Exact Geometric Computation *EGC*

Richtiges Berechnen von Geometrischen Prädikaten mittels Determinanten

Li Chaoran Martin Rohde

Universität Salzburg – PS Wissenschaftliche Präsentation und Arbeitstechniken

3. Februar 2007

Übersicht

Problembeschreibung

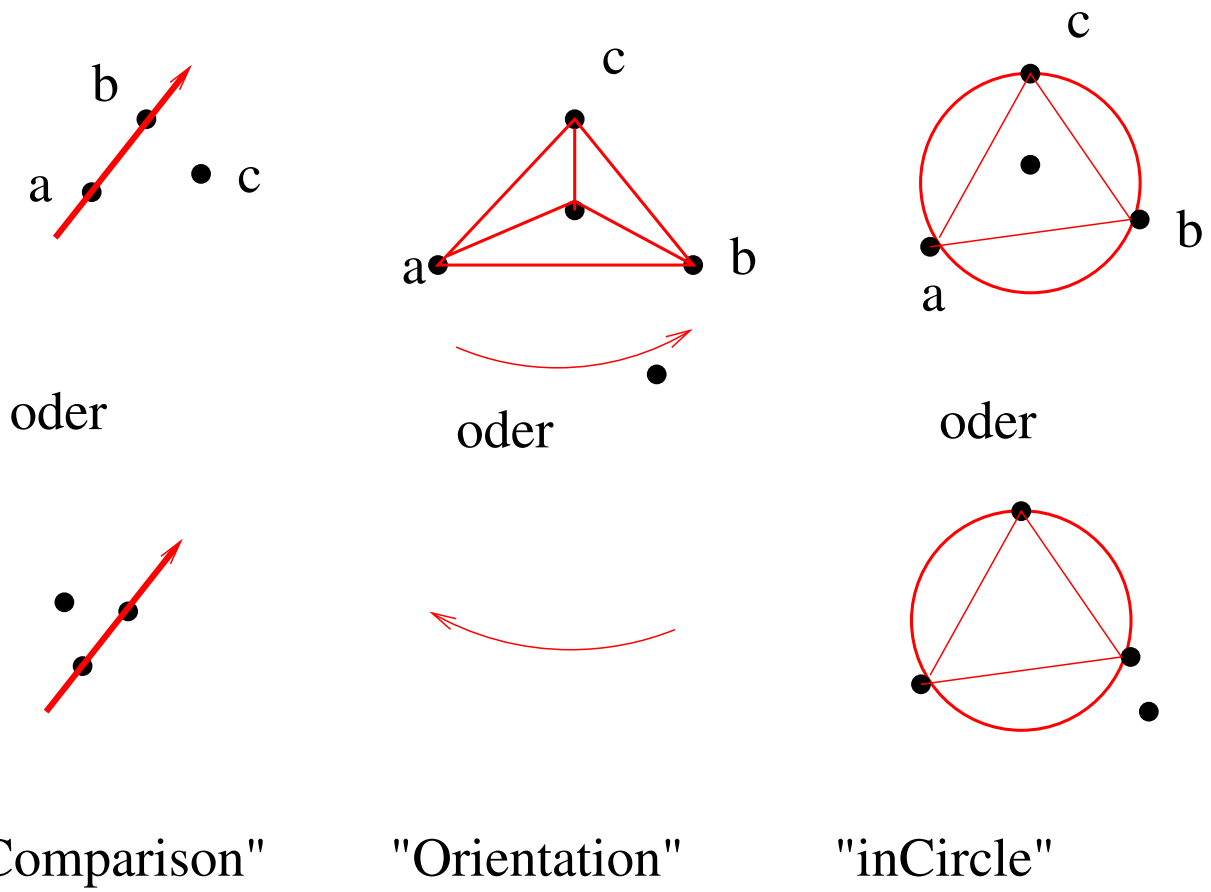


Abbildung: combinatorics and predicates, entnommen aus: Robustness and degeneracies,Oliver Devillers,adaptiert

