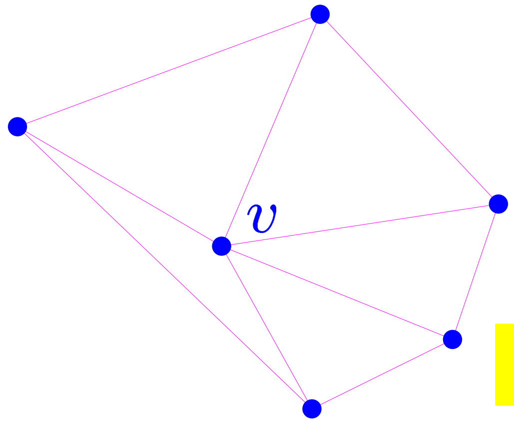
$$\left.\begin{array}{l} pqv \\ qrv \\ rpv \end{array}\right. \text{CCW} \Longrightarrow \boxed{pqr\ \text{CCW}}$$

Toy model



$$\left.\begin{array}{l} pqv \\ qrv \\ rpv \end{array}\right. \text{CCW} \quad \text{and} \quad \boxed{pqr\ \text{CW}}$$

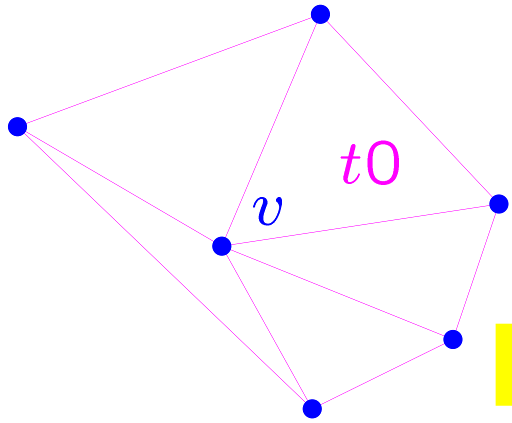
Inexact arithmetic creates bugs



Example :

Turn ccw around v

Inexact arithmetic creates bugs

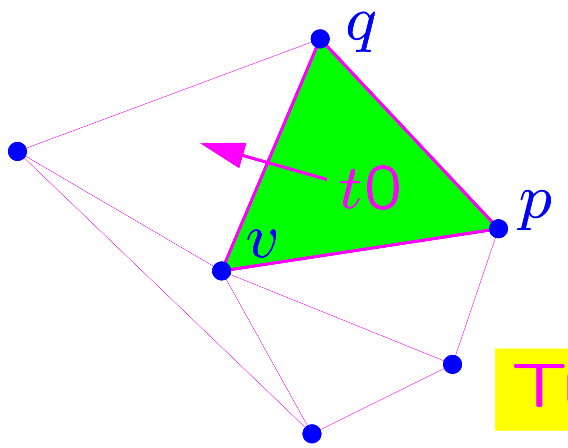


Example :

Turn ccw around v

do
    let vpq ccw triangle
    go to neighbor through qv
while triangle $\neq t0$

Inexact arithmetic creates bugs



Example :

Turn ccw around v

do

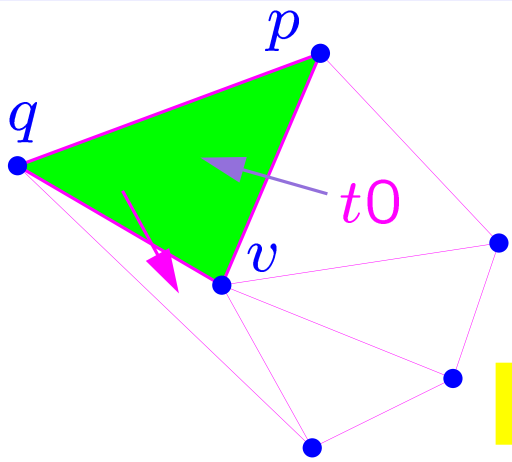
let vpq ccw triangle

go to neighbor through qv

while triangle $\neq t0$

INRIA

Inexact arithmetic creates bugs



Example :

Turn ccw around v

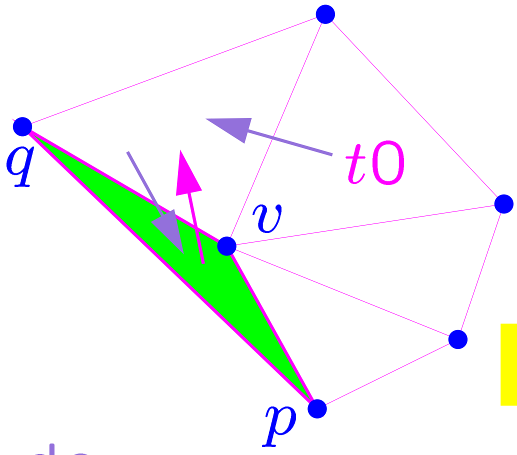
do

  let vpq ccw triangle

  go to neighbor through qv

while triangle $\neq t0$

Inexact arithmetic creates bugs



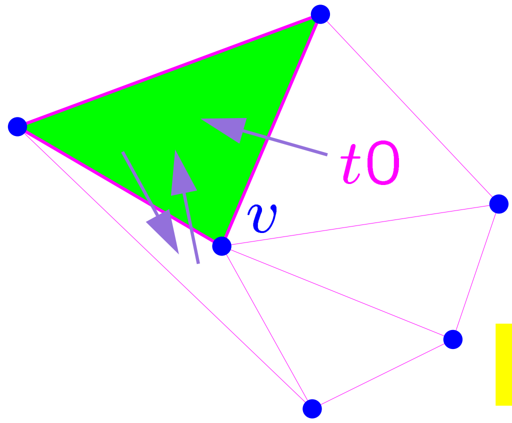
Example :

Turn ccw around v

do
  let vpq ccw triangle
  go to neighbor through qv
while triangle $\neq t0$

INRIA

Inexact arithmetic creates bugs



Example :

Turn ccw around v

do

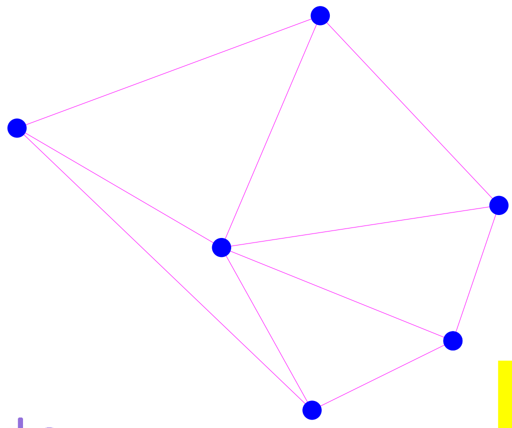
  let vpq ccw triangle

  go to neighbor through qv

while triangle $\neq t0$

Infinite loop

Inexact arithmetic creates bugs

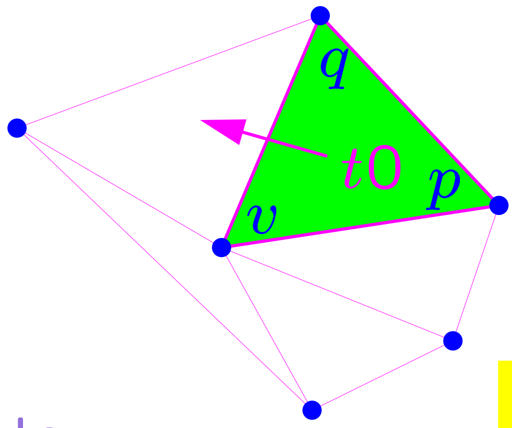


Example :

Turn ccw around v

do
  let vpq ccw triangle
    such that vp incoming edge
  go to neighbor through qv
while triangle $\neq t0$

Inexact arithmetic creates bugs

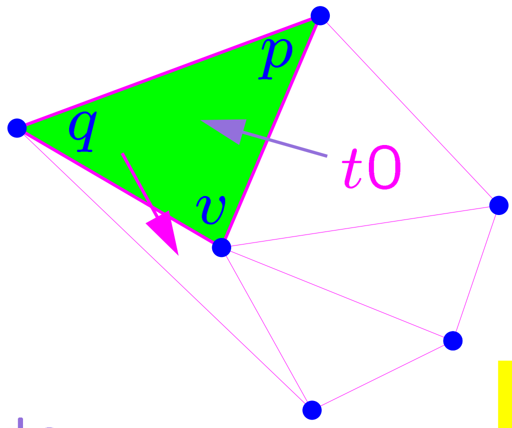


Example :

Turn ccw around v

do
  let vpq ccw triangle
    such that vp incoming edge
  go to neighbor through qv
while triangle $\neq t0$

Inexact arithmetic creates bugs

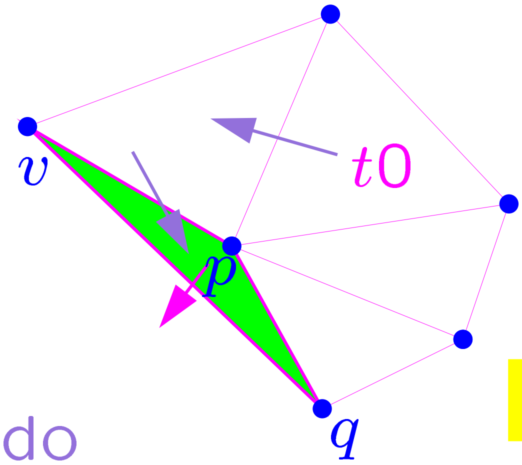


Example :

