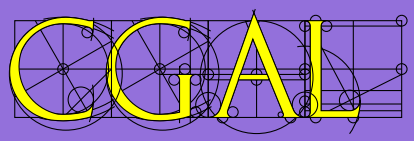
Robustness issues and 

Olivier Devillers

Computational Geometry

Theoretical hypotheses

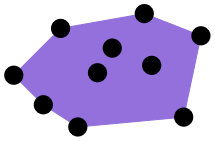
Algorithms

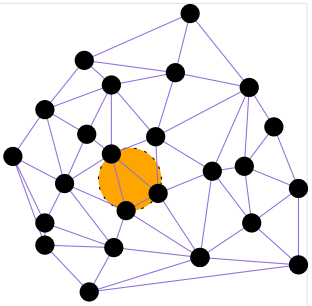
Real RAM
General position

Analysis

Real RAM
Asymptotic complexity

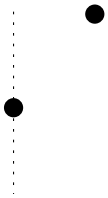
Combinatorial nature

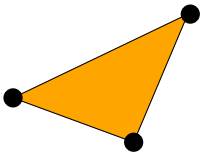


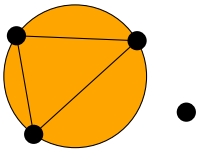


Sorting **Convex hull** **Delaunay**

Comparison *Which side* *In circle*



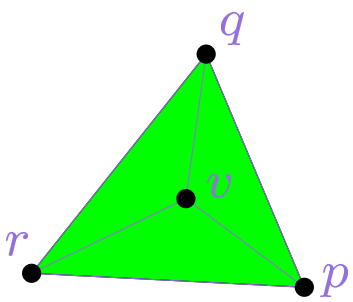




Predicates

Numerical nature

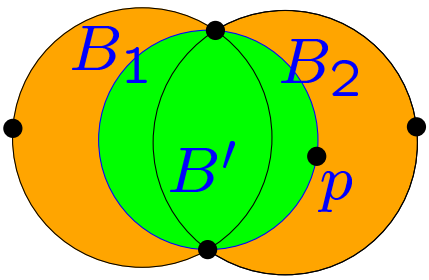
Algorithms rely on geometric theorems



pqv
qrv CCW
rpv

$\Longrightarrow$ pqr CCW

Algorithms rely on geometric theorems

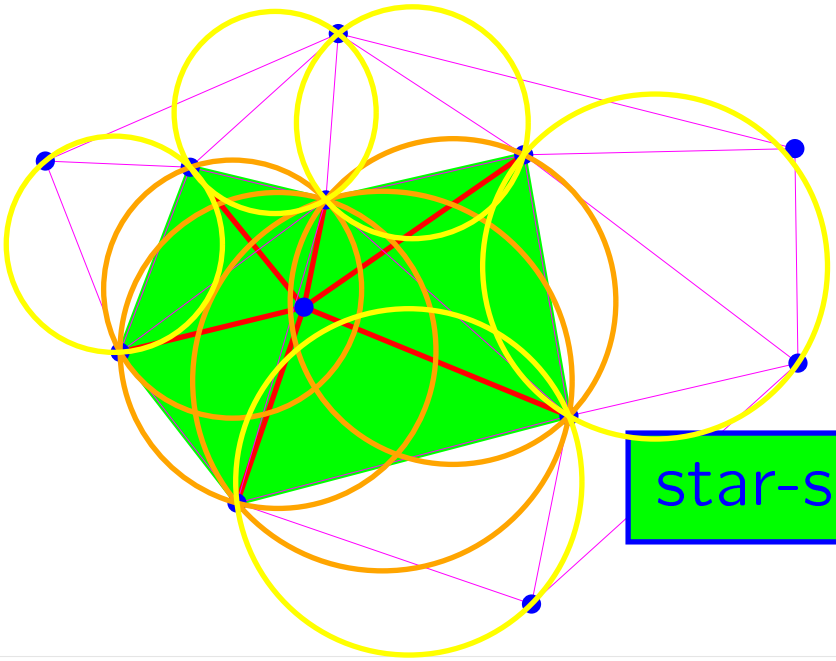


$p \in B_2$
$p \notin B_1$

$\Longrightarrow$

$B' \subset B_1 \cup B_2$

Algorithms rely on geometric theorems



Floating point arithmetic

rounded computations

IEEE 754

24 binary digits
53 binary digits

Toy model

2 decimal digits

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

errors invalidate
geometric theorems

IEEE 754

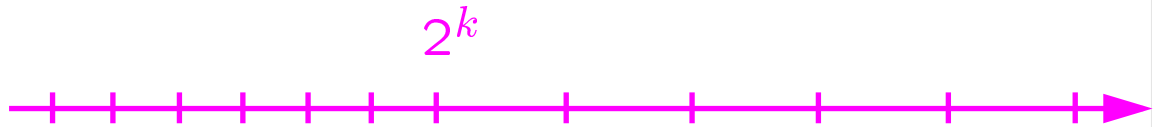
Normalised numbers

Special values

Rounding modes

Denormalised numbers

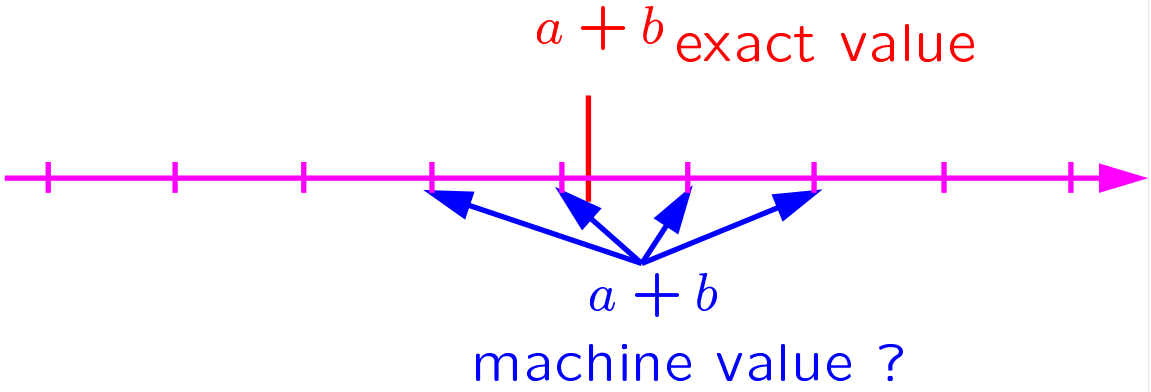
IEEE 754 Normalised numbers



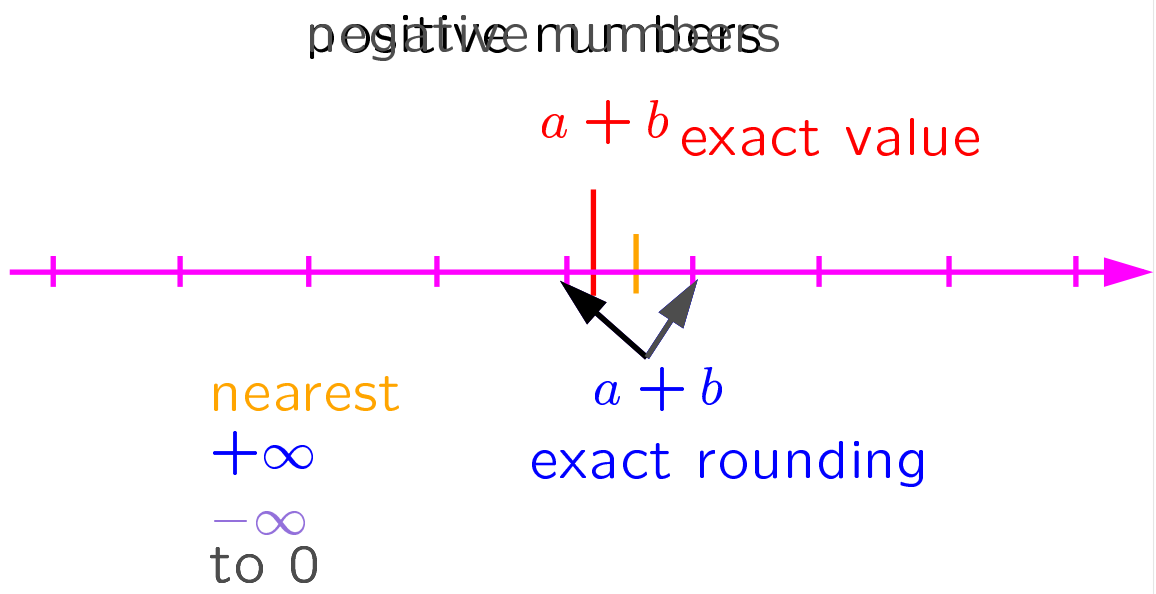
IEEE 754 Special values

Zero		$\frac{1}{\infty}$
-Zero		
∞		Overflow
$-\infty$		$\frac{1}{0}$
NaN	(not a number)	$\frac{0}{0}$