Determinanten

- Comparison
$$\begin{Vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{Vmatrix} = \begin{Vmatrix} a_x - c_x & a_y - c_y \\ b_x - c_x & b_y - c_y \end{Vmatrix}$$
- Orientation
$$\begin{Vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{Vmatrix} = \begin{Vmatrix} a_x - c_x & a_y - c_y \\ b_x - c_x & b_y - c_y \end{Vmatrix}$$
- InCircle
$$\begin{Vmatrix} a_x & a_y & a_x^2 + a_y^2 & 1 \\ b_x & b_y & b_x^2 + b_y^2 & 1 \\ c_x & c_y & c_x^2 + c_y^2 & 1 \\ d_x & d_y & d_x^2 + d_y^2 & 1 \end{Vmatrix} =$$
$$\begin{Vmatrix} a_x - d_x & a_y - d_y * (a_x - d_x^2) + (a_y - d_y^2) \\ b_x - d_x & b_y - d_y * (b_x - d_x^2) + (b_y - d_y^2) \\ c_x - d_x & c_x - d_y * (c_x - d_x^2) + (c_y - d_y^2) \end{Vmatrix}$$

Determinanten

	... > 0	... $= 0$	... < 0
"InCircle"	innerhalb	Rand	außerhalb
"Orientation"	gegen den Uhrzeigersinn	auf der Geraden	im Uhrzeigersinn
"Comparison"	linke Seite	auf der Geraden	rechte Seite

IEEE Floating Point Standard

- 32 Bit IEEE Floating Point Format

Abbildung: 32 Bit IEEE Floating Point Format

$q_{(10)} * 2^{e-127} * (1.M)$

- 64 Bit IEEE Floating Point Format (genauer,aber immer noch ungenau)

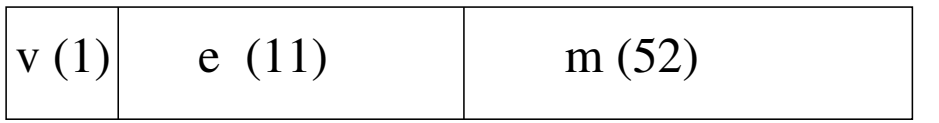


Abbildung: 64 Bit IEEE Floating Point Format

$q_{(10)} * 2^{e-1023} * (1.M)$

Beispiel

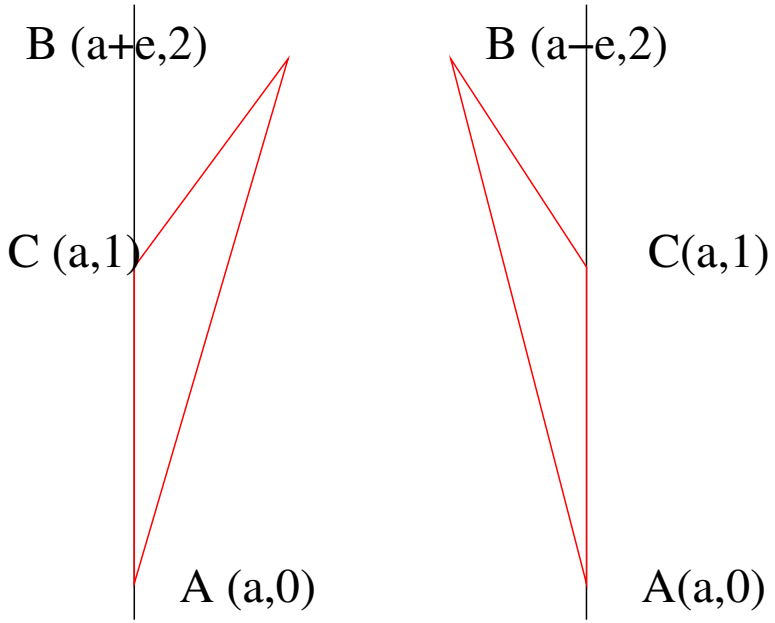


Abbildung: Geometric predicates implemented in FP can fall;entnommen aus: Robust Predicates and Degenerancy,Marek Teichmann, adaptiert

$$a = 1; e = 2^{-999999} \quad |orientation| = \begin{Vmatrix} 1 & 0 & 1 \\ 1+e & 2 & 1 \\ 1 & 1 & 1 \end{Vmatrix} = \begin{Vmatrix} 1-1 & 0-1 \\ 1+e-1 & 2-1 \end{Vmatrix}$$