Turn ccw around v

do
  let vpq ccw triangle
    such that vp incoming edge
  go to neighbor through qv
while triangle $\neq t0$

Inexact arithmetic creates bugs

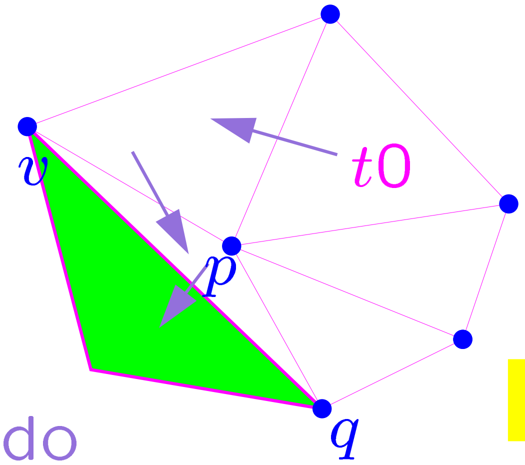


Example :

Turn ccw around v

do
  let vpq ccw triangle
      such that vp incoming edge
  go to neighbor through qv
while triangle $\neq t0$

Inexact arithmetic creates bugs



Example :

Turn ccw around v

Turn ccw around v

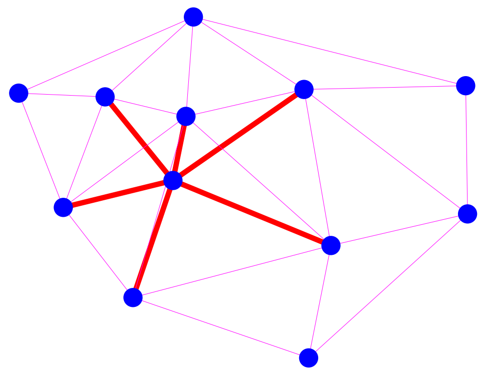
do
  let vpq ccw triangle
      such that vp incoming edge
  go to neighbor through qv
while triangle $\neq t0$

Infinite loop ?
core dumped ?