IEEE 754 Rounding modes

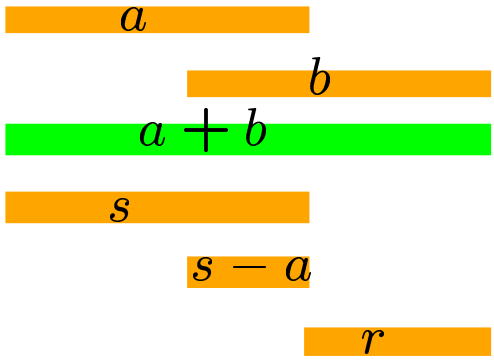


IEEE 754 Rounding modes



IEEE 754 Rounding modes

Use rounding specifications
for exact computations



rounding

exact

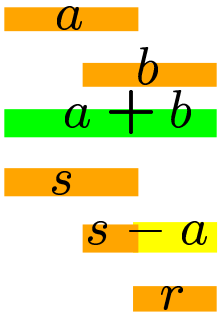
exact

```
// |a| > |b|
s=a+b;
r=b-(s-a);
```

IEEE 754

Rounding modes

Toy model




```
a=97; b=8.2;
// |a| > |b|
s=a+b;
r=b-(s-a);
```

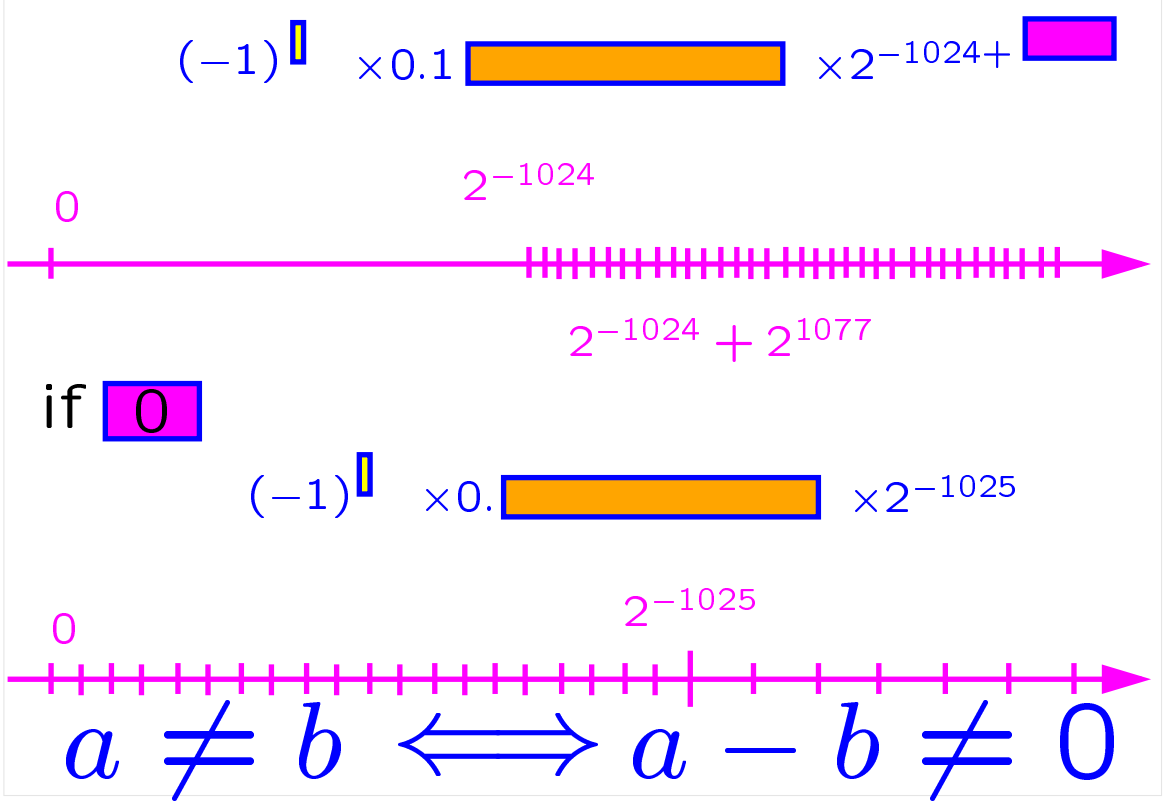
$97 + 8.2 = 105.2$

$97 + 8.2 \rightarrow 110$

$110 - 97 \rightarrow 13$

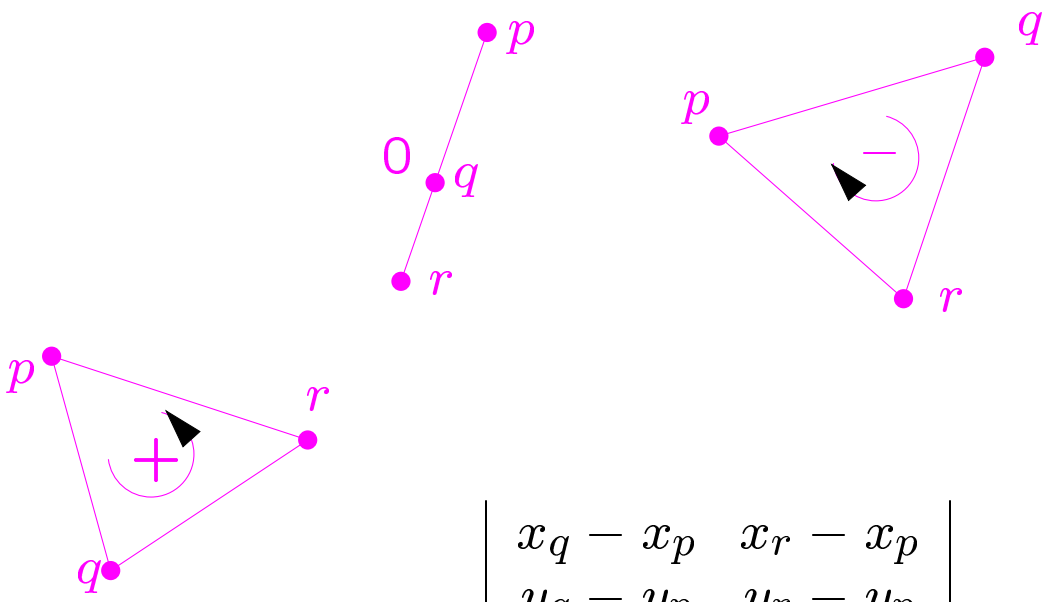
$8.2 - 13 \rightarrow -4.8$

IEEE 754 Denormalised numbers



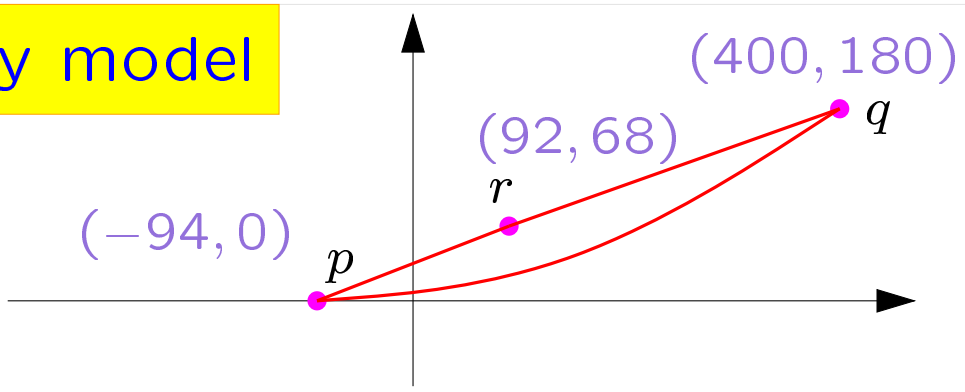
Floating point arithmetic

Orientation test



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

Toy model



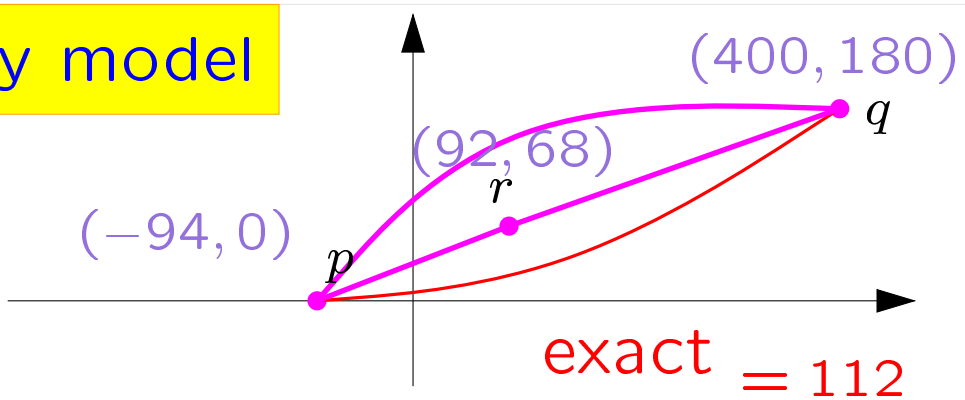
$$(400 + 94) * 68 - (92 + 94) * 180 =$$

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

$$= 112$$

Exact computation

Toy model



