Verified Convex Hull for Inexact Data

Yuki Ohta[1,*]
Katsuhisa Ozaki[2]

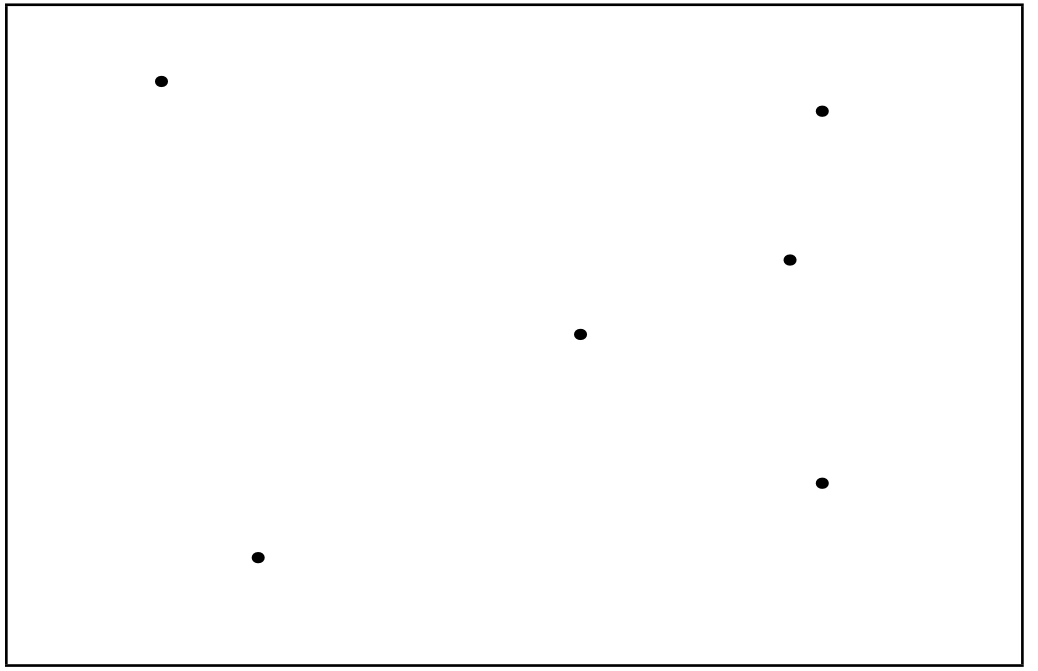
[1]*Graduate School of Engineering and Science, Shibaura Institute of Technology, Japan*
[2]*Department of Mathematical Science, Shibaura Institute of Technology, Japan*
[*]*nb15102@shibaura-it.ac.jp*

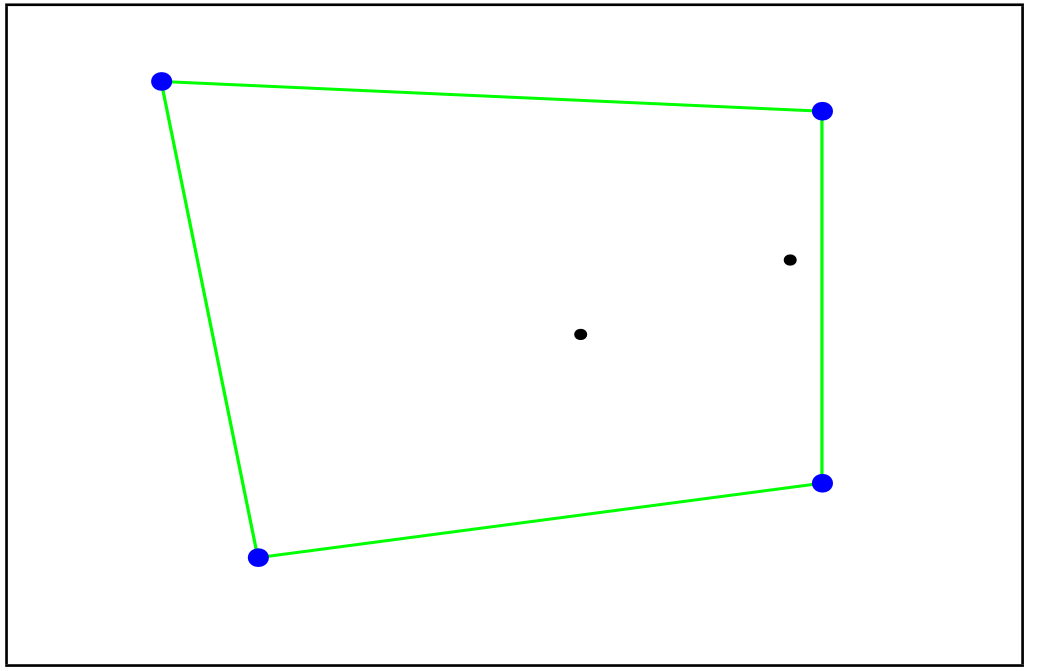
SWIM 2015 @ Charles University
June 10th, 2015

Start

(2D) Convex Hull

The convex hull of a set X of points is the smallest convex set that contains X. $\left(X = \left\{p_i \mid p_i \in \mathbb{R}^2,\ i = \{1, 2, \ldots, n\}\right\}\right)$





Introduction

- Constructing convex hull is very important
  - convex hull is applied many scientific computations
- Numerical computations
  - Very first
  - sometimes not exact

Introduction

This talk is concerned with verified numerical computations

- Computational geometry
  - Convex Hull (using 2D Orientation Problem)
- Error analysis of floating-point arithmetic
  - Floating-point filter

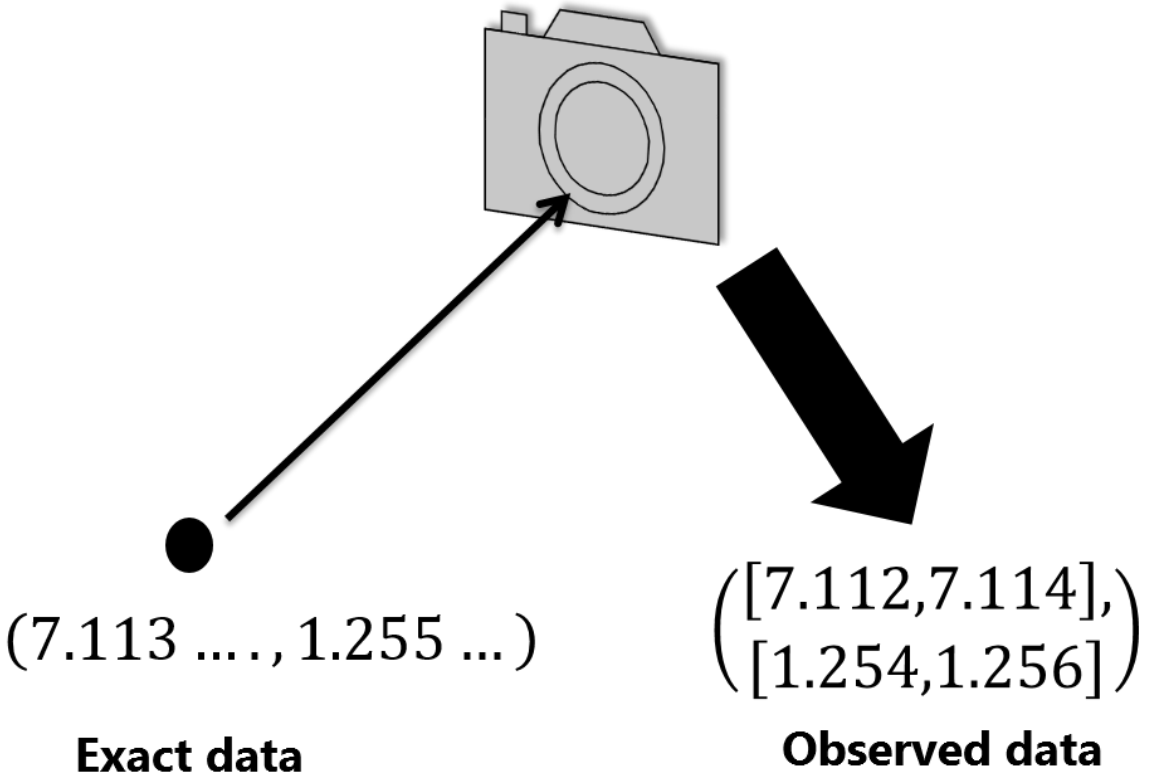
Introduction

It is assumed that input data are represented by floating-point numbers in many papers.