Solution 1

Do not rely on theorems

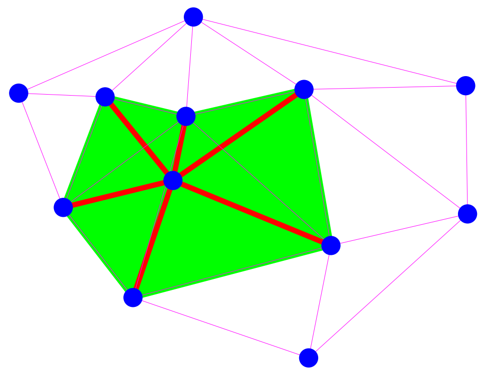
Example



Solution 1

Do not rely on theorems

Example

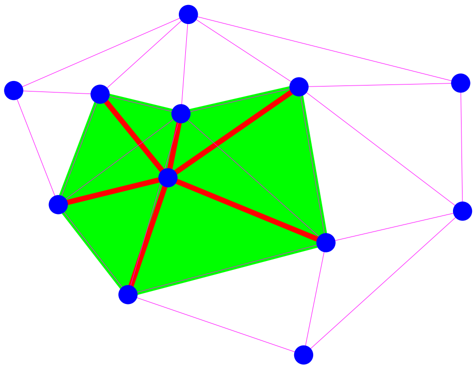


star-shaped

Solution 1

Do not rely on theorems

Example



star-shaped

weaker version

simple polygon

INRIA

simple polygon

purely topological

Special algorithm

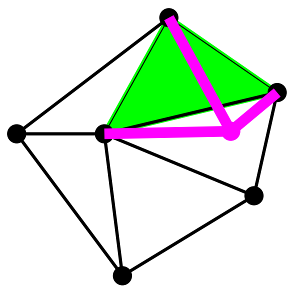
Sugihara & Iri

simple polygon

purely topological

Special algorithm

Sugihara & Iri

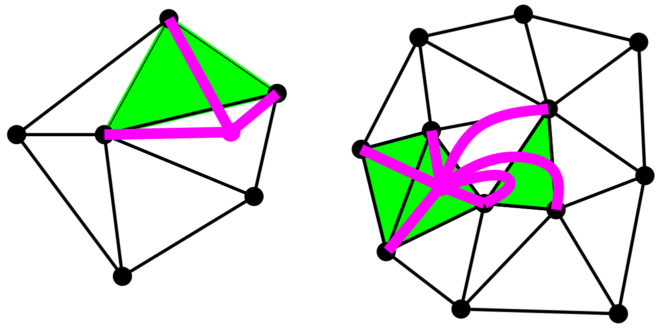


simple polygon

purely topological

Special algorithm

Sugihara & Iri

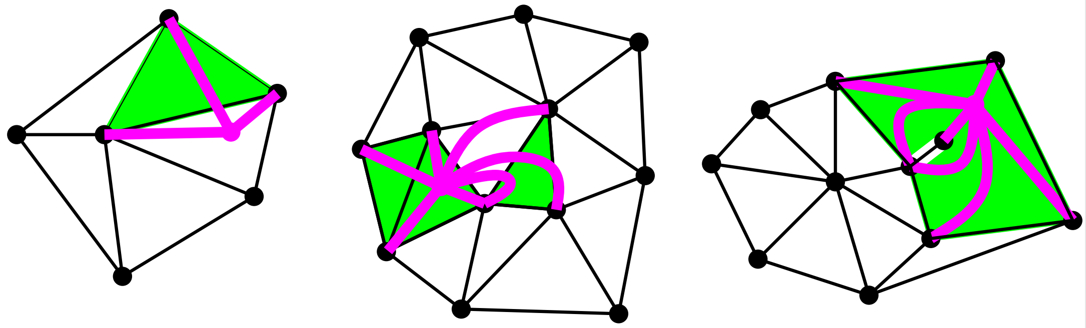


simple polygon

purely topological

Special algorithm

Sugihara & Iri

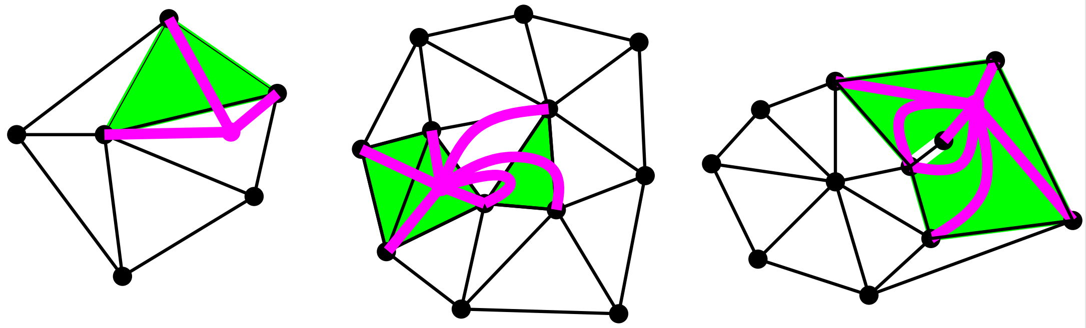


simple polygon

purely topological

Special algorithm

Sugihara & Iri



topological triangulation

robust

Weaker properties

Weaker properties

Non geometrically correct

non Delaunay triangulation
non planar triangulation

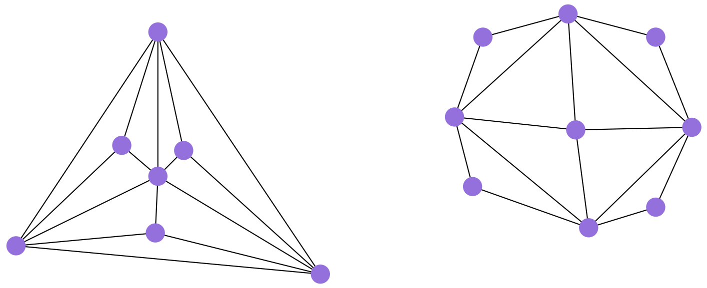
Weaker properties

Non geometrically correct

non Delaunay triangulation
non planar triangulation

even Non topologically correct

non realizable as Delaunay



Solution 1

Do not rely on theorems

not much used

Solution 1

Do not rely on theorems

not much used

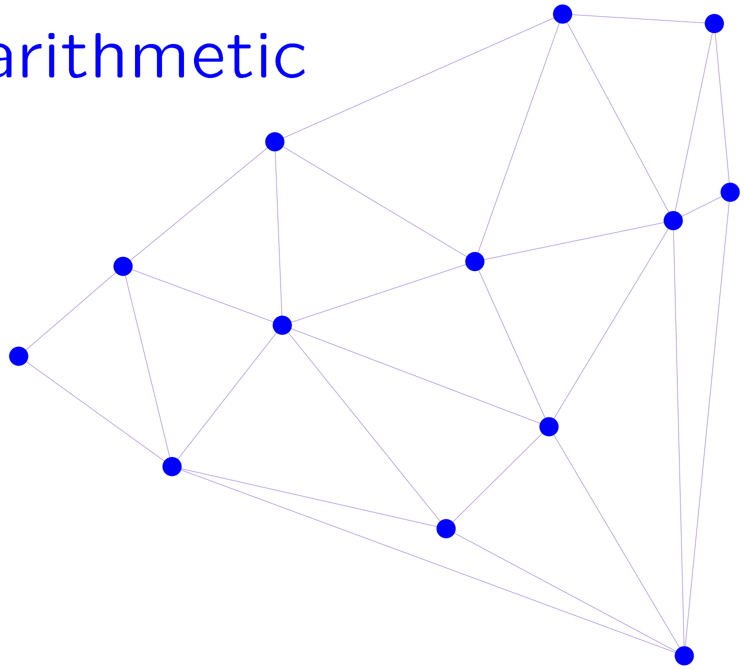
Solution 2

Use exact computations

Use exact computations

Exact data

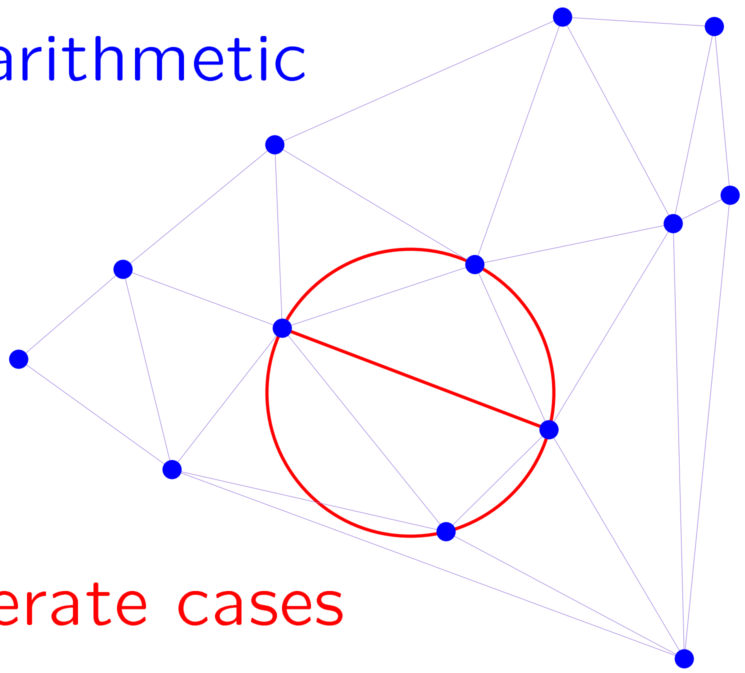
Exact arithmetic



Use exact computations

Exact data

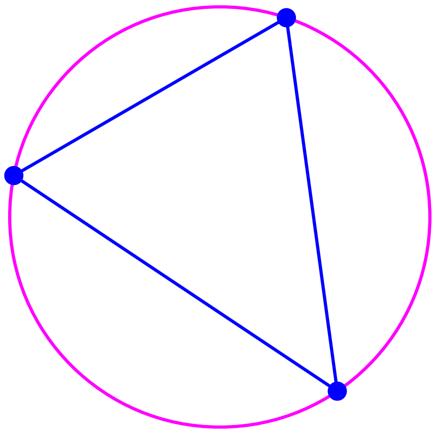
Exact arithmetic



Degenerate cases

INRIA

constructions
versus
predicates

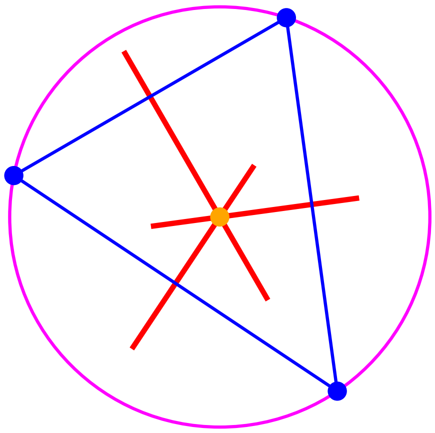


constructions
versus
predicates

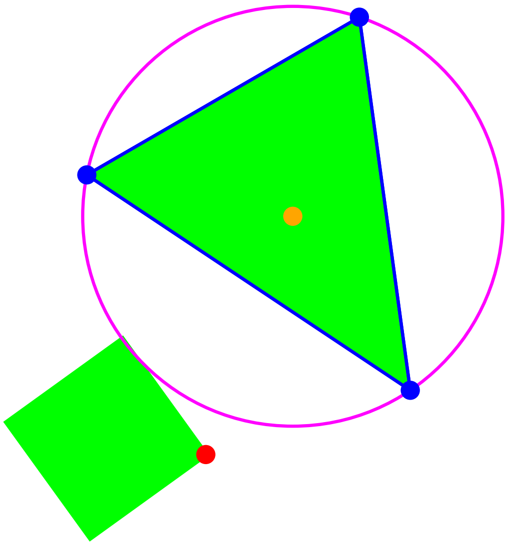
circumcenter

exact

rational



constructions
versus
predicates



circumcenter

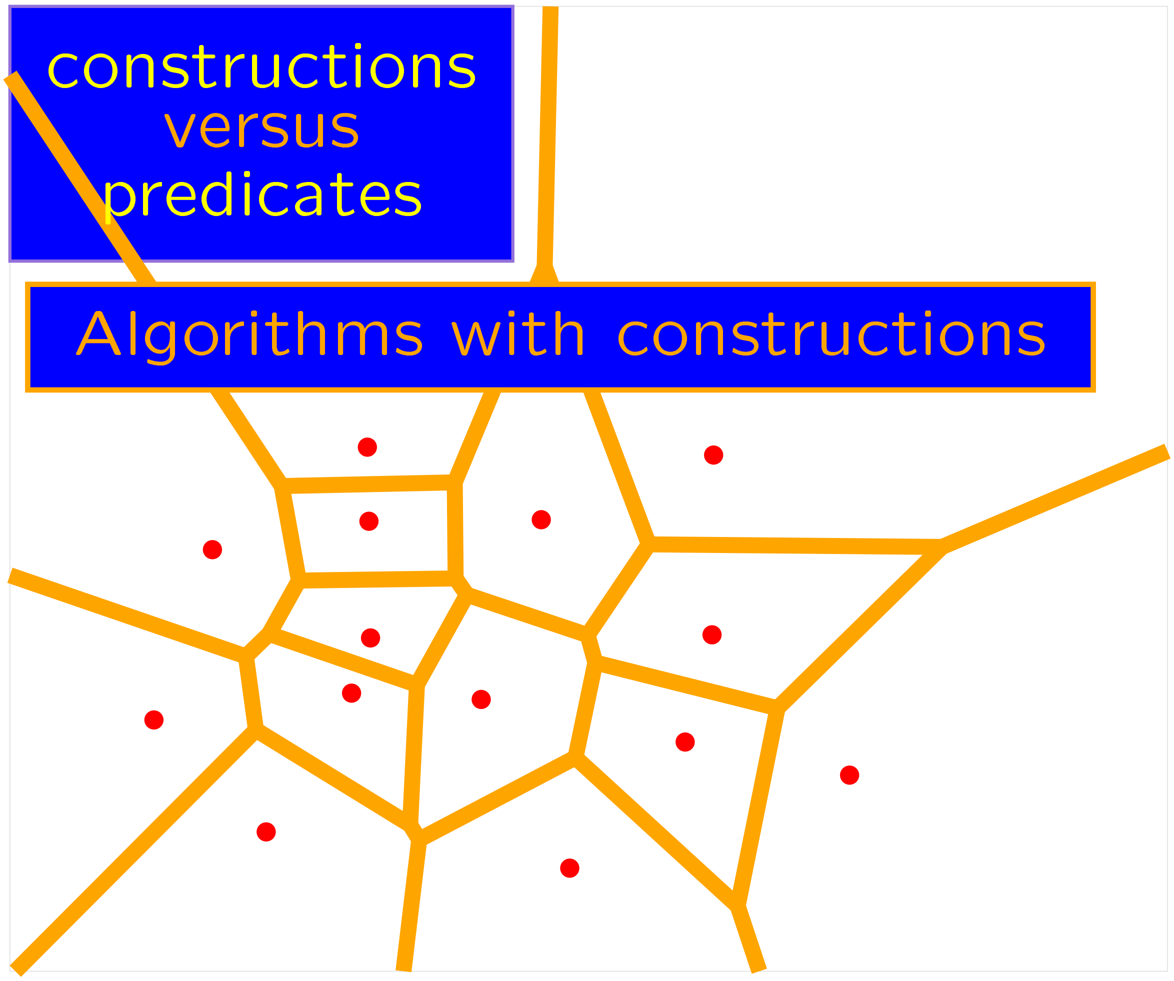
exact

rational

in circle test

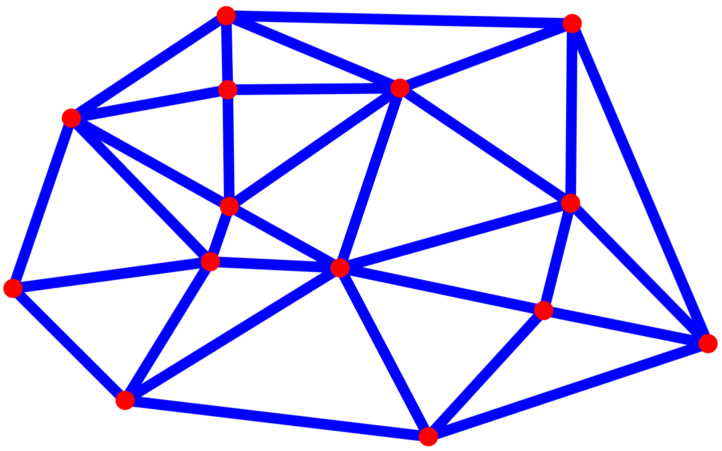
exact

sign of polynomial


constructions
versus
predicates
Algorithms with constructions

constructions
versus
predicates

Predicates only algorithms



Geometric predicate

Filter design

Geometric predicate

sign of polynomial

exact computation

floating point

Geometric predicate

sign of polynomial

exact computation *precision=degree*

→ non constant time

floating point *approximation*

→ sometimes false

Geometric predicate

sign of polynomial

exact computation *precision=degree*
→ non constant time

complexity pb

floating point *approximation*
→ sometimes false

unacceptable

Geometric predicate

sign of polynomial

Exact predicate

$\neq$

Exact arithmetic

Exact sign

$\neq$

Exact value

Given an arithmetic predicate

$\boxed{P(x) > 0 \text{ ?}}$

A filter is

An approximate evaluator

A certifier

An exact evaluator

Approximate evaluator

rounded computation

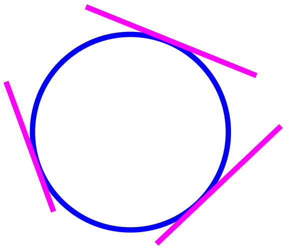
IEEE 754

other

$\boxed{P(x) > 0\ ?}$

geometric approximation

bounding box



Approximate evaluator