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

$$494 * 68 - 186 * 180$$

$$490 * 68 - 190 * 180$$

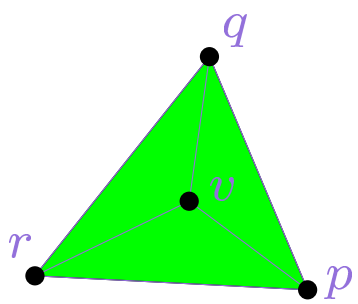
$$= 33320 - 34200$$

$$= 33000 - 34000$$

$$= -1000$$

Floating point computation

Inexact arithmetic violates theorems



pqv
qrv CCW $\Longrightarrow$ pqr CCW
rpv

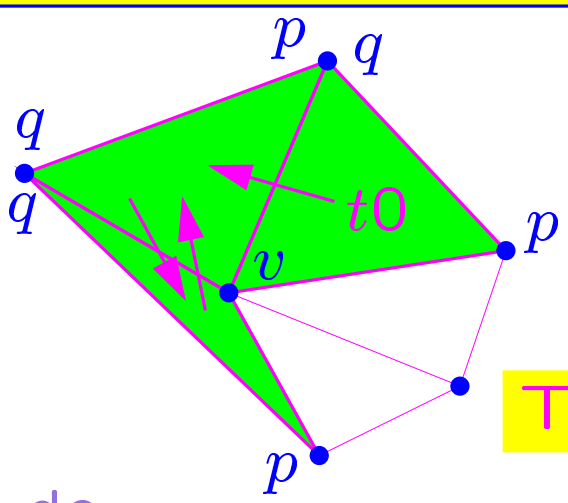
Toy model

q $(400, 180)$

r
v $(92, 68)$
$(-94, 0)$
p $(-5, 34)$

pqv
qrv CCW and pqr CW
rpv

Inexact arithmetic creates bugs



Example :

Turn ccw around v

do

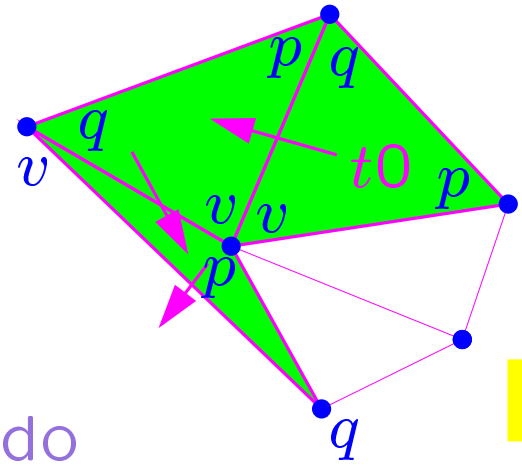
let vpq ccw triangle

go to neighbor through qv

while triangle $\neq t0$

Infinite loop

Inexact arithmetic creates bugs



Example :

Turn ccw around v

Turn ccw around v

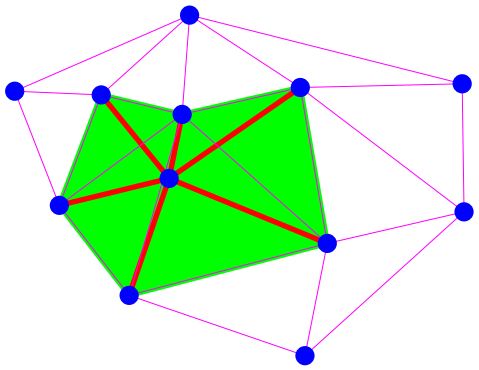
do
  let vpq ccw triangle
    such that vp incoming edge
  go to neighbor through qv
while triangle $\neq t0$

Infinite loop ?
core dumped ?