Lösungsansätze

beliebige Genauigkeit

"so genau wie möglich"

- IEEE 32 Bit Floating Point
- IEEE 64 Bit Floating Point
- Zahl als tuppel $(a_1, a_2, ...a_n)$
  aus 32 bzw 64 Bit Blöcken:

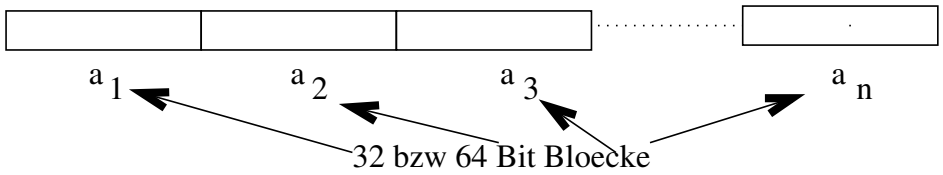


Abbildung: beliebige Genauigkeit

- vorgefertigte Bibliotheken:
  Leda (www.algorithmic-solutions.com/index.htm)
  CGAL (Computational Geometric Algorithms Library, www.cgal.org)

Algebraisch

- Transzendente Zahlen (zB. $\pi, \sqrt{2}$):
  zB.Prädikat:

$$P(x)\ wahr\ mit\ x = sqrt(2)$$

  falls

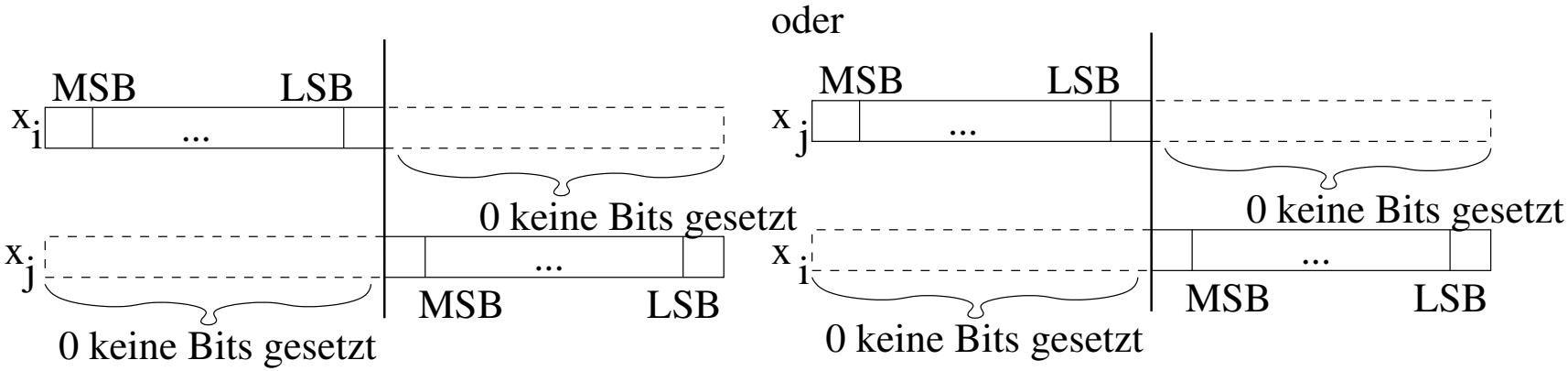
$$x^2 - 2 = 0$$

  Stichwort: Symbolisches Rechnen

Priest und Schewchuk

$$x = x_n + ... + x_2 + x_1 \ ; \ x_i \ \textit{vom Typ Floating Point}$$

- non overlapping (nicht überlappend)



nur Festkomma–Zahlen

Abbildung: Nichtüberlappen von zwei Binärzahlen

- Beispiele: 1100 -10,1 nicht überlappend
  101 10 überlappend
- Vorzeichen durch größte Komponente bestimmt