rounded computation

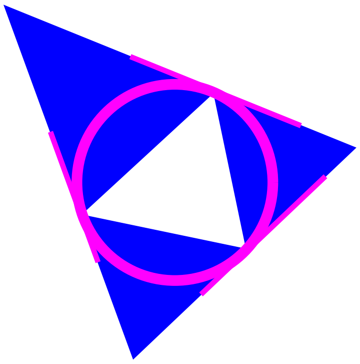
IEEE 754

other

$P(x) > 0$?

geometric approximation

bounding box



O. Beaumont

Certifier

$P(x) > 0$?

error computation

computed $P(x)$

$$\overline{P(x)} = P(x) + \epsilon(x)$$

Certifier

error computation

static

semi-static

dynamic

Certifier

error computation | static

$P(x) > 0$?

Hypotheses on entries

$|x| \leq M$

Restricted operations

P with no division

$$\boxed{|\overline{P(x)} - P(x)| \leq \epsilon(M)}$$

Certifier

error computation	static

Avantages

- Fast
- good success rate

Drawbacks

- P restricted
- Hypotheses on data
- only good success rate

Certifier

error computation | dynamic

$P(x) > 0$?

Error computation at execution

For each arithmetic operation

$$|\overline{P(x)} - P(x)| \leq P_\epsilon(x)$$

Certifier

error computation	dynamic

Avantages

- Simple to use
- Excellent success rate

Drawbacks

- Slower

Certifier

error computation

semi-static

$\boxed{P(x) > 0\ ?}$

Intermediate solutions

$$\boxed{|\overline{P(x)} - P(x)| \leq \epsilon(x)}$$

with ϵ simpler than f_ϵ

Exact evaluator

If filter fails

Exact arithmetic

Depends on data

Integer (GMP)

Float expensions [Priest-Shewchuk]

CGAL real

An example in CGAL

Sylvain Pion

An example in CGAL

The predicate

template <class FT>
OrientationC2
(FT x0,y0,x1,y1,x2,y2)
{
return sign $\left(\begin{vmatrix} x1 - x0 & x2 - x0 \\ y1 - y0 & y2 - y0 \end{vmatrix} \right)$
}

INRIA

An example in CGAL

OrientationC2<double>

wrong rounded computation

An example in CGAL

OrientationC2<double>

wrong rounded computation

OrientationC2<Gmpz>

expensive exact computation

(data are integers)

An example in CGAL

OrientationC2<Interval>

Compute $\overline{P(x)}$ and $P_\epsilon(x)$

Uses Interval arithmetic

An example in CGAL

OrientationC2<Special-nt>

Compute $\epsilon : M \rightarrow \epsilon(M)$

Uses special number type
for error computation

An example in CGAL

Dynamic filter

OrientationC2
<FilteredNT<int,Gmpz>>

data are int

use interval arithmetic

if filter failure

exact computation using Gmpz

An example in CGAL

Static filter

OrientationC2
<FilteredNT<int,Gmpz>>

data are int

Upper bound known

Compute once the static error

use rounded arithmetic

if filter failure

exact computation using Gmpz

An example in CGAL

Semi-static filter

OrientationC2
<FilteredNT<int,Gmpz>>

data are int

If data $<$ Upper bound
use rounded arithmetic
else or if filter failure
Compute upper bound
and static error
use rounded arithmetic
if filter failure
exact computation using Gmpz

Filter efficiency

Filter efficiency

Sign of polynomial P

If $P(x)$ is small filters fail

Filter efficiency

Sign of polynomial P

If $P(x)$ is small **filters fail**

Probability $P(x) \leq \epsilon$?

Data distribution ?

INRIA SOPHIA ANTIPOLIS

Filter efficiency

Sign of polynomial P

If $P(x)$ is small — filters fail

Probability $P(x) \leq \epsilon$?

Data distribution ?