Solution 1

Do not rely on theorems

Example



star-shaped

weaker version

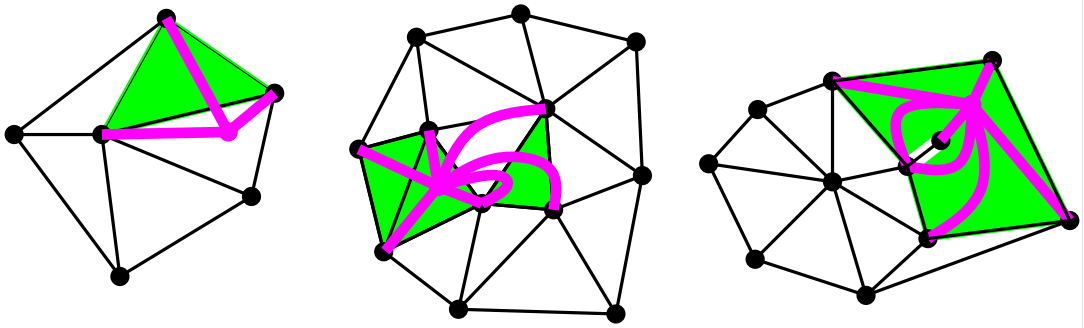
simple polygon

simple polygon

purely topological

Special algorithm

Sugihara & Iri



topological triangulation

robust

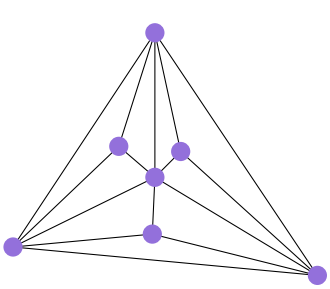
Weaker properties

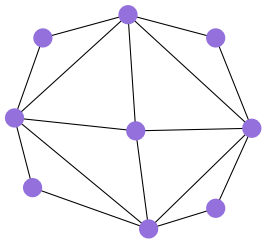
Non geometrically correct

non Delaunay triangulation
non planar triangulation

even Non topologically correct

non realizable as Delaunay

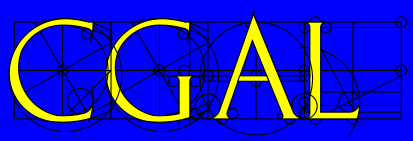




Solution 1

Do not rely on theorems

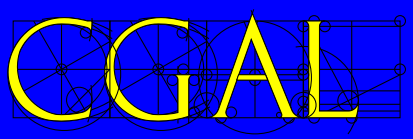
not used in

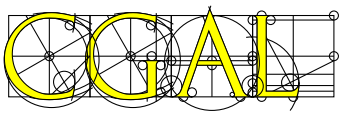


Solution 2

Use exact computations

Used in

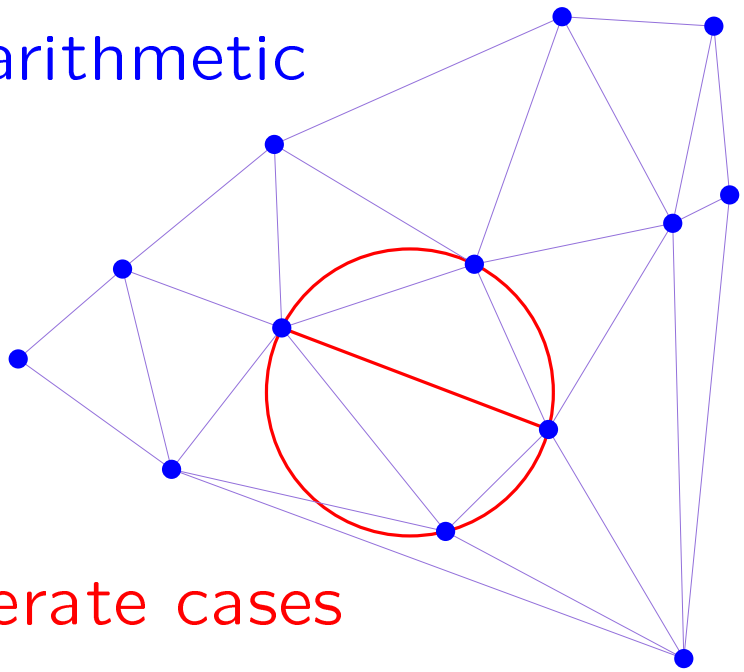


 relies on theorems

Use exact computations

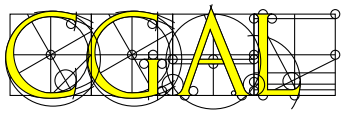
Exact data

Exact arithmetic



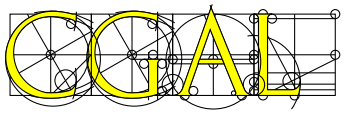
Degenerate cases

Exact arithmetic



```
typedef CGAL-Point-2
  <CGAL-Cartesian
    <leda-real>
  >                Point;
```