- The original data cannot always be represented by floating-point numbers.
- Some data include errors like observed data and computed results.

The concern is to obtain the exact or nearly exact convex hull.

Image of observed data



Even if we have the original data, observed data may not be represented by floating-point numbers.

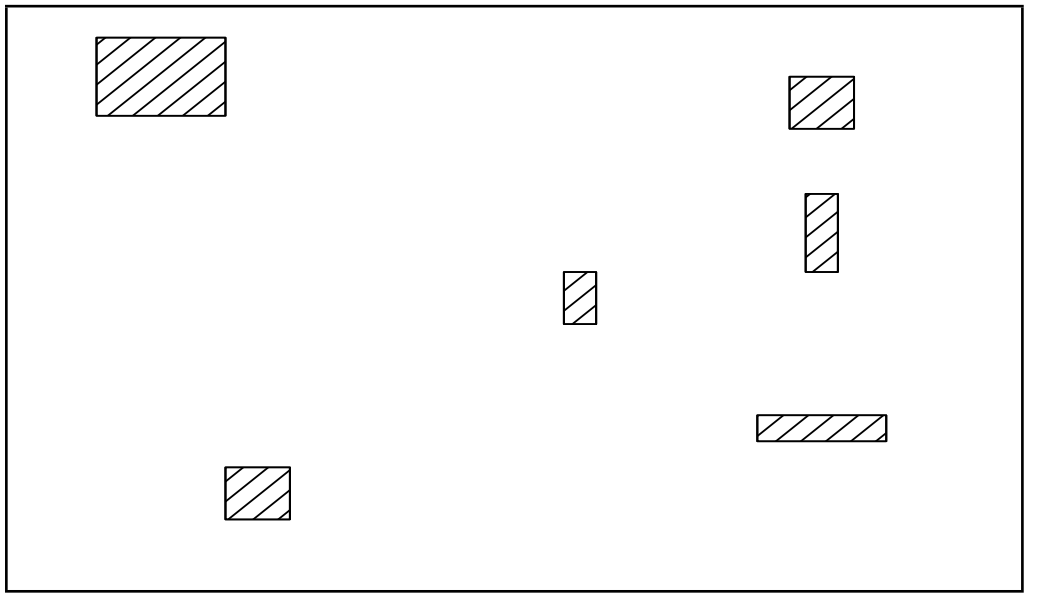
- observed data
  → interval data

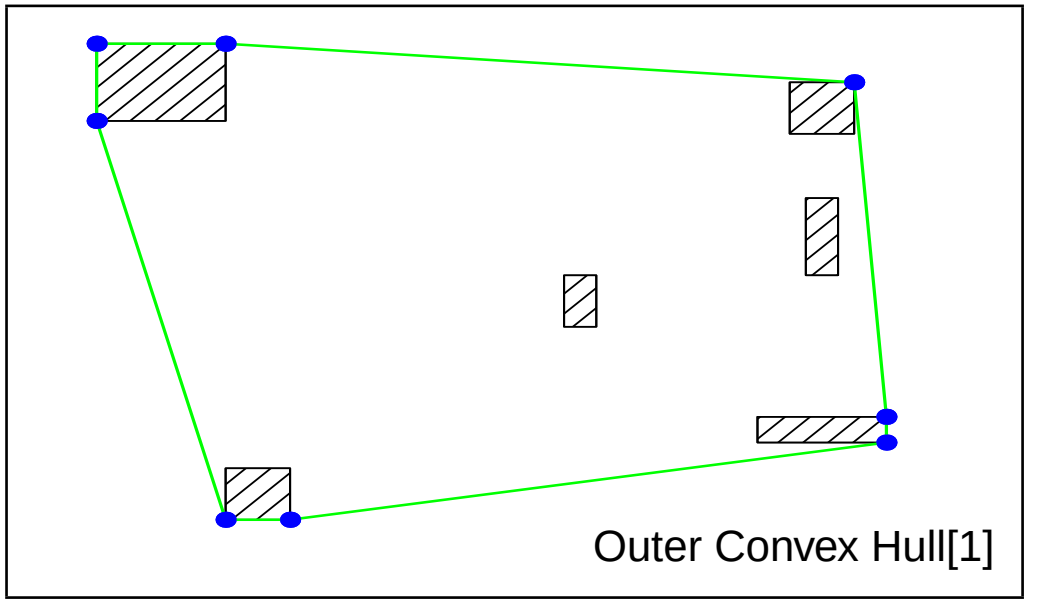
We treat input data
as “interval data”.

(2D) Convex Hull for interval data

Let $\mathbb{IR} = \{x \mid x_1 \leq x \leq x_2,\ x_1 \leq x_2,\ x, x_1, x_2 \in \mathbb{R}\}$.

The convex hull of a set X' of regions is the smallest convex that contains X'. $\left(X' = \left\{p'_i \mid p'_i \in \mathbb{IR}^2,\ i = \{1, 2, \ldots, n\}\right\}\right)$