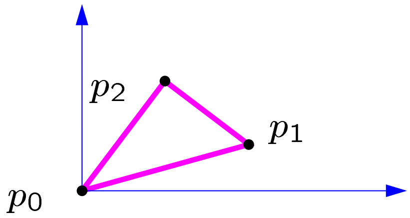
Data distribution

Predicate

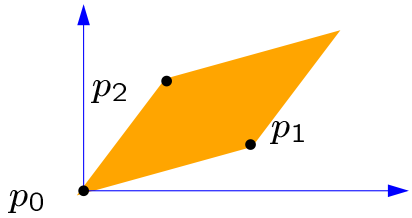
Algorithm

INRIA

$P(x)$	Orientation predicate

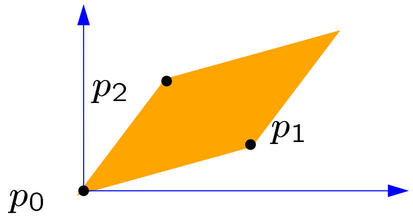


$P(x)$	Orientation predicate

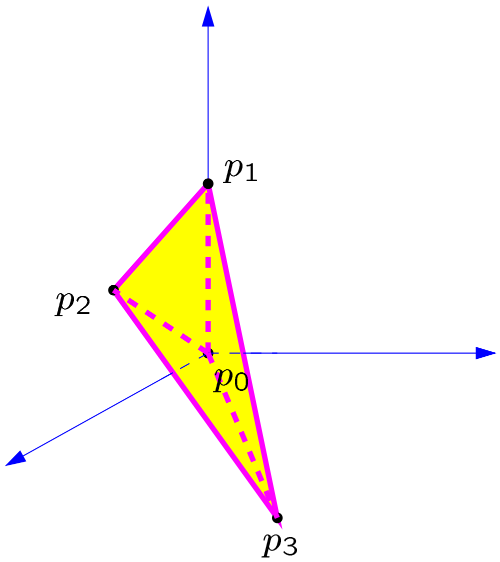


Area

$P(x)$	Orientation predicate

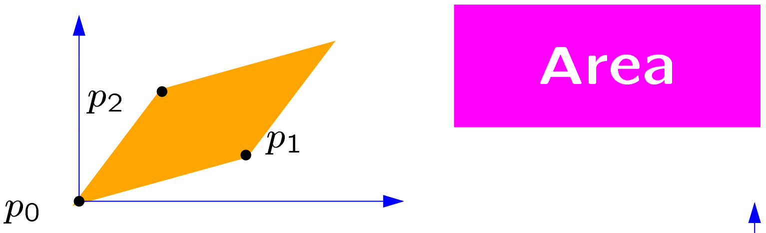


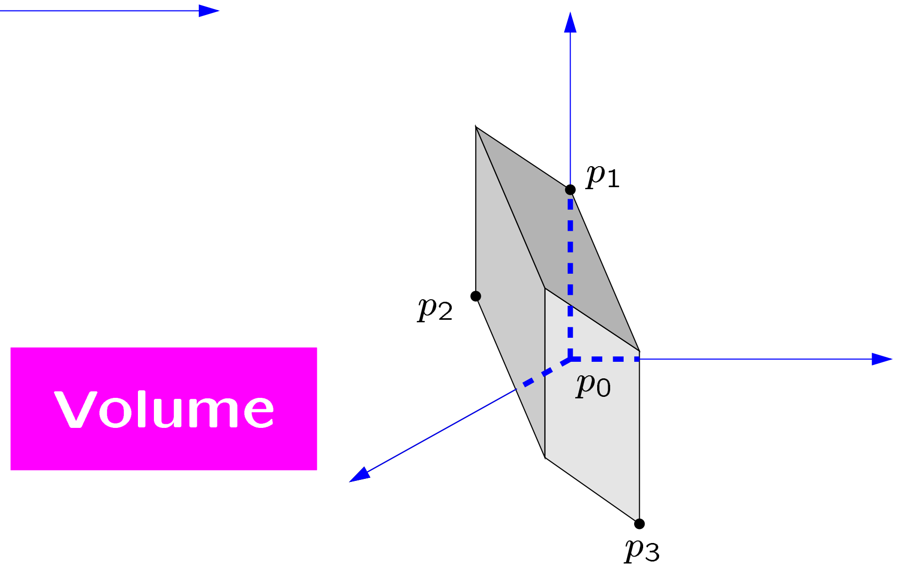
Area



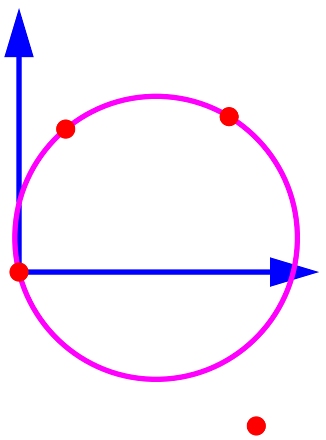
INRIA

$P(x)$	Orientation predicate

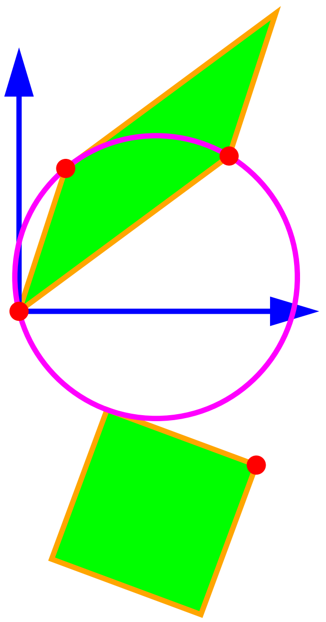




$P(x)$	Insphere predicate



$P(x)$	Insphere predicate

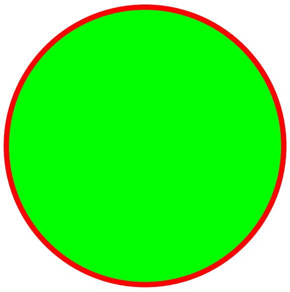


Volume $\times$ power

$P(x)$	Data distribution

Static filter

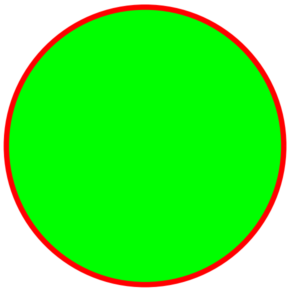
Numerical Precision and Geometrical Incoherence

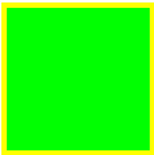
$P(x)$	Data distribution
Static filter	
Uniformly in sphere	

$P(x)$	Data distribution

Static filter

Uniformly in sphere



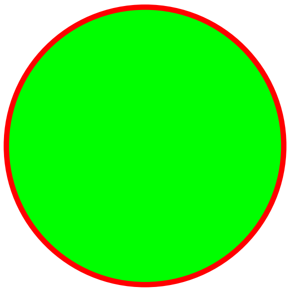


Uniformly in cube

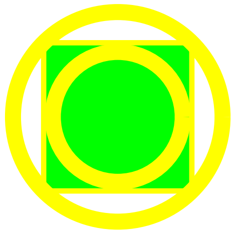
$P(x)$	Data distribution

Static filter

Uniformly in sphere



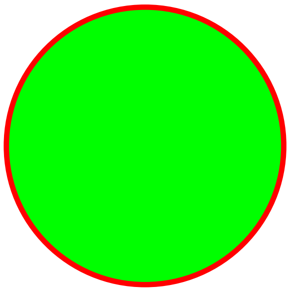
Uniformly in cube

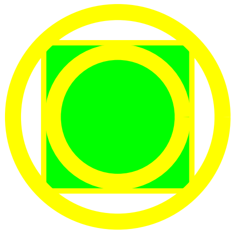


$P(x)$	Data distribution

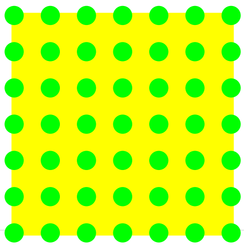
Static filter

Uniformly in sphere





Uniformly in cube



Discrete in cube

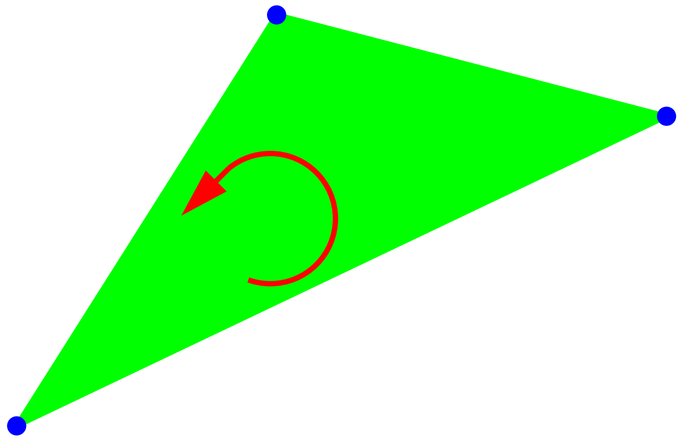
Filter efficiency

Probability $P(x) \leq$ static error ?

Filter efficiency

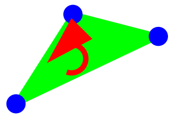
Probability $P(x) \leq$ static error ?

2D orientation 10^{-15}

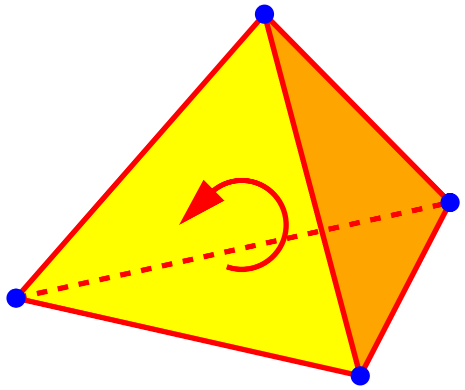


Filter efficiency

Probability $P(x) \leq$ static error ?

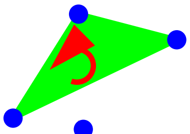
2D orientation  10^{-15}

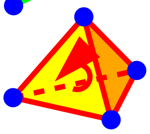
3D orientation $5 \cdot 10^{-14}$



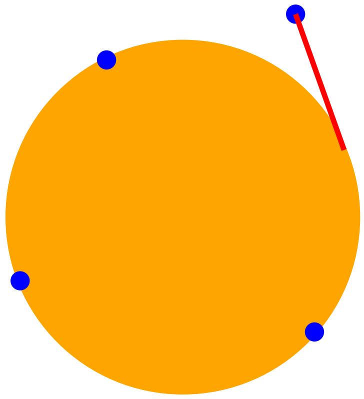
Filter efficiency

Probability $P(x) \leq$ static error ?

2D orientation  10^{-15}

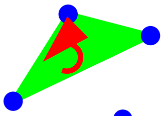
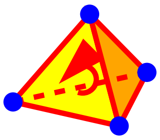
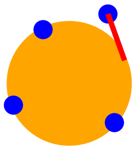
3D orientation  $5 \cdot 10^{-14}$

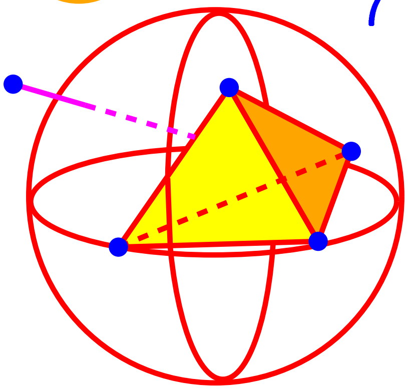
2D incircle 10^{-11}



Filter efficiency

Probability $P(x) \leq$ static error ?