Exact arithmetic



template

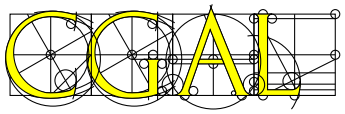
rounded arithmetic

rounded computations

exact arithmetic

exact computations

Exact arithmetic



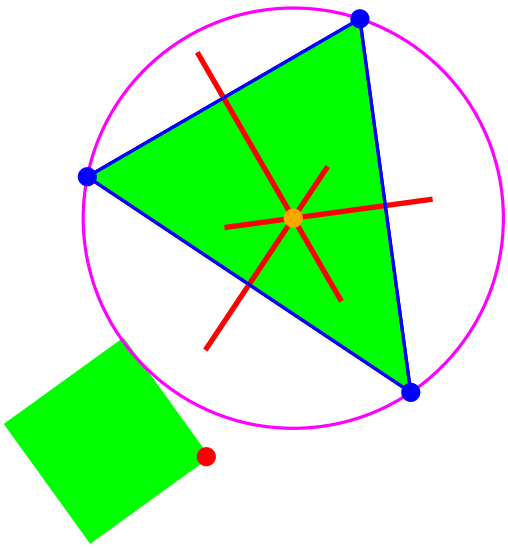
int
float
double
rounded

integer
rational
real

GMP

mpz
mpq
mpf

constructions
versus
predicates



circumcenter

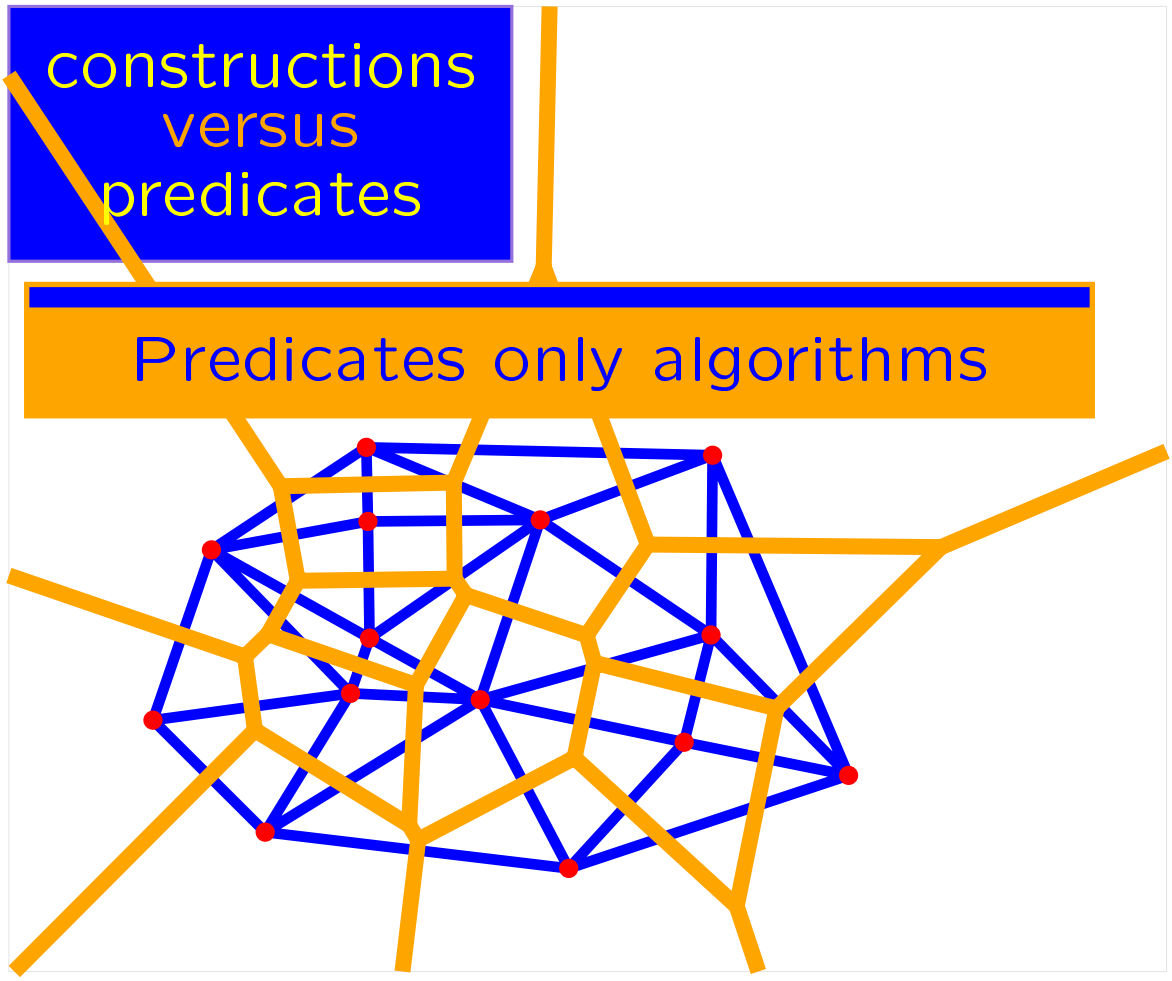
exact

rational

in circle test

exact

sign of polynomial


constructions
versus
predicates
Predicates only algorithms

Geometric predicate

sign of polynomial

exact computation → non constant time

precision=degree

complexity pb

floating point → sometimes false

approximation

unacceptable

Geometric predicate

sign of polynomial

Exact predicate

$\neq$

Exact arithmetic

Exact sign

$\neq$

Exact value

Given an arithmetic expression

A filter is

An approximate evaluator

A certifier

An exact evaluator

Approximate evaluator

rounded computation

Certifier

error computation

a priori

at execution

if $|result| \geq error$ sign is certified

- Filters in CGAL

Exact number types

CEDA
- integer
- rational

GMP
- mpz
- mpq
- mpf

No filters

Exact value computed

Predicates in CGAL

Exact predicate
using exact computation

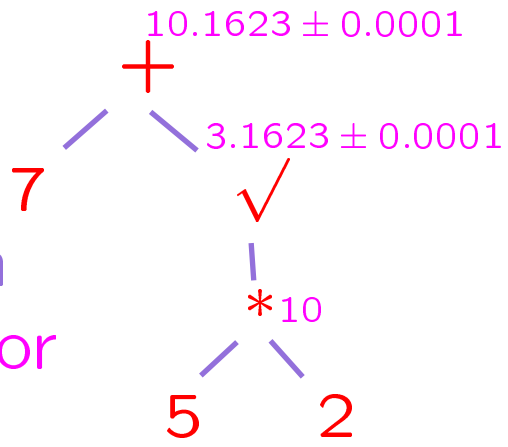
```
template <class FT>
CGAL-OrientationC2
                (FT x0,y0,x1,y1,x2,y2)
{
  return sign( | x1 − x0   x2 − x0 | )
               | y1 − y0   y2 − y0 |
}
```

Filters in CGAL

CEDA `real`

supports $+ - * / \sqrt{} = \neq \leq \ldots$

a leda-real is



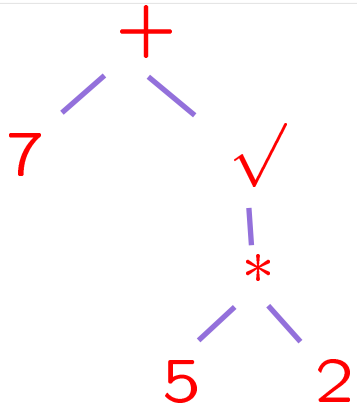
symbolic definition

rounded value, error

- Filters in CGAL

CEDA real

If needed

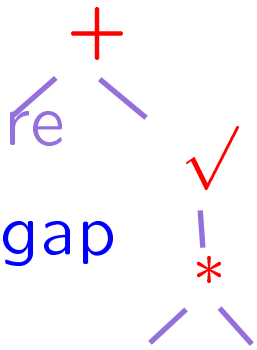


Evaluation performed
for comparison only

Up to sufficient precision

Given the tree structure

minimal gap



Predicates in CGAL

Exact predicate using CGAL real

```
template <class FT>
CGAL-OrientationC2
                (FT x0,y0,x1,y1,x2,y2)
{
  return sign( | x1 − x0   x2 − x0 | )
               | y1 − y0   y2 − y0 |
}
```

Filters in

Error computation

Interval arithmetic
any input

Static error
Semi-static error
input on grid

Predicates in

Exact predicate
using filters

template <> **specialization**
CGAL-OrientationC2
(Sft x0,y0,x1,y1,x2,y2)

Sft is a specific number type

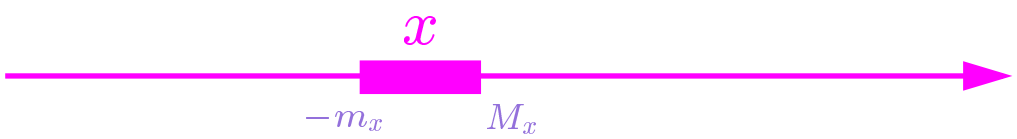
{
. . .
}

- Predicates in

```
template <>
CGAL-OrientationC2
                (Sft x0,y0,x1,y1,x2,y2)
{
  D=approx( | x1-x0  x2-x0 | )
            | y1-y0  y2-y0 |
  if |D| ≥ error() return sign(D);
  return exactsign
                ( | x1-x0  x2-x0 | )
                | y1-y0  y2-y0 |
}
```

Interval arithmetic

$$x \in [-m_x, M_x]$$



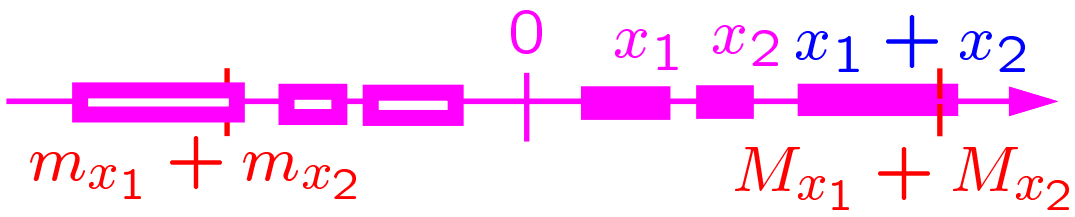
IEEE `double` m_x, M_x

IEEE rounding mode $+\infty$

Interval arithmetic

$x_1 + x_2$

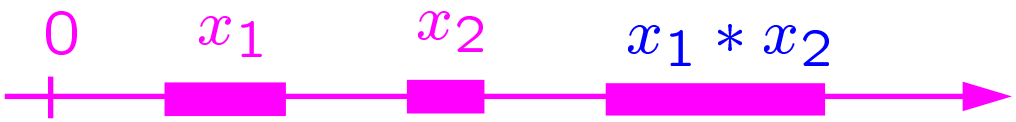
$$M_{x_1+x_2} = M_{x_1} + M_{x_2}$$
$$m_{x_1+x_2} = m_{x_1} + m_{x_2}$$



Interval arithmetic

$x_1 * x_2$

$$m_{x_1 * x_2} = (-m_{x_1}) * m_{x_2}$$
$$M_{x_1 * x_2} = M_{x_1} * M_{x_2}$$

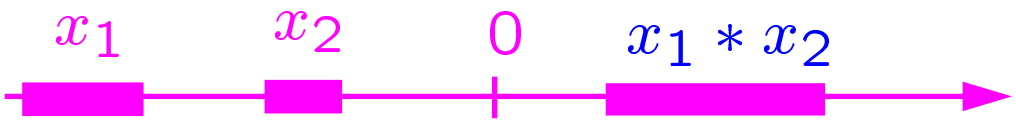


$x_1, x_2 \geq 0$

Interval arithmetic

$x_1 * x_2$

$$m_{x_1 * x_2} = (-M_{x_1}) * M_{x_2}$$
$$M_{x_1 * x_2} = m_{x_1} * m_{x_2}$$



$x_1, x_2 \leq 0$

Interval arithmetic

$x_1 * x_2$

$$m_{x_1 * x_2} = -(M_{x_1} * m_{x_2})$$
$$M_{x_1 * x_2} = (-m_{x_1}) * M_{x_2}$$