Priest und Schewchuk

- einheitliche Zahlendarstellung nicht notwendig
- Fehler absichtlich zulassen: a,b p-bit floating Point Zahlen

$$|a| \geq |b|$$

Ergebnis: $a + b = x + y$,x Approximation, y Rundungsfehler
wichtig $x > y$ nicht überlappend

Priest und Schewchuk

- lazy evaluation (faule Berechnung)

$$E = (a_x - b_x)^2 + (a_y - b_y)^2$$

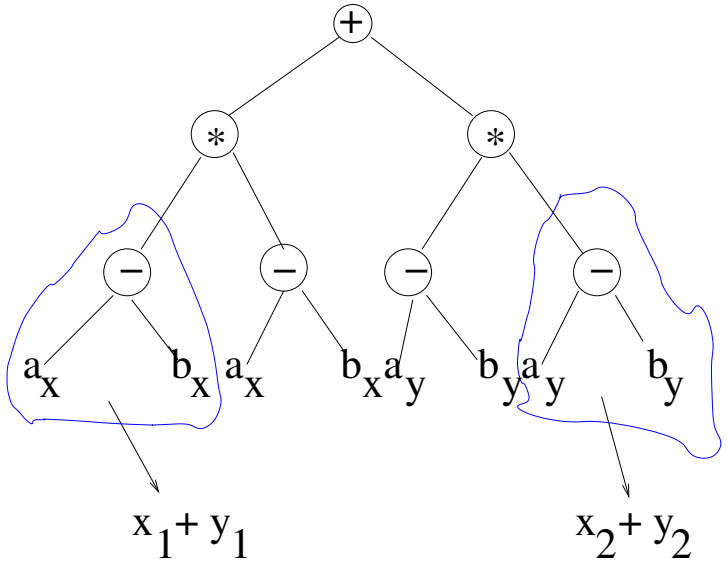


Abbildung: Expression Tree, entnommen aus: Robust Adaptive Floating Point Geometric Predicates,J.R. Shewchunk,adaptiert

$$= (x_1 + y_1)^2 + (x_2 + y_2)^2$$
$$= (x_1^2 + x_2^2) + (2x_1y_1 + 2x_2y_2) + (y_1^2 + y_2^2)$$

Approximation (S. Fortune, V. Wyk, C.Burnikel, S.Funke, M.Seel)

- $|x - x'| \leq err$
- x =E

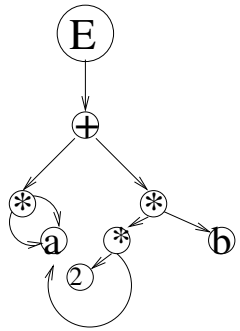


Abbildung: Expression Graph for $E = a^2 + 2 * a * b$,entnommen aus The Exact Computation Paradigm,adaptiert

- 1. falls x = x' kein E wird berechnet.
- 2. Bestimme q_x = Separationsgrenze(datentypabhängig): $|x| < q_x$ dann x = 0 ;
- 3.Intervallbestimmung für $[a, x, b]$ so dass $[a, x, 0)$ oder $(0, x, b]$ oder [0] wenn $[x - err, x + err] < q_x$

Approximation

- Fehler-Grenze berechnen(reicht diese für Berechnung des Vorzeichens aus ?)