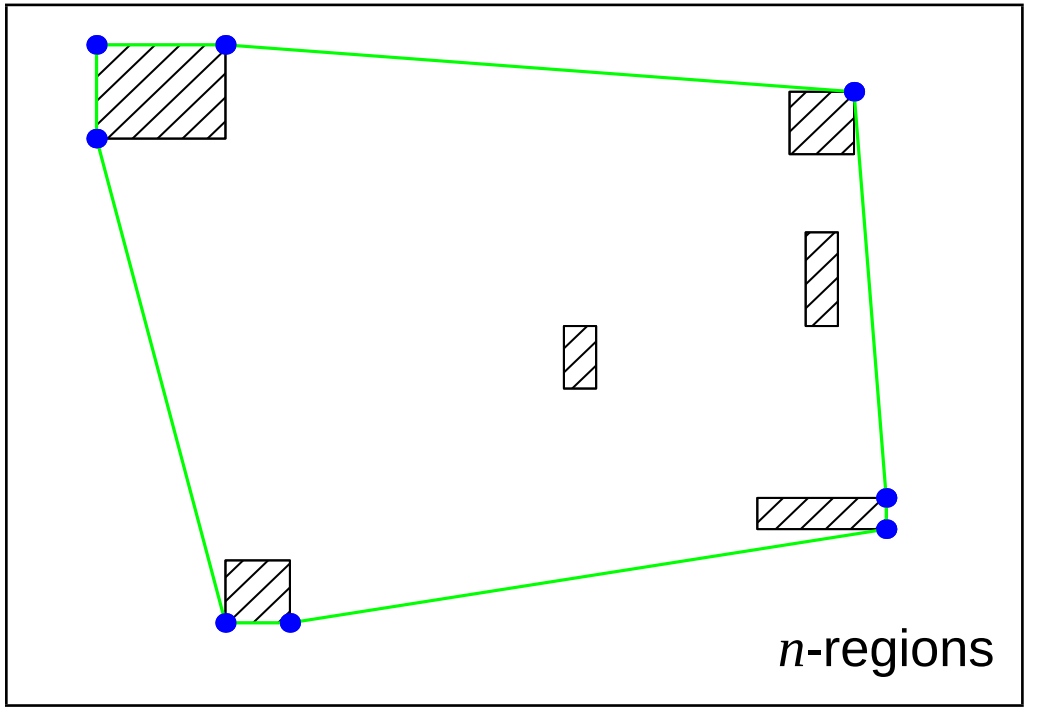


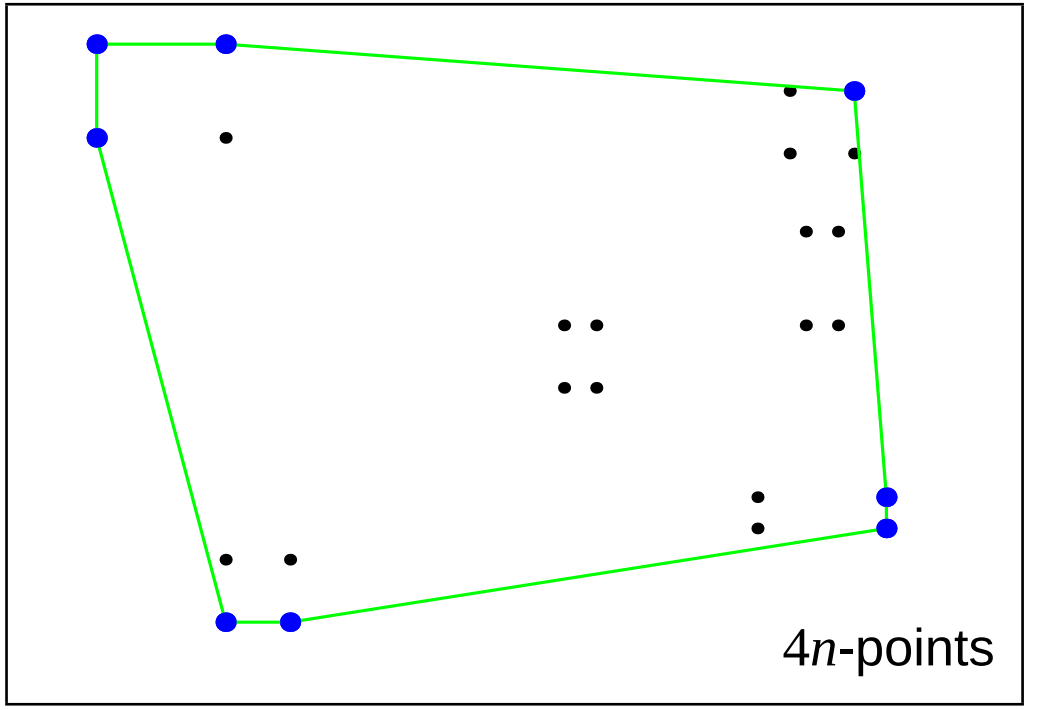


[1] T. Nagai, S. Yasutome, N. Tokura, Convex Hull Problem with Imprecise Input, Discrete and Computational Geometry, Vol. 1763, 2000, 207–219.

Convex hull for end-points

Convex hull for n-regions are convex hull for end-points of each regions





Convex hull for end-points

Convex hull for n-regions and convex hull for $4n$-points are same.

n is number of input points.
Left:computing time for n points
Right:computing time for $4n$ points
Items in table are computing time (sec)

We want to gain convex hull without using $4n$-points.
$\Rightarrow$ using floating-point filter

computing time to construct convex hull

n	n-points	$4n$-points
10^2	0.000044	0.000120
10^3	0.000062	0.000228
10^4	0.000539	0.002203
10^5	0.006401	0.027559
10^6	0.071787	0.294599
10^7	0.760685	3.139348

Outline

1. Introduction
2. 2D Orientation Problem
3. Our work
   - floating-point filters
     - floating-point number (previous research)
     - real number
     - interval data
   - iterative convex hull algorithm
4. Conclusion

2D Orientation Problem

Let $A'(a_x, a_y), B'(b_x, b_y), C'(c_x, c_y) \in \mathbb{R}^2$

- Assume that there is an **oriented line** passing from A' to B'.
- The problem is to clarify which the point C' is

This problem is boiled down to
the sign of the following 3-by-3 matrix determinant:

$$D' = \begin{vmatrix} a'_x & a'_y & 1 \\ b'_x & b'_y & 1 \\ c'_x & c'_y & 1 \end{vmatrix} = (a'_x - c'_x)(b'_y - c'_y) - (b'_x - c'_x)(a'_y - c'_y)$$

2D Orientation Problem for $\mathbb{R}^2$

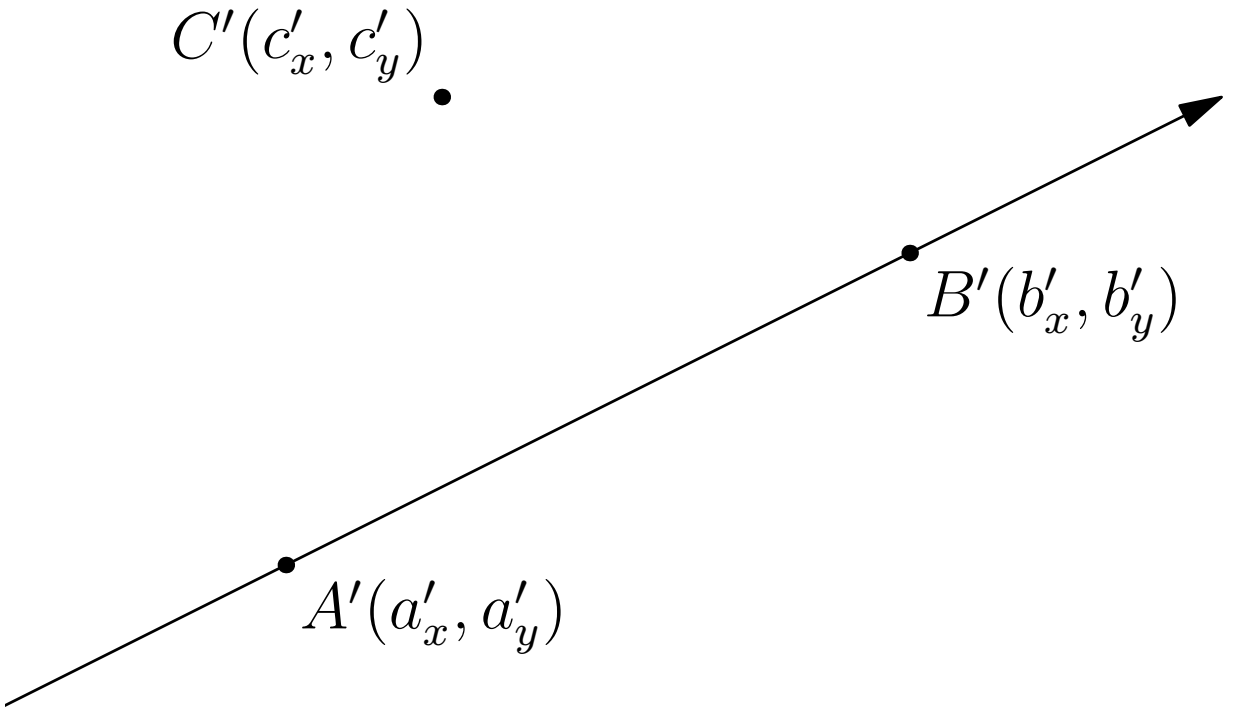
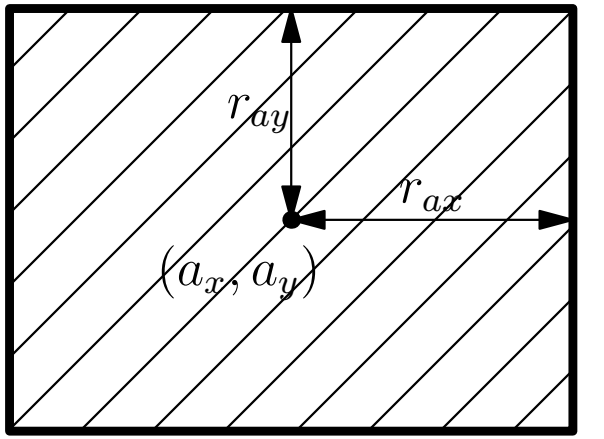


Figure 1: 2d orientation problem ($D' > 0$)

Notation

Let $\mathbb{F}$ be a set of floating-point numbers.

$\mathrm{fl}(\cdots)$: each operation in the parenthesis is evaluated by floating-point arithmetic



Let $D' := (a'_x - c'_x)(b'_y - c'_y) - (a'_y - c'_y)(b'_x - c'_x),$

$a_x = \mathrm{fl}(a'_x), \quad b_x = \mathrm{fl}(b'_x), \quad c_x = \mathrm{fl}(c'_x),$

$a_y = \mathrm{fl}(a'_y), \quad b_y = \mathrm{fl}(b'_y), \quad c_y = \mathrm{fl}(c'_y),$