$x_1 \leq 0, x_2 \geq 0$

Interval arithmetic

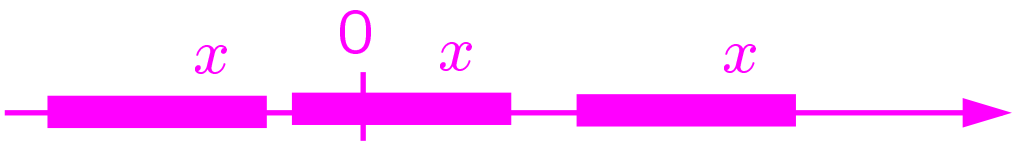
x_1/x_2

$x \geq 0$

if $0 \in [-m_x, M_x]$ exception

if $0 > M_x$ return $-$

if $0 < m_x$ return $+$



- Filters in

  Interval arithmetic

  CGAL-Interval-nt
  supports $+ - * / \sqrt{}$

  CGAL-Filtered-exact<CT,ET>

  CT
  construction and storage

  ET
  exact type

Predicates in CGAL

Exact predicate
using interval arithmetic to filter

```
template <class CT, class ET>
CGAL-OrientationC2                    specialization
  (CGAL-Filtered-exact<FT,ET>
                        x0,y0,x1,y1,x2,y2)
{
  typedef CGAL-Interval-nt IA;
    try { ... }
    catch { ... }
}
```

```
template <class CT, class ET>
   CGAL-OrientationC2
    (CGAL-Filtered-exact<FT,ET> x0,y0,x1,y1,x2,y2)
{  typedef CGAL-Interval-nt IA;
   try {
     S=CGAL-OrientationC2(
                    convert-to<IA>(x0),
                    convert-to<IA>(y0),
                    convert-to<IA>(x1),
                    convert-to<IA>(y1),
                    convert-to<IA>(x2),        );
                    convert-to<IA>(y2)
      }
    catch { . . . }
 }
```

```
template <class CT, class ET>
   CGAL-OrientationC2
   (CGAL-Filtered-exact<FT,ET> x0,y0,x1,y1,x2,y2)
{  typedef CGAL-Interval-nt IA;
   try {
        S=CGAL-OrientationC2( using IA   );
   }
   catch (IA::unsafe-comparison) {
    S=CGAL-OrientationC2(
                  convert-to<ET>(x0),
                  convert-to<ET>(y0),
                  convert-to<ET>(x1),
                  convert-to<ET>(y1),
                  convert-to<ET>(x2),
     }            convert-to<ET>(y2)      );
  }
```

```
template <class CT, class ET>
   CGAL-OrientationC2
    (CGAL-Filtered-exact<FT,ET>x0,y0,x1,y1,x2,y2)
{ typedef CGAL-Interval-nt IA;
  try {
        IEEE754 rounding to +∞
         S=CGAL-OrientationC2( using IA   );
  }     IEEE754 rounding to nearest
  catch (IA::unsafe-comparison) {
        IEEE754 rounding to nearest
         S=CGAL-OrientationC2( using ET );
  }
  return S;
}
```

Filters in

CGAL-Filtered-exact<CT,ET>

```
template <class CT, class ET>
  CGAL-OrientationC2
    (CGAL-Filtered-exact<FT,ET> x0,y0,x1,y1,x2,y2)
{
  • • •
}
```

Automatic production

script

template version ⟶ specialized version

Filters in

Error computation

Interval arithmetic
any input

Static error
Semi-static error
input on grid

Static error

```
double a,b;
```

IEEE rounding mode nearest

DATA

Known upper bound

Predicate

Sign of polynomial

no division

Worst case error

Static error

`double a,b;`

if

$|a| \leq A$

$|b| \leq B$

$A + B \leq 2^k$ =-1023+

then

rounding error (a+b) $\leq 2^{k-54}$

(-1) $\times 0.1$ $\times 2^{-1024+}$

53 bits

Static error

$$\left(\begin{vmatrix} x1-x0 & x2-x0 \\ y1-y0 & y2-y0 \end{vmatrix} \right)$$

$$|x0|, |y0|, |x1| \ldots \leq 2^k$$

$$|x1 - x0| \ldots \leq 2^{k+1}$$

$$\epsilon(x1 - x0) \ldots \leq 2^{k+1-54} = 2^{k-53}$$

$$|(x1 - x0) * (y2 - y0)| \ldots \leq 2^{2k+2}$$

$$\epsilon((x1 - x0) * (y2 - y0)) \ldots$$
$$\leq 2 \cdot 2^{k-53} * 2^{k+1} + 2^{2k+2-54}$$
$$= 2^{2k-51}$$

$$\left| \begin{matrix} \cdot & \cdot \\ \cdot & \cdot \end{matrix} \right| \leq 2^{2k+3}$$

$$\epsilon\left(\left| \begin{matrix} \cdot & \cdot \\ \cdot & \cdot \end{matrix} \right| \right) \leq 2 \cdot 2^{2k-51} + 2^{2k+3-54}$$
$$= 3 \cdot 2^{2k-51}$$

Predicates not in CGAL !

```
template <>
CGAL-OrientationC2
        (INT<k> x0,y0,x1,y1,x2,y2)
{ double X0=x0,Y0=y0,X1=x1,
         Y1=y1,X2=x2,Y2=y2;
```

$D = \begin{vmatrix} X1 - X0 & X2 - X0 \\ Y1 - Y0 & Y2 - Y0 \end{vmatrix}$;

if $|D| \geq 3 * 2^{2k-51}$ return sign(D);

return exactsign $\left(\begin{vmatrix} x1 - x0 & x2 - x0 \\ y1 - y0 & y2 - y0 \end{vmatrix} \right)$

}

Semi-static error

Static error

```
double a,b;
```

IEEE rounding mode nearest

DATA (Known upper bound)

Predicate Sign of polynomial
no division

Data sensitive error

Semi-static error

`double a,b;`

if

$$|a+b| \leq 2^k \leq 2|a+b|$$

then

$$\text{rounding error } (a+b) \leq 2^{k-54}$$

$$\text{rounding error } (a+b) \leq 2^{-53}|a+b|$$

Semi-static error

$$\left(\begin{vmatrix} x1-x0 & x2-x0 \\ y1-y0 & y2-y0 \end{vmatrix} \right)$$

$|x1 - x0| \ldots$

$$\epsilon(x1 - x0) \ldots \leq 2^{k-53}|x1 - x0|$$

$|(x1 - x0) * (y2 - y0)| \ldots$

$$\begin{aligned} &\epsilon((x1 - x0) * (y2 - y0)) \ldots \\ &\leq 2 \cdot 2^{-53}(|x1 - x0| * |y2 - y0|) \\ &\qquad +2^{-53}|(x1 - x0)(y2 - y0)| \\ &\leq 3 \cdot 2^{-53}(|x1 - x0| * |y2 - y0|) \end{aligned}$$

Semi-static error

$$\left(\begin{vmatrix} x1-x0 & x2-x0 \\ y1-y0 & y2-y0 \end{vmatrix} \right)$$

$$\epsilon(x1-x0)\ldots \leq 2^{k-53}|x1-x0|$$

$$\epsilon((x1-x0)*(y2-y0))\ldots$$
$$\leq 3\cdot 2^{-53}(|x1-x0|*|y2-y0|)$$

$$\begin{vmatrix} \cdot & \cdot \\ \cdot & \cdot \end{vmatrix}$$

$$\epsilon(\begin{vmatrix} \cdot & \cdot \\ \cdot & \cdot \end{vmatrix})$$

$$\leq 3\cdot 2^{-53}$$
$$(|x1-x0|*|y2-y0|+|x2-x0|*|y1-y0|))$$
$$+2^{-53} \begin{vmatrix} \cdot & \cdot \\ \cdot & \cdot \end{vmatrix}$$
$$\leq 2^{-51}$$
$$(|x1-x0|*|y2-y0|+|x2-x0|*|y1-y0|)$$

Predicates not in !

template <>
CGAL-OrientationC2 (Sft x0,y0,x1,y1,x2,y2)

{ double X0=x0,Y0=y0,X1=x1,
Y1=y1,X2=x2,Y2=y2;

D= $\begin{vmatrix} X1-X0 & X2-X0 \\ Y1-Y0 & Y2-Y0 \end{vmatrix}$;

if $|D| \geq 2^{-51} \left(\begin{array}{rl} & |X1-X0| * |X2-X0| \\ + & |Y1-Y0| * |Y2-Y0| \end{array} \right)$;

return sign(D);

return exactsign $\left(\begin{vmatrix} x1-x0 & x2-x0 \\ y1-y0 & y2-y0 \end{vmatrix} \right)$