2D orientation		10^{-15}
3D orientation		$5 \cdot 10^{-14}$
2D incircle		10^{-11}
3D insphere		$7 \cdot 10^{-10}$



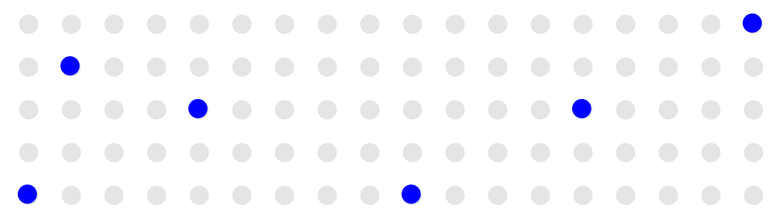
Degeneracies

Degeneracies

Discrete data

$\Longrightarrow$

Degeneracies occur

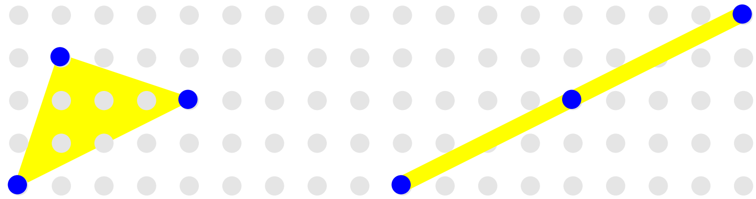


Degeneracies

Discrete data

$\Longrightarrow$

Degeneracies occur

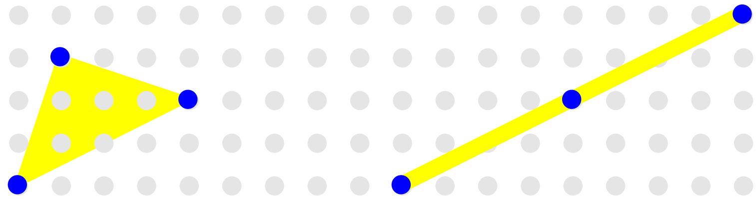


Degeneracies

Discrete data

$\Longrightarrow$

Degeneracies occur



Degeneracy $=$ $P(x) = 0$

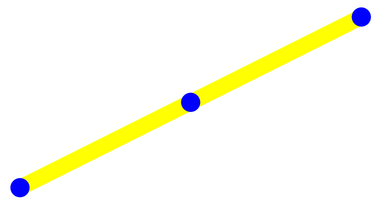
Degeneracies

2D orientation colinear

3D orientation

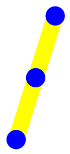
2D incircle

3D insphere



Degeneracies

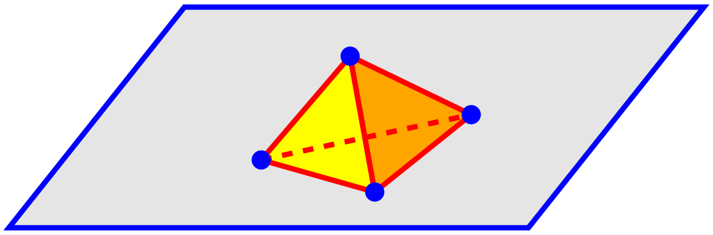
2D orientation — colinear



3D orientation — coplanar

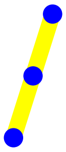
2D incircle — cocircular

3D insphere

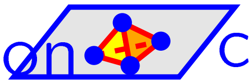


INRIA

Degeneracies



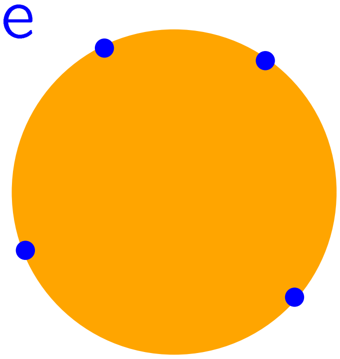
2D orientation colinear



3D orientation coplanar

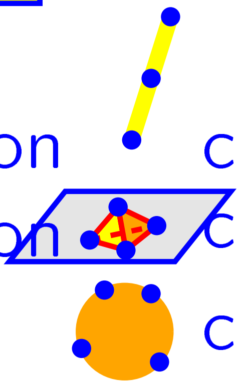
2D incircle cocircular

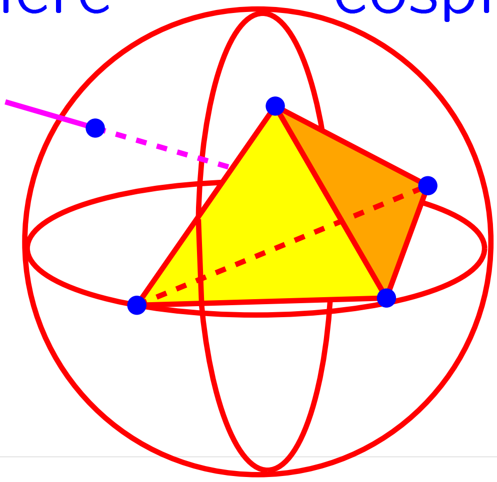
3D insphere



Degeneracies

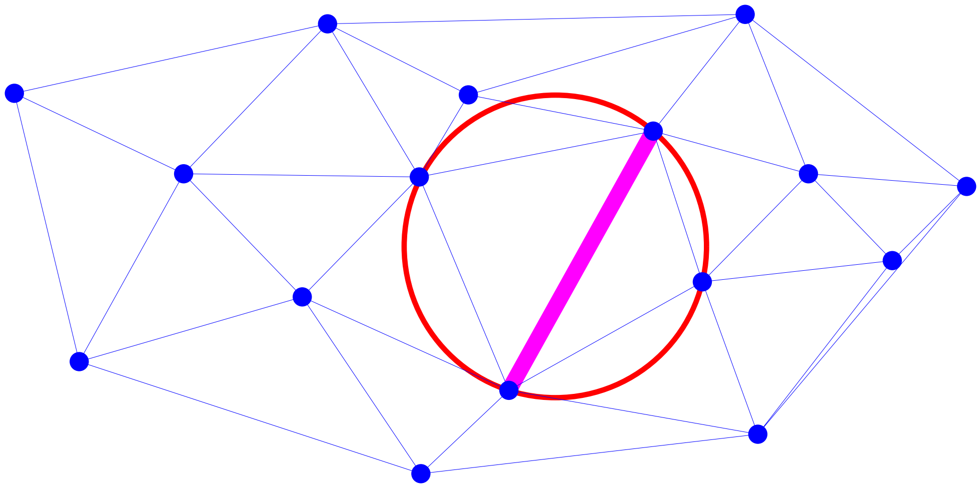
2D orientation	colinear
3D orientation	coplanar
2D incircle	cocircular
3D insphere	cospherical





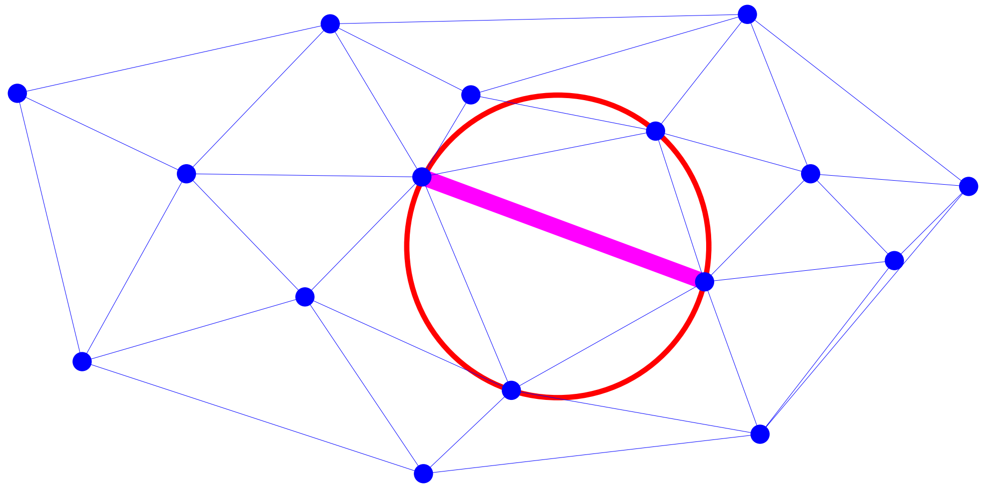
Degeneracies

What the result is ?



Degeneracies

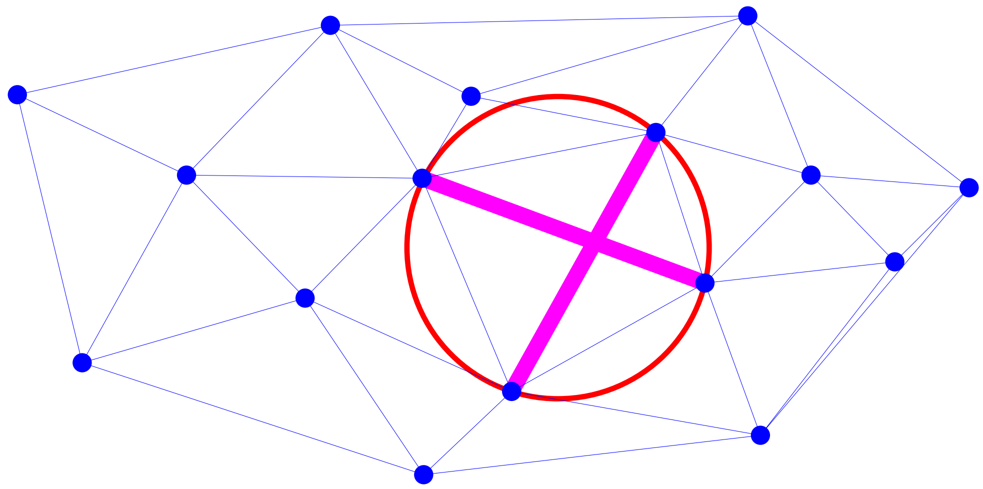
What the result is ?



May be, you don't care.

Degeneracies

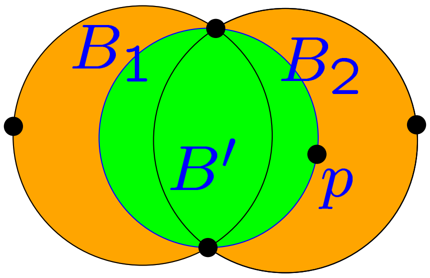
What the result is ?