$D := (a_x - c_x)(b_y - c_y) - (a_y - c_y)(b_x - c_x),$

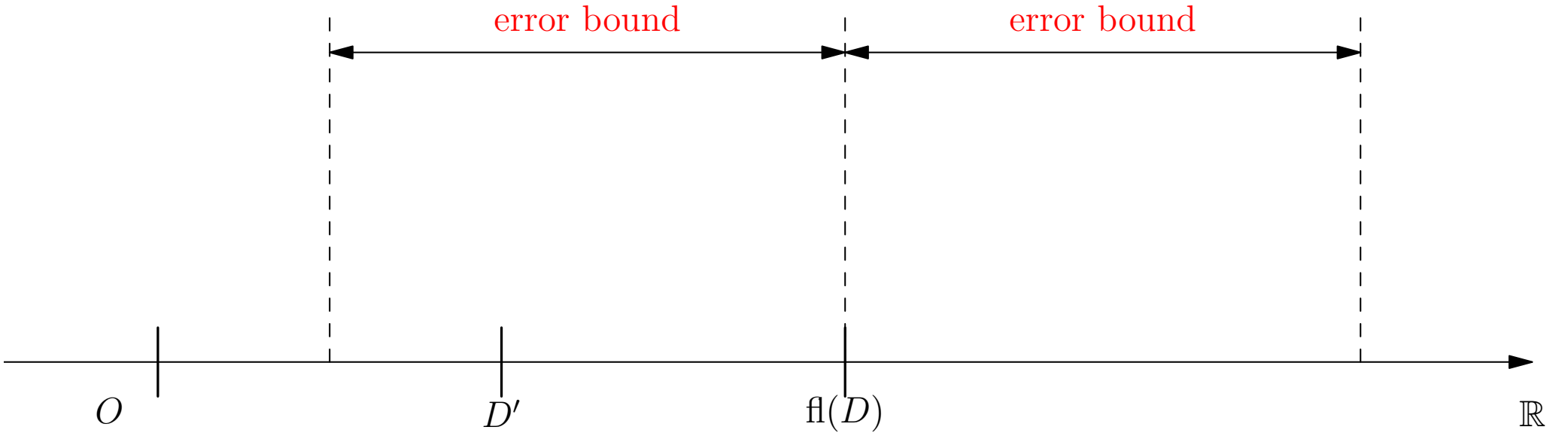
$\mathrm{fl}(D) := \mathrm{fl}\left((a_x - c_x)(b_y - c_y) - (a_y - c_y)(b_x - c_x)\right),$

$r_{ax}, r_{ay}, r_{bx}, r_{by}, r_{cx}, r_{cy} \in \mathbb{F}$

Image of floating-point filter

If $|\mathrm{fl}(D)|$ is grater than error bound of $\mathrm{fl}(D)$, then $\mathrm{sign}(\mathrm{fl}(D))$ is guaranteed.

Example: Case of guaranteeing $\mathrm{sign}(\mathrm{fl}(D))$



D': original determinant, $\mathrm{fl}(D)$: approximate determinant

Floating-point Filter (previous research)

Shewchuck proposed filters for points inputs[2].
If the following inequality is satisfied, $\mathrm{sign}(D)$ **and** $\mathrm{sign}(\mathrm{fl}(D))$ are the same.

$$\mathrm{fl}\,(|D|) > \mathrm{fl}\left((3\mathbf{u}+16\mathbf{u}^2)\left(|(a_x-c_x)(b_y-c_y)|+|(b_x-c_x)(a_y-c_y)|\right)\right)$$

$\mathbf{u}$: the relative rounding error unit

For example, $\mathbf{u}$ is 2^{-53} for binary64.

[2] J. R. Shewchuk : Adaptive precision floating-point arithmetic and fast robust geometric predicates, Discrete & Computational Geometry, Vol.18, 1997, 305–363.

Floating-point Filter (for real number)

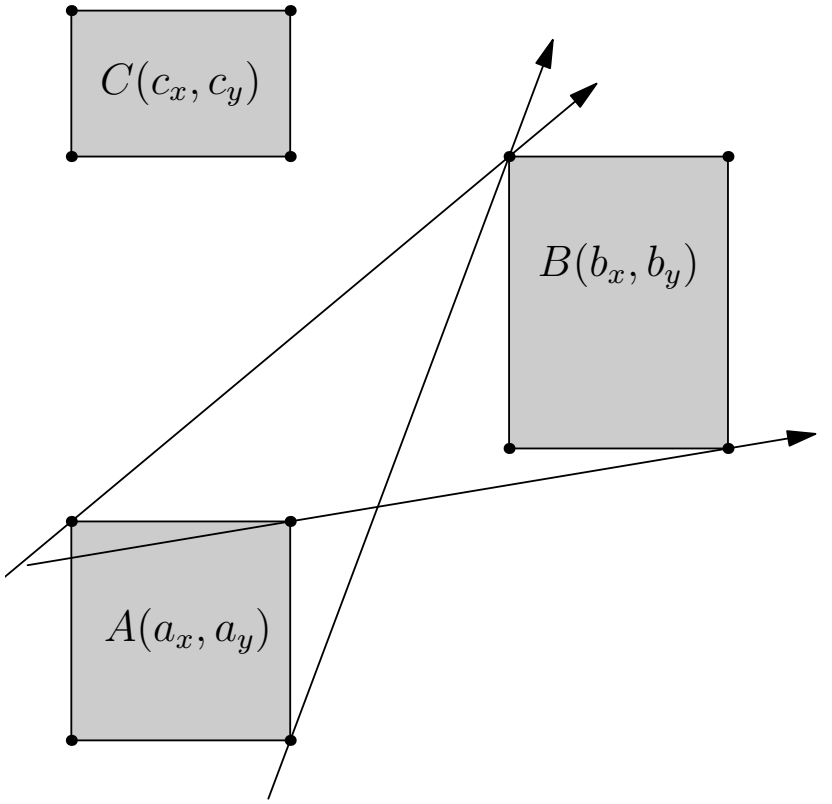
If the following inequality is satisfied, $\mathrm{sign}(D')$ **and** $\mathrm{sign}(\mathrm{fl}(D))$ are the same.

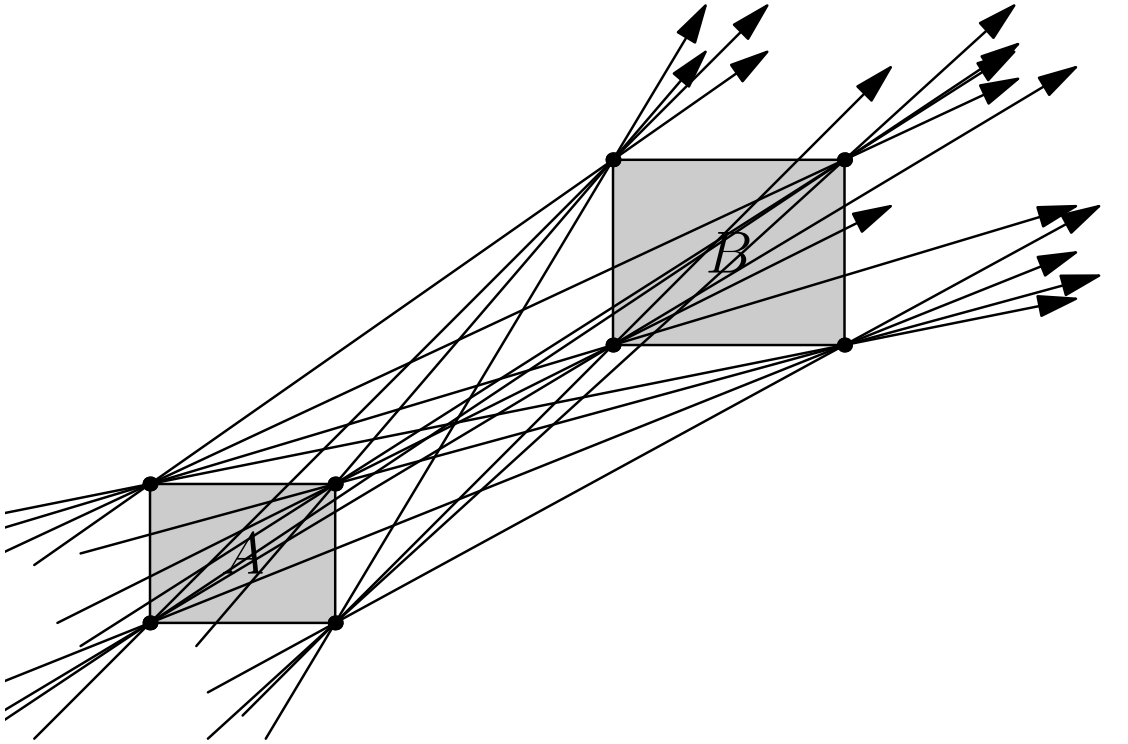
$$\begin{aligned}\mathrm{fl}(|D|) > \mathrm{fl}\Big(&(3\mathbf{u}+24\mathbf{u}^2)\left(|(a_x-c_x)(b_y-c_y)|+|(b_x-c_x)(a_y-c_y)|\right)\\ &+(2\mathbf{u}+24\mathbf{u}^2)\left((|a_x|+|c_x|)(|b_y|+|c_y|)+(|b_x|+|c_x|)(|a_y|+|c_y|)\right)\\ &+\mathbf{u}_N\left(((|a_x|+|c_x|)+(|b_y|+|c_y|))+((|b_x|+|c_x|)+(|a_y|+|c_y|))\right)+\mathbf{u}_N\Big)\end{aligned}$$

$\mathbf{u}$: the relative rounding error unit
$\mathbf{u}_N$: the minimum number of all positive normalized floating-point numbers

For example, $\mathbf{u}$ is 2^{-53} and $\mathbf{u}_N$ is 2^{-1022} for binary64.