}

- Predicates in CGAL

CGAL-Fixed-precision-nt

grid $2^{24} \times 2^{24}$

grid size 2^k

Exact degree 2 computations

Static error

Semi-static error

Predicates in CGAL

template <> CGAL-OrientationC3
(Fixed x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3)

```
{   double X0=x0,Y0=y0,Z0=z0,X1=x1,Y1=y1,Z1=z1,
           X2=x2,Y2=y2,Z2=z2,X3=x3,Y3=y3,Z3=z3;
```

$$// \left| \begin{array}{ccc} X1-X0 & X2-X0 & X3-X0 \\ Y1-Y0 & Y2-Y0 & Y3-Y0 \\ Z1-Z0 & Z2-Z0 & Z3-Z0 \end{array} \right| ;$$

M1,M2,M3 = minors;

U1,U2,U3 = last line;

D=M1*U1+M2*U2+M3*U3;

• • •

```
}
```

Predicates in

template <> CGAL-OrientationC3
(Fixed x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3)

{

M1,M2,M3 = minors;
U1,U2,U3 = last line;

static error

D=M1*U1+M2*U2+M3*U3;

if $|D| \geq 9 \cdot 2^{22+3k}$ return sign(D)

$\epsilon = \frac{3}{2^{53}}$|M1*U1| + |M2*U2| + |M3*U3|;

if $|D| \geq \epsilon$ return sign(D)

semi-static error

compute exactly(M1*U1+M2*U2+M3*U3);
return sign;

}

Filter efficiency

Sign of polynomial P

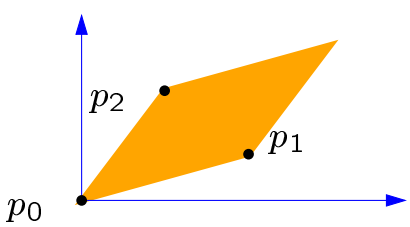
If $P(x)$ is small **filters fail**

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

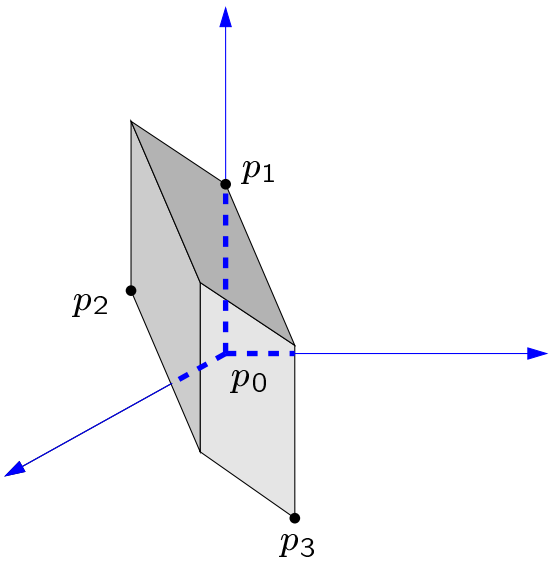
Data distribution ?

Data distribution
Predicate
Algorithm

$P(x)$	Orientation predicate

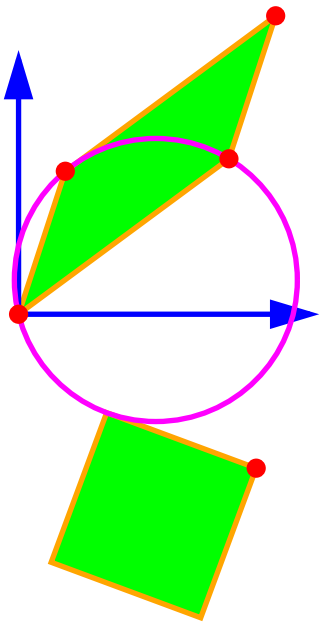


Area



Volume

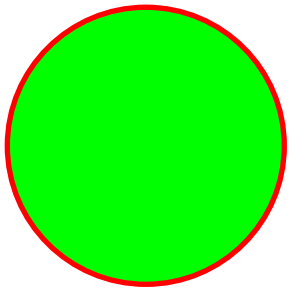
$P(x)$	Insphere predicate



Volume $\times$ power

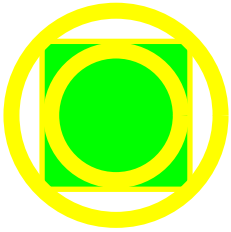
$P(x)$	Data distribution

Static filter

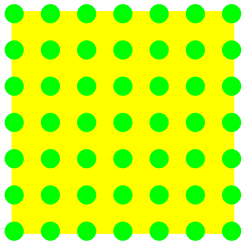


Uniformly in sphere

Uniformly in cube

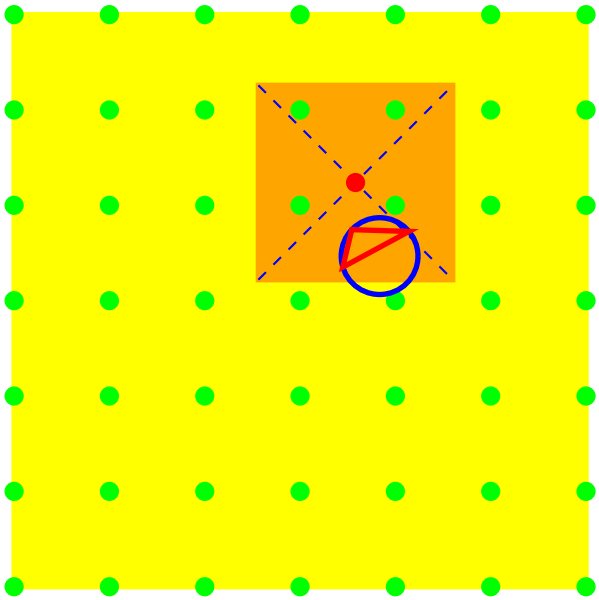


Discrete in cube



$P(x)$ | Data distribution

Semi-static filter



Discrete in bounding cube

Filter efficiency

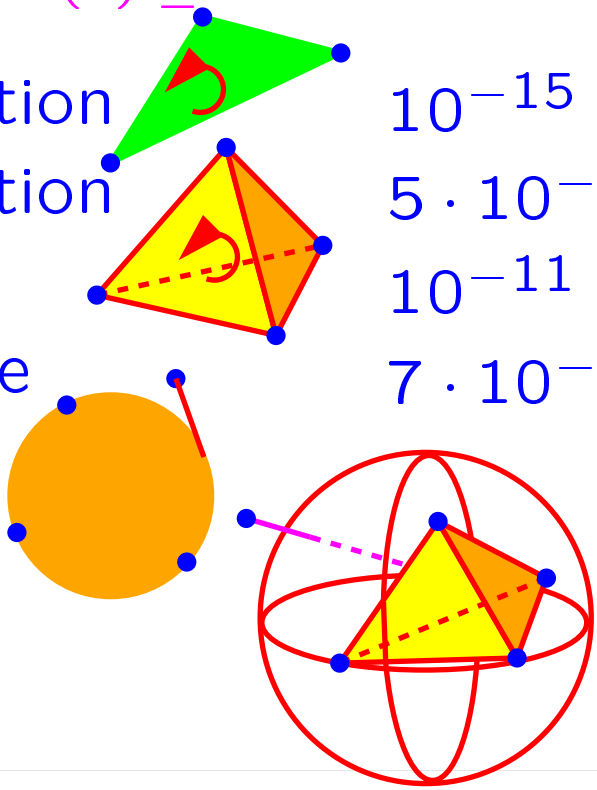
Probability $P(x) \leq$ static error ?