May be, you don't care.

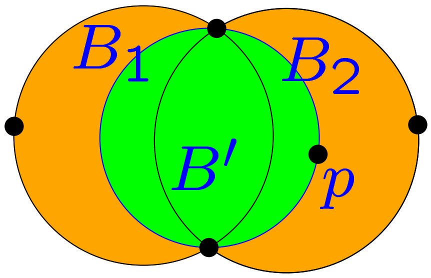
How to choose ?

Algorithms rely on geometric theorems



$p \in B_2$
$p \notin B_1$ $\Longrightarrow$ $\boxed{B' \subset B_1 \cup B_2}$

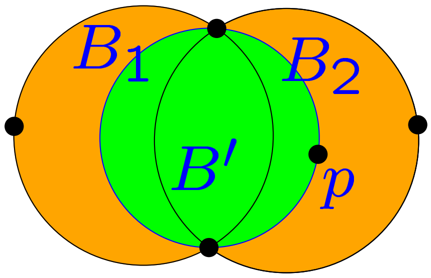
Algorithms rely on
geometric theorems



$p \in B_2$
$p \notin B_1$ $\Longrightarrow$ $B' \subset B_1 \cup B_2$

Inexact arithmetic creates bugs

Algorithms rely on
geometric theorems



$$\begin{array}{l} p \in B_2 \\ p \notin B_1 \end{array} \Longrightarrow \boxed{B' \subset B_1 \cup B_2}$$

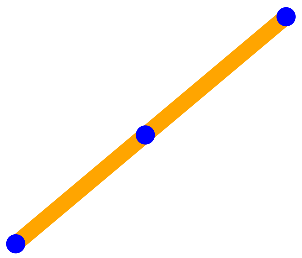
Inexact arithmetic creates bugs

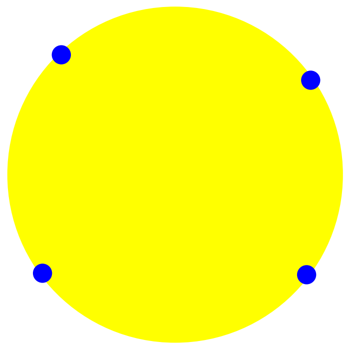
Non coherent choice for
degeneracies creates bugs

Degeneracies

Choices must be coherent

$\boxed{\text{SoS}}$ Simulation of Simplicity



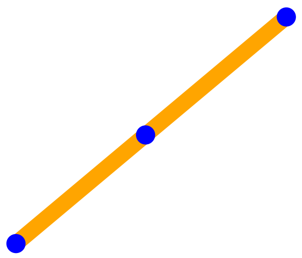


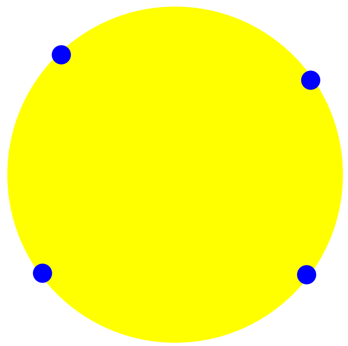
Degeneracies

Choices must be coherent

SoS Simulation of Simplicity

Perturbation scheme





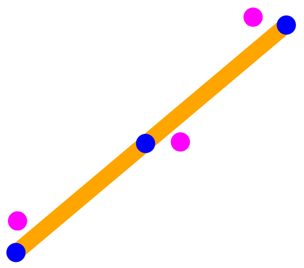
Data

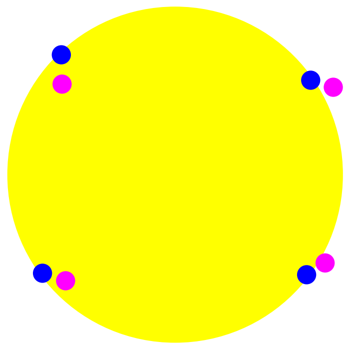
Degeneracies

Choices must be coherent

SoS Simulation of Simplicity

Perturbation scheme





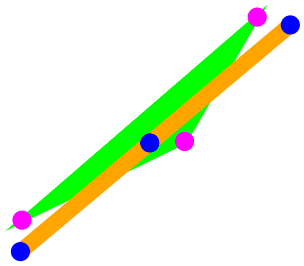
Data $\longrightarrow$ Data(ϵ)

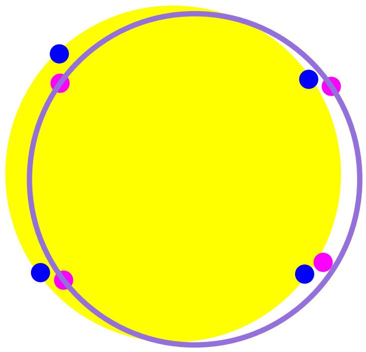
Degeneracies

Choices must be coherent

SoS Simulation of Simplicity

Perturbation scheme

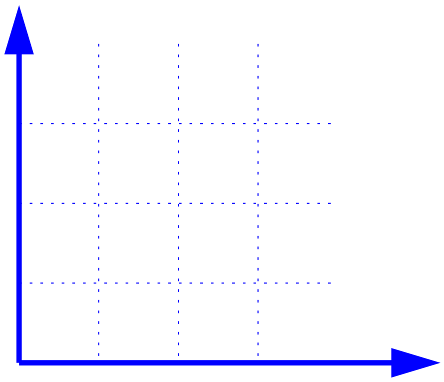




Data $\longrightarrow$ Data(ϵ)

Degeneracies

Predicate	Polynomial(Data)
compare x	$x_2 - x_1$



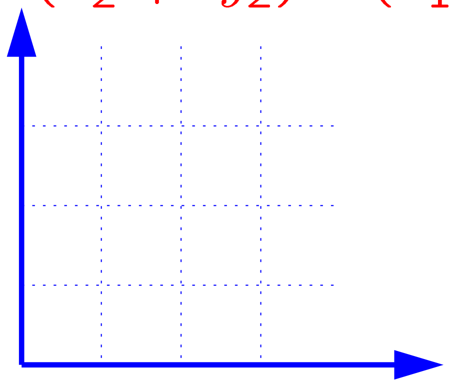
Numerical Precision and Geometrical Incoherence

Degeneracies

Predicate	Polynomial(Data)
compare x	$x_2 - x_1$

Polynomial(Data,ϵ)

$(x_2 + \epsilon y_2) - (x_1 + \epsilon y_1)$



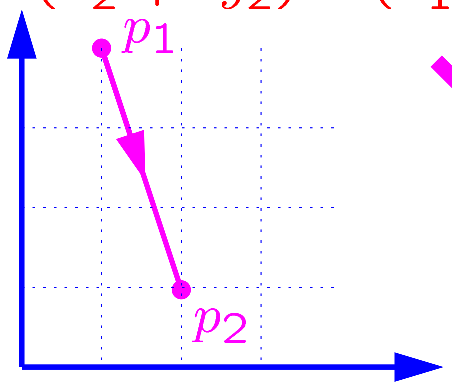
Degeneracies

Predicate	Polynomial(Data)
compare x	$x_2 - x_1$

Polynomial(Data,ϵ)

$(x_2 + \epsilon y_2) - (x_1 + \epsilon y_1)$

Actual Data



Polynomial(ϵ)

$1 - 3\epsilon$

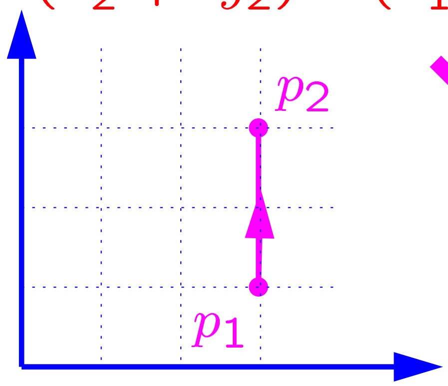
Degeneracies

Predicate	Polynomial(Data)
compare x	$x_2 - x_1$

Polynomial(Data,ϵ)

$(x_2 + \epsilon y_2) - (x_1 + \epsilon y_1)$

Actual Data



Polynomial(ϵ)

$0 + 2\epsilon$

Degeneracies

Perturbation scheme

Actual Data $\longrightarrow$ Data(ϵ)

Polynomial(Data,ϵ)

$\longrightarrow$ Polynomial(ϵ)

such that

Degeneracies

Perturbation scheme

Actual Data $\longrightarrow$ Data(ϵ)

Polynomial(Data,ϵ)

$\longrightarrow$ Polynomial(ϵ)

such that
for all Data
for all relevant predicate

$$\boxed{\text{Polynomial}(\epsilon) \neq 0}$$

INRIA SOPHIA ANTIPOLIS

Degeneracies

[Edelsbrunner Mücke]

polynomials

SoS

Degeneracies

[Edelsbrunner Mücke] polynomials

SoS

$$p'_i = p_i + \left(\epsilon^{2^{id}}, \epsilon^{2^{id+1}}, \ldots, \epsilon^{2^{id+d-1}}\right)$$

INRIA

Degeneracies

[Edelsbrunner Mücke] polynomials

SoS

$$p'_i = p_i + \left(\epsilon^{2^{id}}, \epsilon^{2^{id+1}}, \ldots, \epsilon^{2^{id+d-1}}\right)$$