Floating-point Filter (for interval data)





Floating-point Filter (for interval data)

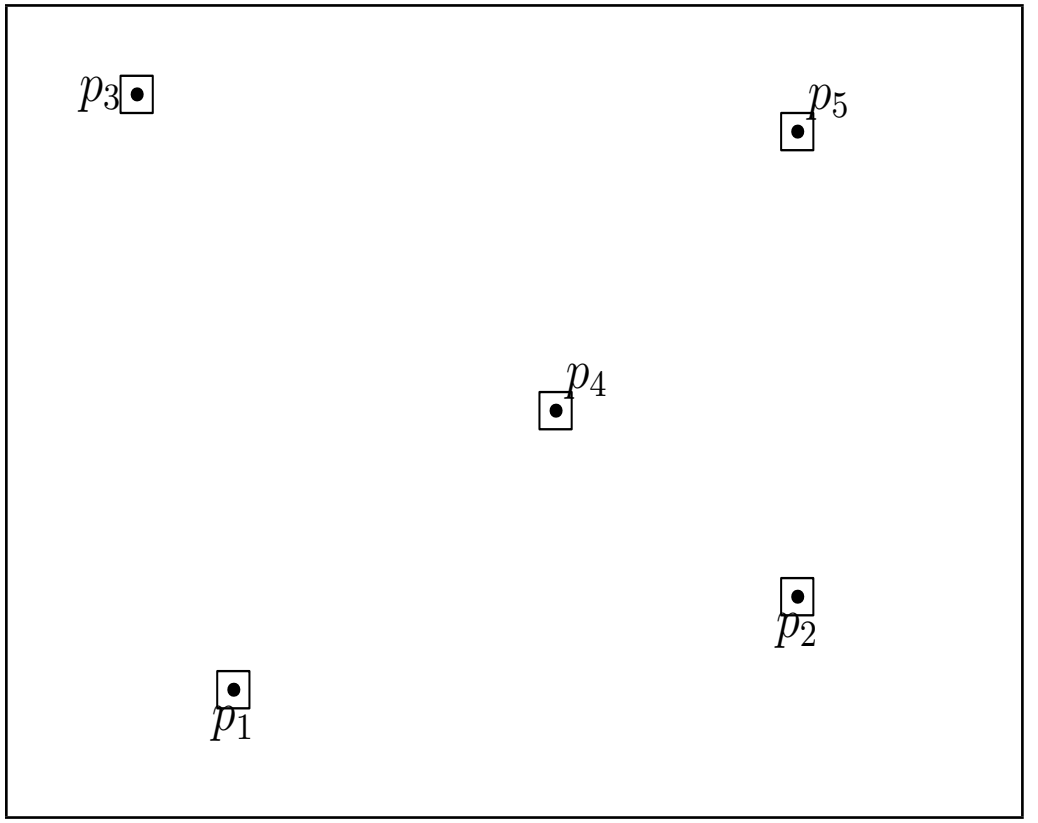
If the following inequality is satisfied, $\mathrm{sign}(D)$ **and** $\mathrm{sign}(\mathrm{fl}(D))$ are same.

$$\begin{aligned}\mathrm{fl}(|D|) > \mathrm{fl}\Big((3\mathbf{u}+16\mathbf{u}^2)\big(|(a_x-c_x)(b_y-c_y)|+|(b_x-c_x)(a_y-c_y)|\big)\\ +(1+10\mathbf{u})\big((|a_x|+|c_x|)(r_{by}+r_{cy})+(r_{bx}+r_{cx})(|a_y|+|c_y|)\big)\\ +(1+10\mathbf{u})\big((r_{ax}+r_{cx})(|b_y|+|c_y|)+(|b_x|+|c_x|)(r_{ay}+r_{cy})\big)\\ +(1+10\mathbf{u})\big((r_{ax}+r_{cx})(r_{by}+r_{cy})+(r_{bx}+r_{cx})(r_{ay}+r_{cy})\big)\Big)\end{aligned}$$

$\mathbf{u}$: the relative rounding error unit

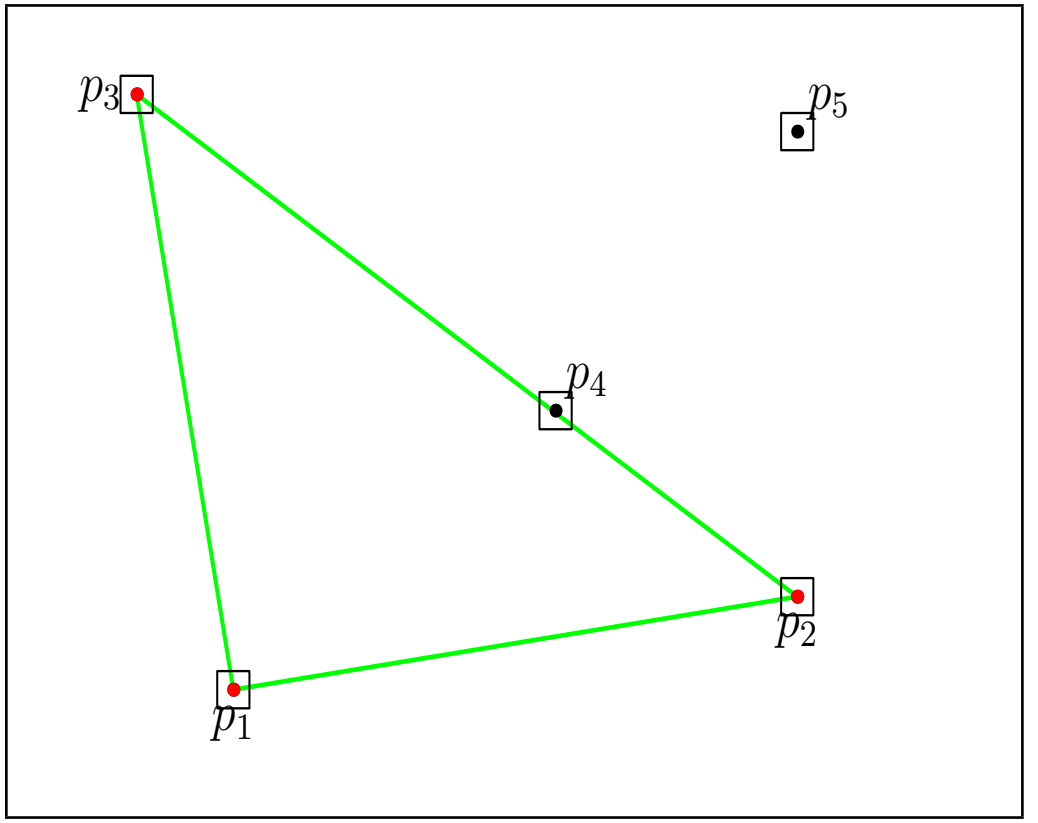
For example, $\mathbf{u}$ is 2^{-53} for binary64.