2D orientation 10^{-15}

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

2D incircle 10^{-11}

3D insphere $7 \cdot 10^{-10}$

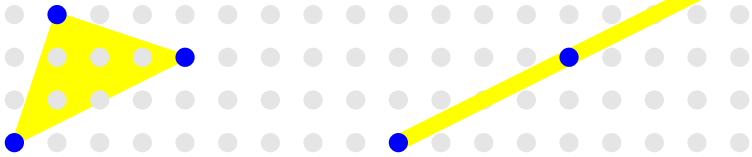


Degeneracies

Discrete data

$\Longrightarrow$

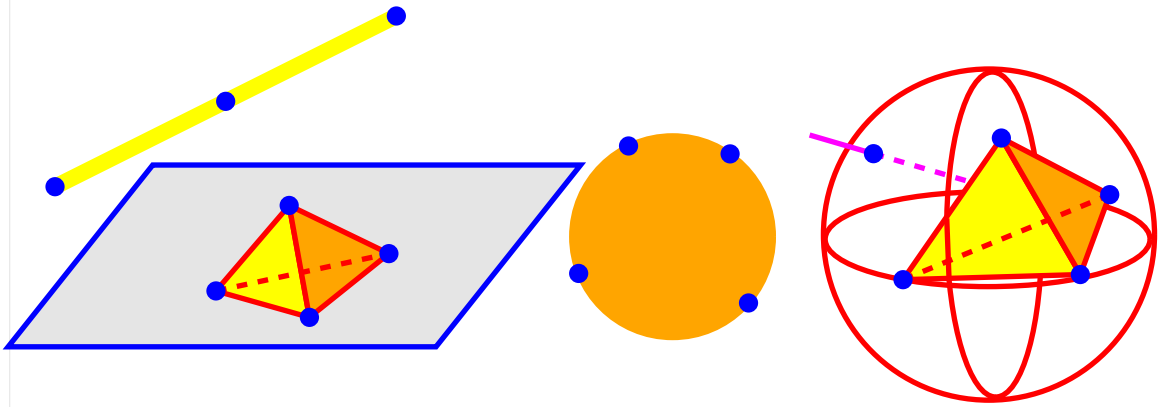
Degeneracies occur



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

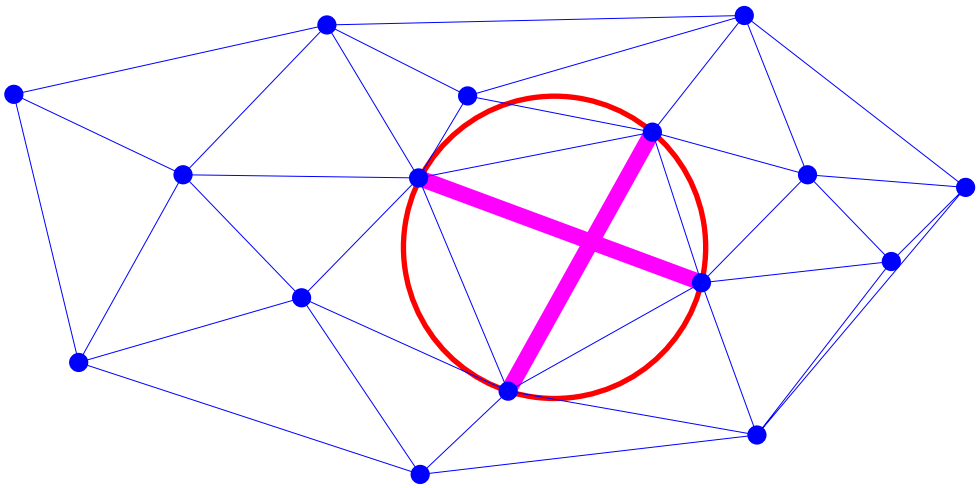
Degeneracies

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



Degeneracies

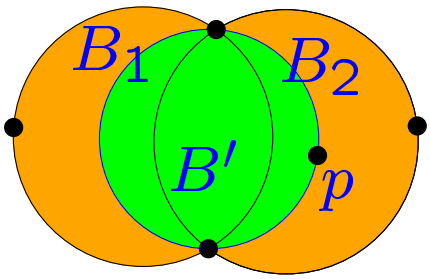
What the result is ?



May be, you don't care.

How to choose ?

Algorithms rely on geometric theorems



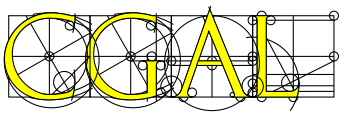
$p \in B_2$
$p \notin B_1$
$\Longrightarrow$
$B' \subset B_1 \cup B_2$

Inexact arithmetic creates bugs

Non coherent choice for degeneracies creates bugs

Degeneracies

Choices must be coherent

 relies on theorems

Degeneracies

Used in CGAL

Particular cases

Predicate ⟶ + - 0

Degeneracies

Used in CGAL

Perturbation scheme

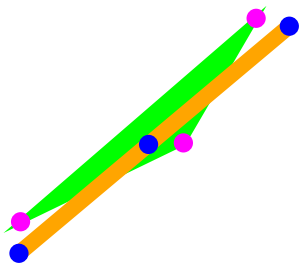
CGAL-Fixed-precision-nt

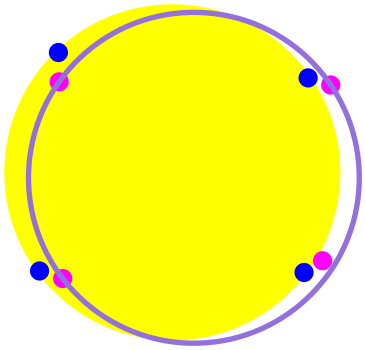
Degeneracies

Choices must be coherent

SoS Simulation of Simplicity

Perturbation scheme





Data $\longrightarrow$ Data(ϵ)

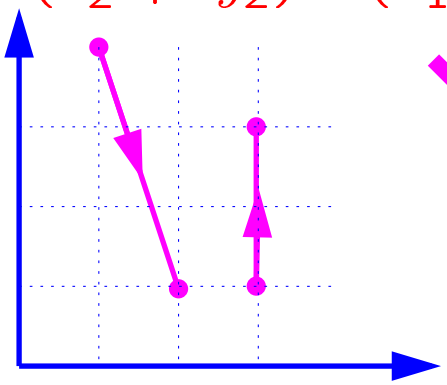
Degeneracies

Predicate | Polynomial(Data)

compare x $\quad x_2 - x_1$

Polynomial(Data,ϵ) $\quad$ Actual Data

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



Polynomial(ϵ)

$1 - 3\epsilon$

$0 + 2\epsilon$

Degeneracies

Perturbation scheme

Actual Data → Data(ϵ)

Polynomial(Data,ϵ)

→ Polynomial(ϵ)

such that
for all Data
for all relevant predicate

Polynomial(ϵ) $\neq 0$

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

Specific perturbation "world" perturbation

Ideal perturbation

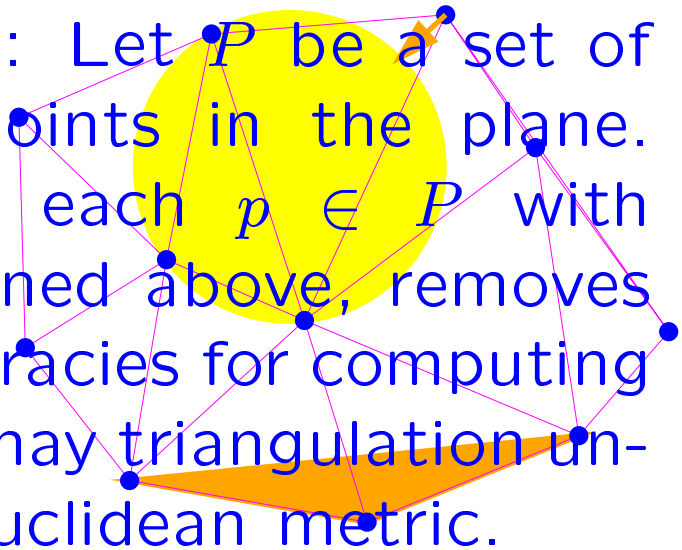
- Valid output
- ∞ly small
- Canonical output
- small overhead
- same precision

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.



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

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

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

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

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

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

$+\epsilon^3 D'$

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

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

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

$= 0$ 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

Many simplifications

next term:

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

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

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