Points numbering

High degree in ϵ

Very general

INRIA

Specific perturbation
"world" perturbation

Ideal perturbation

Valid output

∞ly small

Canonical output

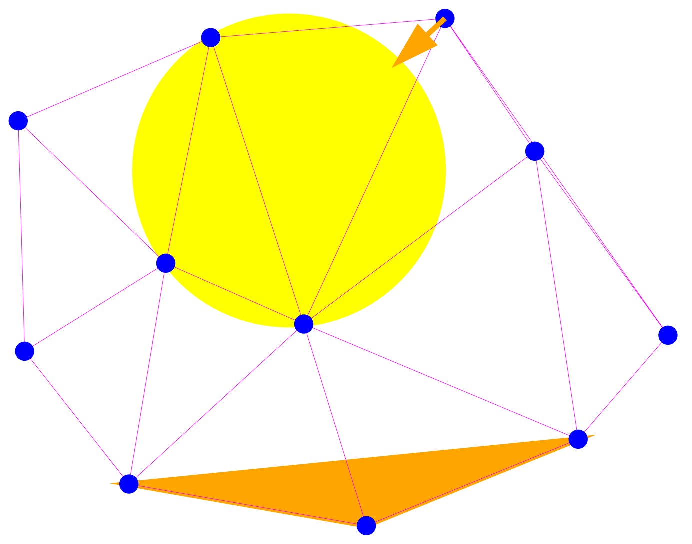
small overhead

same precision

Perturbation for Delaunay

2D orientation

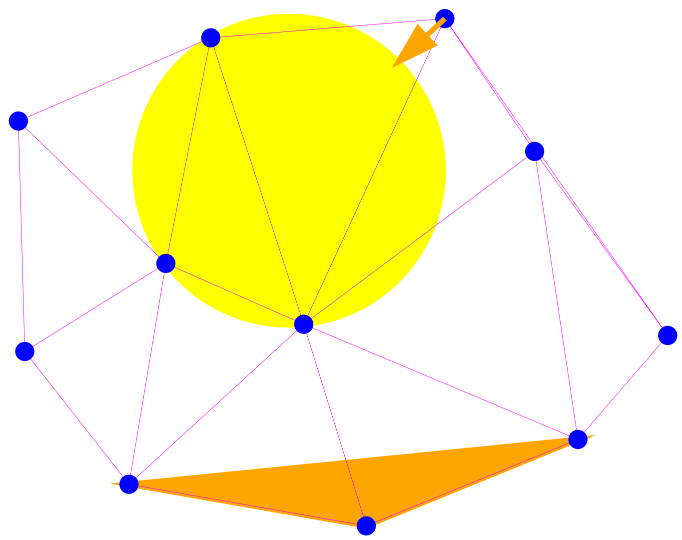
2D in circle



Perturbation for Delaunay

2D orientation 2D in circle

$$x + \epsilon y, y + \epsilon^2 x + \epsilon^3 (x^2 + y^2)$$



Perturbation for Delaunay

2D orientation 2D in circle

$$x + \epsilon y, y + \epsilon^2 x + \epsilon^3 (x^2 + y^2)$$

Theorem: Let P be a set of distinct points in the plane. Replacing each $p \in P$ with p', as defined above, removes all degeneracies for computing the Delaunay triangulation under the Euclidean metric.

INRIA SOPHIA ANTIPOLIS

Perturbation for Delaunay

2D orientation

$$x + \epsilon y, y + \epsilon^2 x + \epsilon^3 (x^2 + y^2)$$

Perturbation for Delaunay

2D orientation

$x + \epsilon y, y + \epsilon^2 x + \epsilon^3 (x^2 + y^2)$

$$(1-\epsilon^3)\begin{vmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{vmatrix} + \epsilon^3 \begin{vmatrix} 1 & p_x & p_x^2+p_y^2 \\ 1 & q_x & q_x^2+q_y^2 \\ 1 & r_x & r_x^2+r_y^2 \end{vmatrix} + \epsilon^4 \begin{vmatrix} 1 & p_y & p_x^2+p_y^2 \\ 1 & q_y & q_x^2+q_y^2 \\ 1 & r_y & r_x^2+r_y^2 \end{vmatrix}$$

Perturbation for Delaunay

2D orientation

$$x + \epsilon y, y + \epsilon^2 x + \epsilon^3 (x^2 + y^2)$$

$$(1-\epsilon^3)\begin{vmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{vmatrix} + \epsilon^3 \begin{vmatrix} 1 & p_x & p_x^2+p_y^2 \\ 1 & q_x & q_x^2+q_y^2 \\ 1 & r_x & r_x^2+r_y^2 \end{vmatrix} + \epsilon^4 \begin{vmatrix} 1 & p_y & p_x^2+p_y^2 \\ 1 & q_y & q_x^2+q_y^2 \\ 1 & r_y & r_x^2+r_y^2 \end{vmatrix}$$

Points lifted on the paraboloid

vertical plane

plane parallel to y axis

plane parallel to x axis

Impossible

Perturbation for Delaunay

2D in circle

$x + \epsilon y, y + \epsilon^2 x + \epsilon^3 (x^2 + y^2)$

Perturbation for Delaunay

2D in circle

$$x + \epsilon y, y + \epsilon^2 x + \epsilon^3 (x^2 + y^2)$$

$$D = \begin{vmatrix} 1 & p_x & p_y & p_x^2 + p_y^2 \\ 1 & q_x & q_y & q_x^2 + q_y^2 \\ 1 & r_x & r_y & r_x^2 + r_y^2 \\ 1 & s_x & s_y & s_x^2 + s_y^2 \end{vmatrix}$$

$$+2\epsilon \begin{vmatrix} 1 & p_x & p_y & p_x p_y \\ 1 & q_x & q_y & q_x q_y \\ 1 & r_x & r_y & r_x r_y \\ 1 & s_x & s_y & s_x s_y \end{vmatrix}$$

$$+\epsilon^2 \begin{vmatrix} 1 & p_x & p_y & p_y^2 \\ 1 & q_x & q_y & q_y^2 \\ 1 & r_x & r_y & r_y^2 \\ 1 & s_x & s_y & s_y^2 \end{vmatrix}$$

$$+2\epsilon^2 \begin{vmatrix} 1 & p_x & p_y & p_x p_y \\ 1 & q_x & q_y & q_x q_y \\ 1 & r_x & r_y & r_x r_y \\ 1 & s_x & s_y & s_x s_y \end{vmatrix} \qquad +\epsilon^3 D'$$

Perturbation for Delaunay

$x + \epsilon y, y + \epsilon^2 x + \epsilon^3(x^2 + y^2)$

a conic C through p, q, r

$$ax^2 + by^2 + cxy + dx + ey + f = 0$$

then

INRIA

Perturbation for Delaunay

$x + \epsilon y, y + \epsilon^2 x + \epsilon^3 (x^2 + y^2)$

a conic C through p, q, r

$$ax^2 + by^2 + cxy + dx + ey + f = 0$$

then
$$\begin{vmatrix} 1 & p_x & p_y & C(p) \\ 1 & q_x & q_y & C(q) \\ 1 & r_x & r_y & C(r) \\ 1 & s_x & s_y & C(s) \end{vmatrix}$$

$$= \begin{vmatrix} 1 & p_x & p_y & \\ 1 & q_x & q_y & C(s) \\ 1 & r_x & r_y & \end{vmatrix}$$

$= 0$ if degeneracy unsolved

INRIA SOPHIA ANTIPOLIS

Perturbation for Delaunay

$x+\epsilon y, y+\epsilon^2 x+\epsilon^3(x^2+y^2)$

a conic C through p,q,r

$$ax^2+by^2+cxy+dx+ey+f=0$$

then
$$\begin{vmatrix} 1 & p_x & p_y & C(p) \\ 1 & q_x & q_y & C(q) \\ 1 & r_x & r_y & C(r) \\ 1 & s_x & s_y & C(s) \end{vmatrix}$$

$$= \begin{vmatrix} 1 & p_x & p_y & \\ 1 & q_x & q_y & C(s) \\ 1 & r_x & r_y & \end{vmatrix}$$

$= 0$ if degeneracy unsolved

s on any conic through pqr

$pqrs$ colinear

Perturbation for Delaunay

$x + \epsilon y, y + \epsilon^2 x + \epsilon^3 (x^2 + y^2)$

$pqrs$ colinear

Perturbation for Delaunay

$x+\epsilon y, y+\epsilon^2 x+\epsilon^3(x^2+y^2)$

$pqrs$ colinear

Many simplifications

next term:

(case $p_x = q_x = r_x = s_x$)

$$\epsilon^7 \begin{vmatrix} 1 & p_y & p_y^2 & p_y^3 \\ 1 & q_y & q_y^2 & q_y^3 \\ 1 & r_y & r_y^2 & r_y^3 \\ 1 & s_y & s_y^2 & s_y^3 \end{vmatrix}$$

Perturbation for Delaunay

$$x + \epsilon y, y + \epsilon^2 x + \epsilon^3 (x^2 + y^2)$$

Tests of degree 4
Reuse of many computations
Degeneracy solved except
if equality of points

Perturbation of the world

Tool to conceive perturbation
particular purposes

Efficient implementation

Canonical output

Independent from numbering

Geometric control of output