Convex Hull Algorithm



Case of using incremental convex hull algorithm

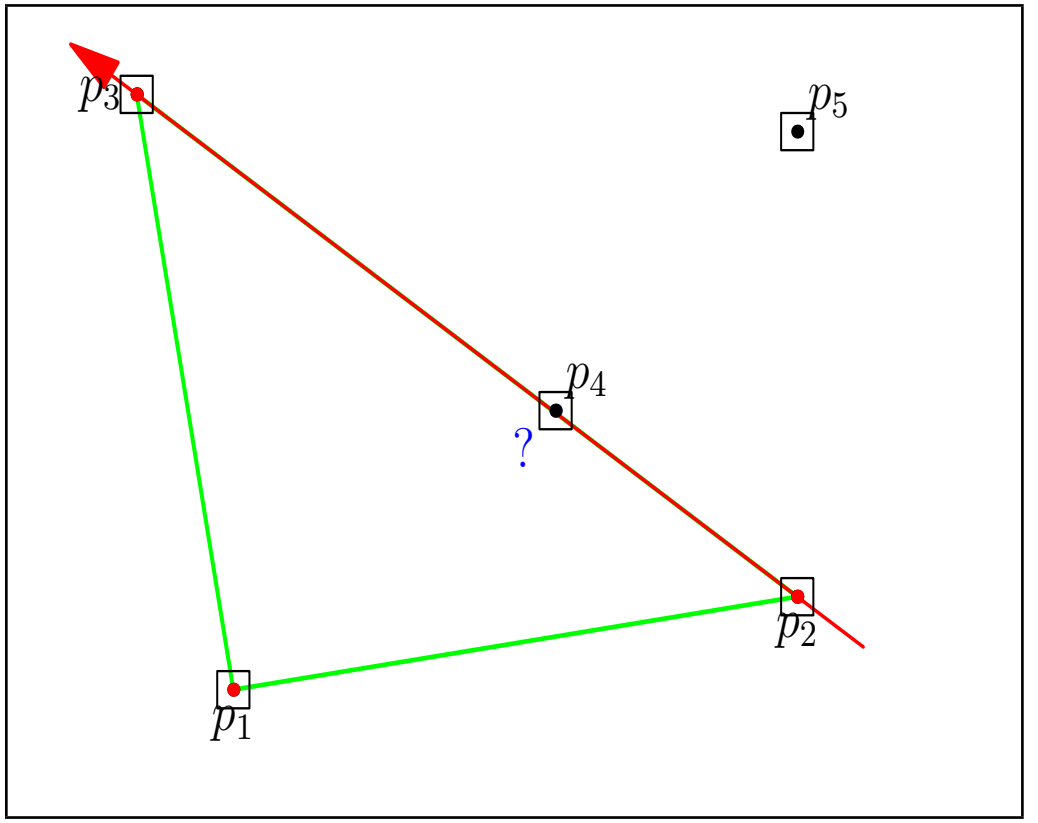
Convex Hull Algorithm



The green triangle is the convex hull for p_1, p_2 and p_3.

Next, we discuss which p_4 is inside or outside of the convex hull.

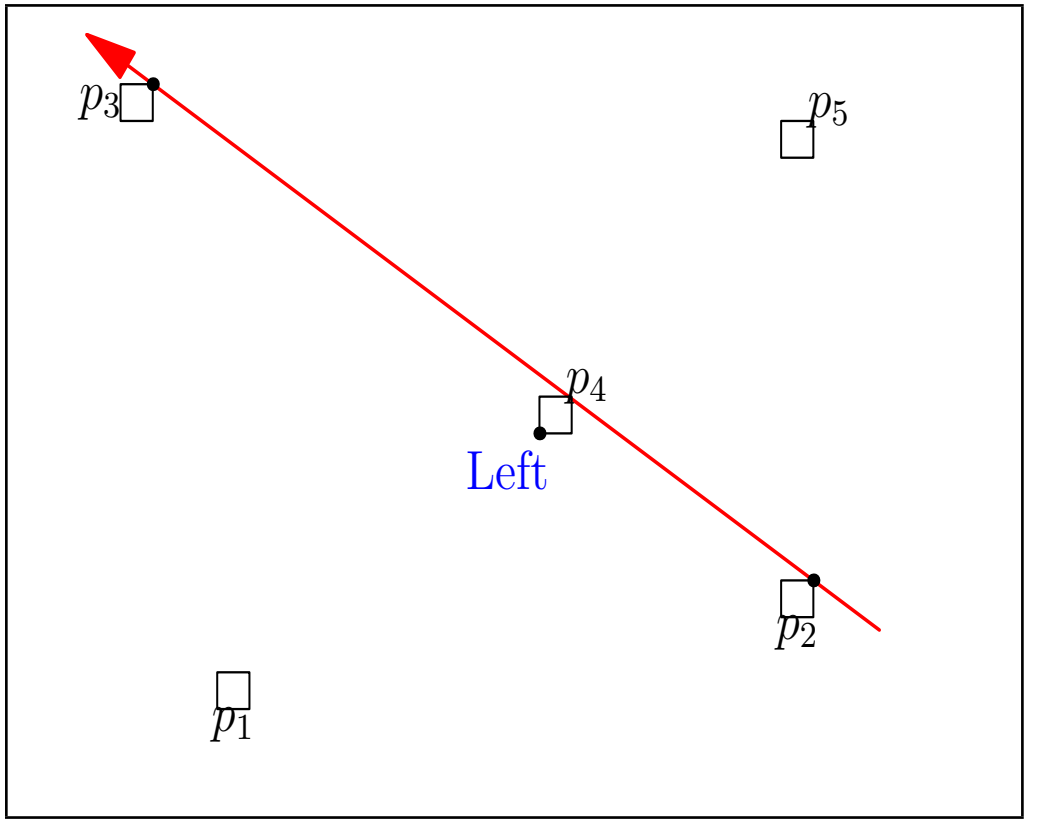
Convex Hull Algorithm

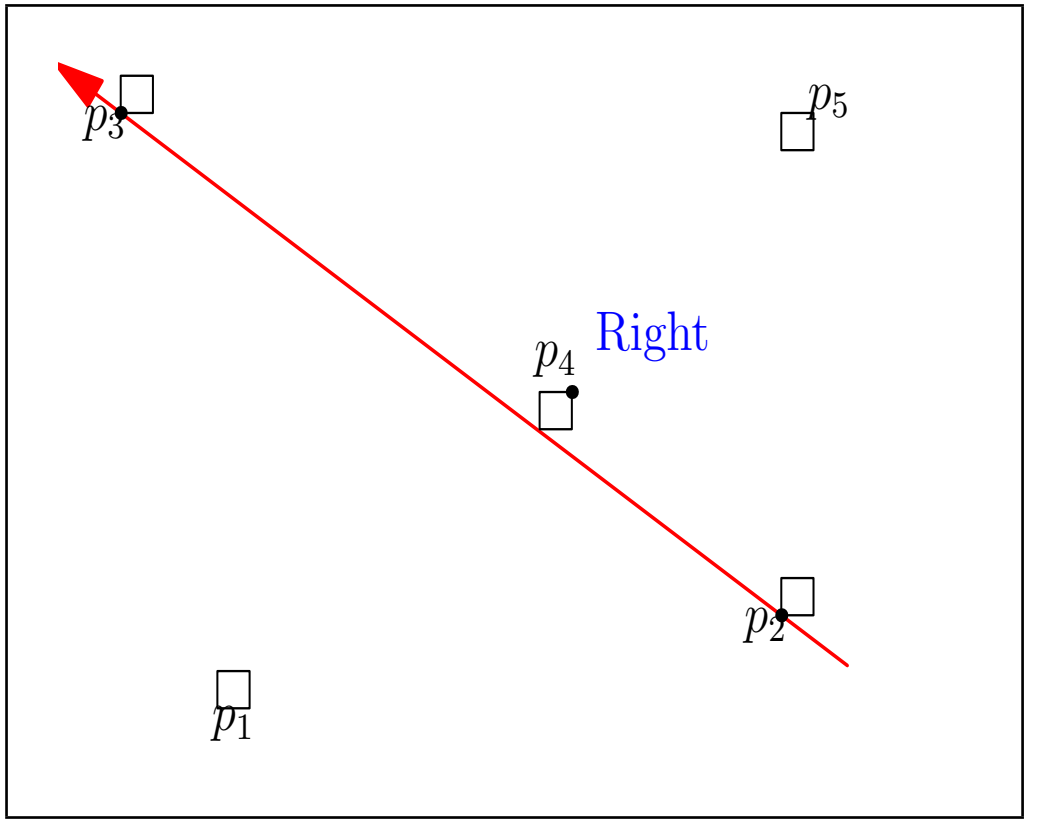


p_4 is very close to the oriented line p_2p_3.

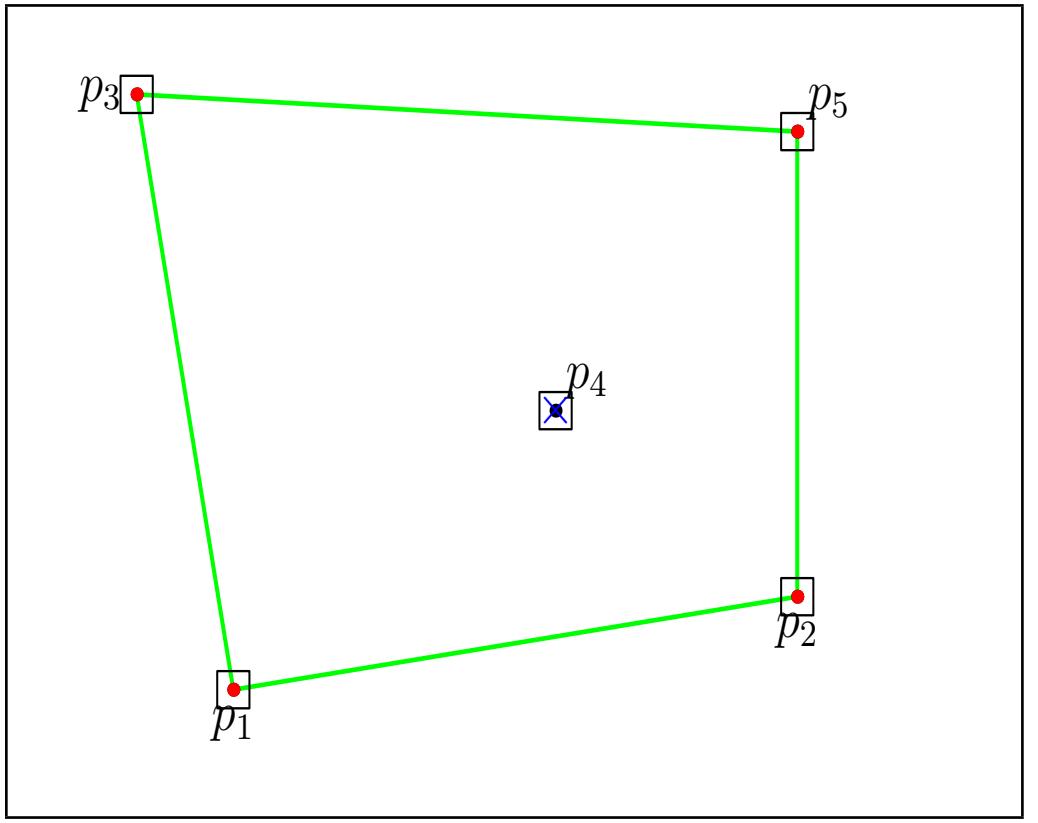
Then, floating-point filter cannot guarantee which p_4 is left or right.

Convex Hull Algorithm





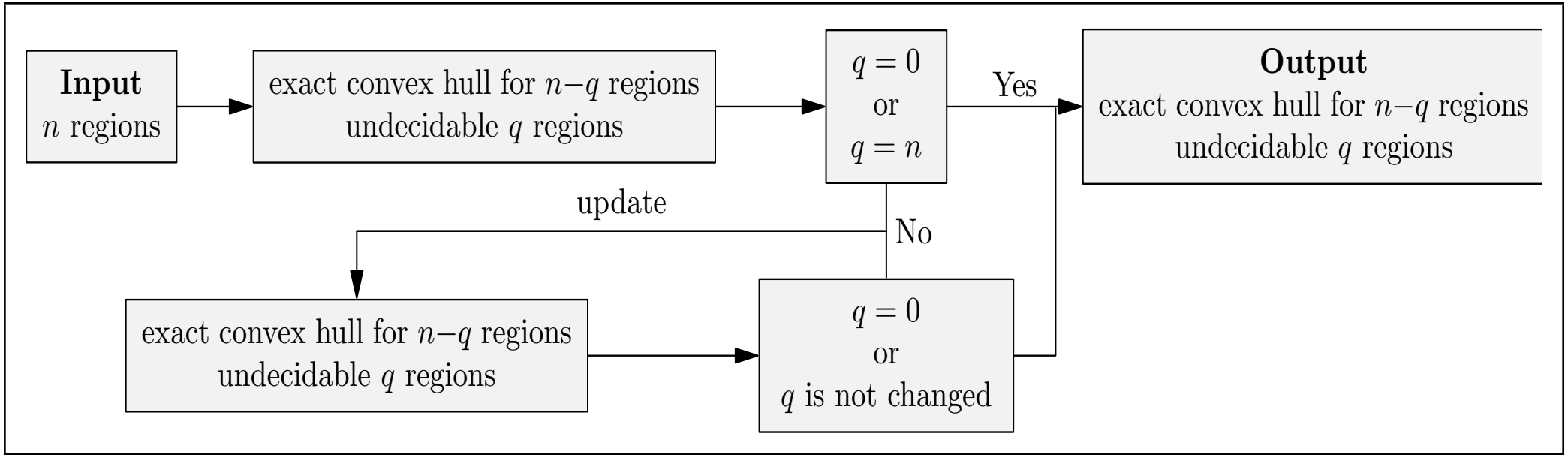
Convex Hull Algorithm

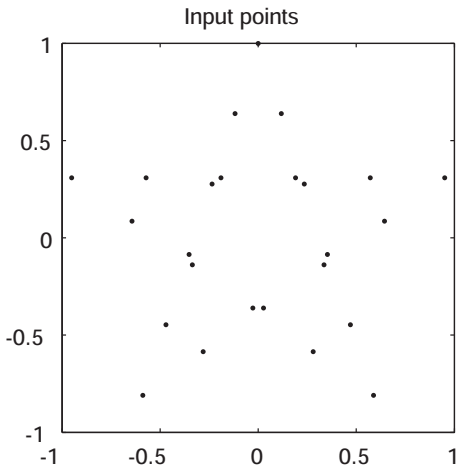


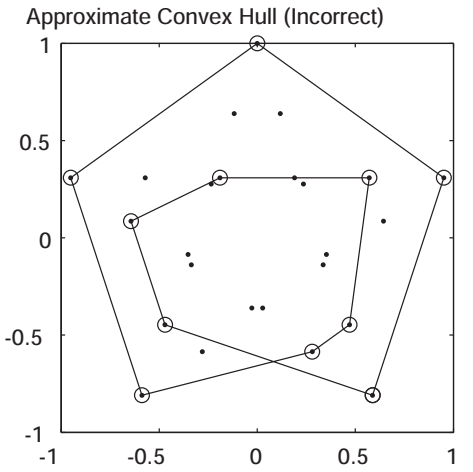
We skip to clarify which p_4 is.

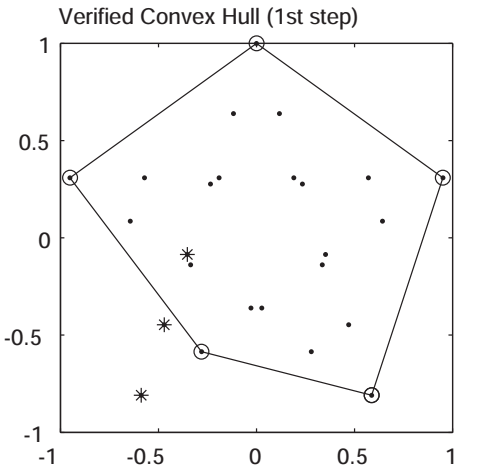
Our strategy is to skip this estimation and we discuss this problem later.

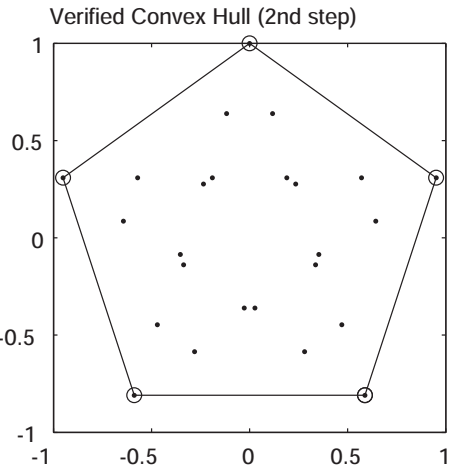
In lucky case, it is easy to judge that p_4 is inside of final convex hull.


Input
n regions
exact convex hull for $n-q$ regions
undecidable q regions
$q = 0$
or
$q = n$
Yes
Output
exact convex hull for $n-q$ regions
undecidable q regions
update
No
exact convex hull for $n-q$ regions
undecidable q regions
$q = 0$
or
q is not changed


Input points


Approximate Convex Hull (Incorrect)


Verified Convex Hull (1st step)


Verified Convex Hull (2nd step)

Numerical Experiment

There are 2 methods to construct convex hull for interval data.

- (F1)
  n-regions $\Rightarrow$ $4n$-points $\rightarrow$ convex hull

- (F2)
  n-regions $\rightarrow$ h-regions for convex hull and undecidable q-regions
  $\Rightarrow$ $4h$-points and $4q$-points $\rightarrow$ convex hull
  ($h + q \leq n$)

Image of (F1)

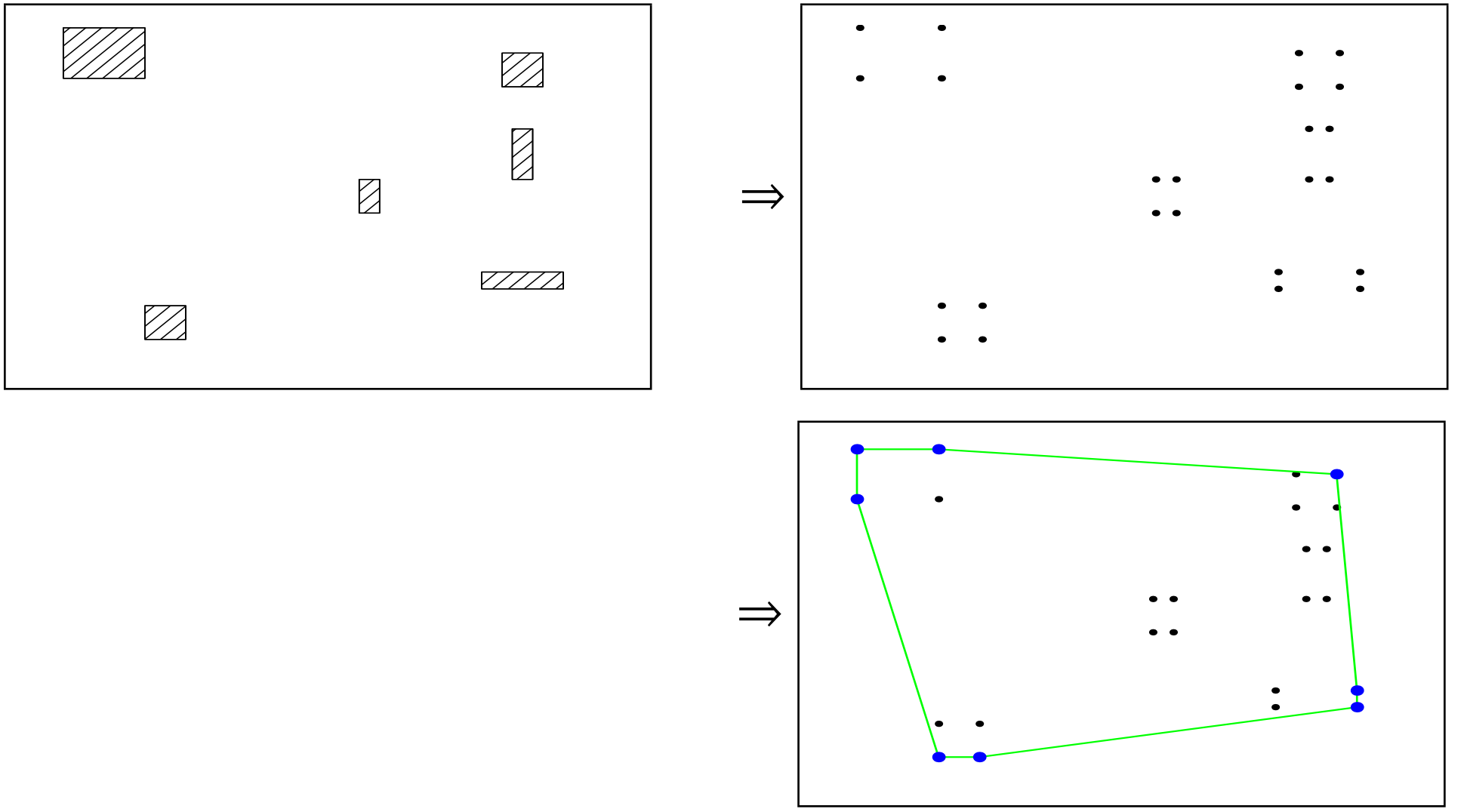
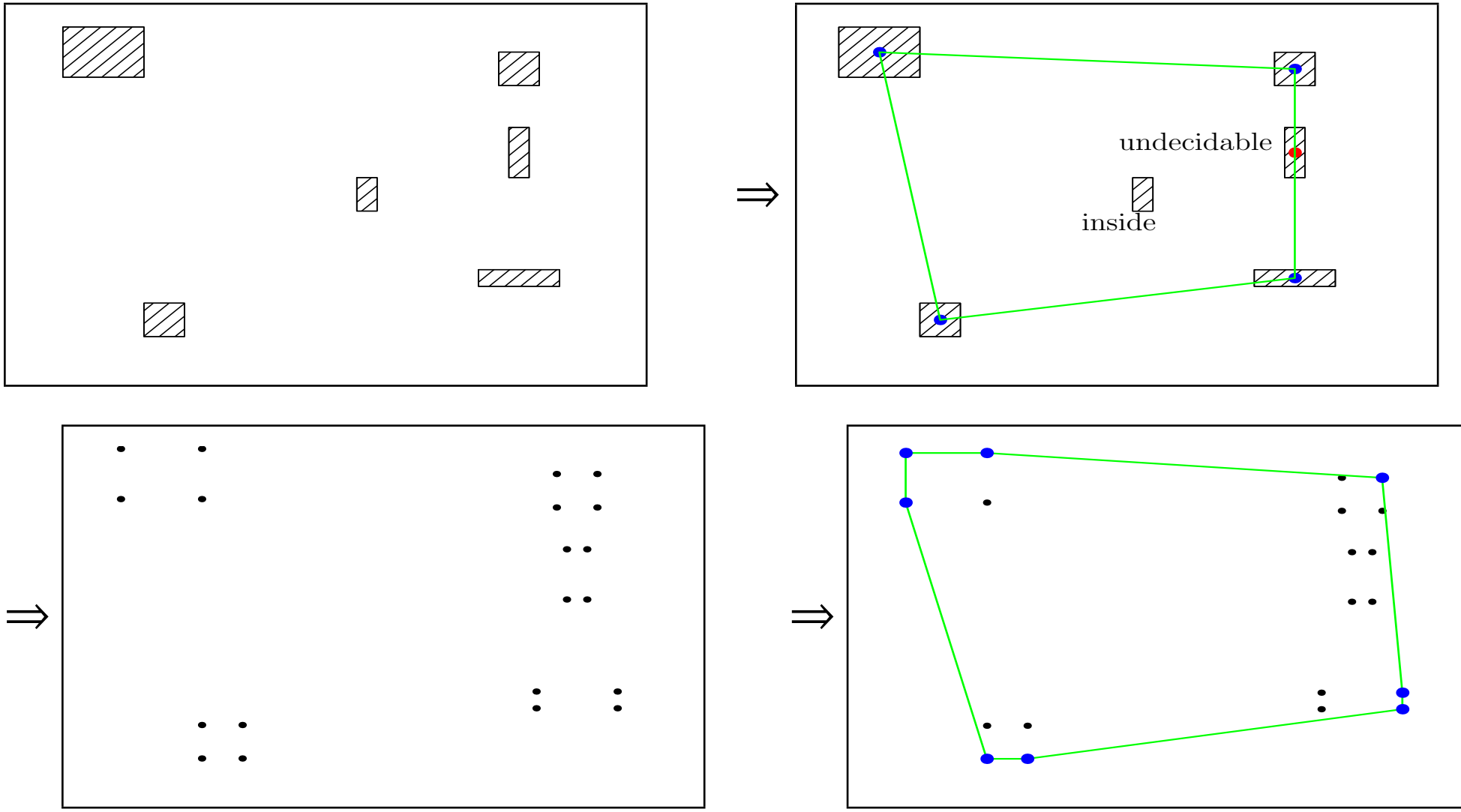


Image of (F2)



Test (well-conditioned)

Using “randn” for middle of interval data, $0.1*$“rand” for radius of interval data (average of 100 times)

n	(F1)	(F2)
10^2	0.000049	0.000072
10^3	0.000474	0.000227
10^4	0.006206	0.002089
10^5	0.084369	0.019253
10^6	0.887855	0.238547
10^7	9.759882	2.516800

(F1):
Use our algorithm for end-point data
(F2):
Use our algorithm for interval data → end-point data

Computational environment

OS：Windows 7 Professional

CPU：Intel(R) Core(TM) i7-4600U CPU 2.10 GHz 2.70 GHz

MATLAB：R2014a

Items in table are computing time(sec).

Conclusion

We discussed how to derive convex hull
for general interval data by numerical computation.

- Floating-point filter for 2D orientation problem
- iterative convex hull algorithm

Thank you very much for your attention.