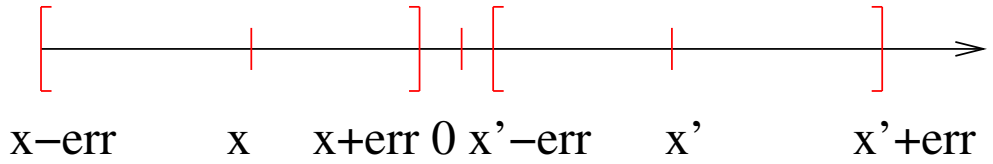


Abbildung: Fehlerschranke

- fully dynamic(Fehler-Grenze vollständig zur Laufzeit bestimmt)
- relative Fehlergrenzen

$$|e' - e| \leq \varepsilon_e * |e'|$$

hoher zusätzlicher Aufwand

Approximation

- semi static (halb statisch)

$$|e' - e| \leq e'_{sup} * ind_e * 2^{-p}$$

  $e'_{sup} = |e'|$ *dynamisch*
  ind_e statisch berechnet

- statisch
  $bitlength_{+-} = 1 + max(bitlength_{op_1} + bitlength_{op_2})$
  $bitlength_{*} = bitlength_{op_1} * bitlength_{op_2}$

Topologische Lösungen

- numerischer Teil, kombinatorischer Teil
- Dreieck Richtung berechnen:

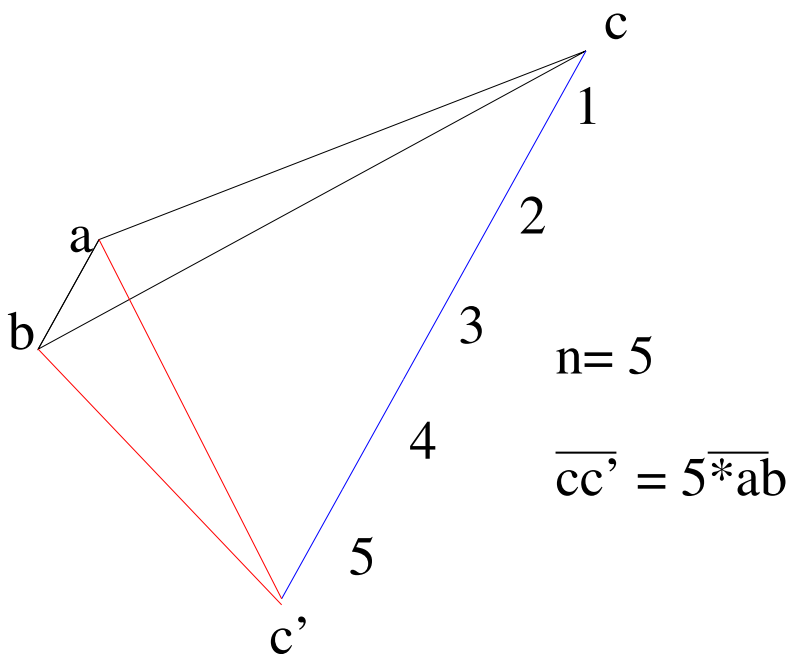


Abbildung: Geometrical exact computation, entnommen aus: Arithmetic for geometric computation, Olivier Devillers ,adaptiert

- $\Leftarrow$ Verzerrung , keine exakte Lösung,
  schnelle Berechnung

verwendete Literatur

1. Chee Yap, Kurt Mehlhorn,"Towards Robust Geometric Computation",Max-Planck Institute of Computer Science,June 2001
2. Steven Fortune, J. Van Wyk ," Statistic Analysis Yield Efficient Exact Integer Arithmetic for Computional Geometric ",ACM , July 1996
3. Christoph Burnikel,Stefan Funke,Michael Seel ," Exact Geometric Computation using Cascadading",International Journal of Computational Geometry and Applications
4. Stefan Schirra ," Invited Lecture: Real Numbers and Robustness in Computational Geometry"," Real Numbers and Computers 6,7-21",nov 2004
5. Jonathan Richard Shewchuk ," Rboust Adaptive Floating-Point Geometric Predicates"," School of Computer Science Carnegie Mellon University"
6. K.Sugihara,M. Iri,H.Inagaki, T. Imai ," Topology-oriented implementation- an approach to robust geometric algorithms ," Algorithmica ",27(1):5-20 , 2000

verwendete Literatur

1. O.Devillers," Arithmetic for geometric computation" ,http://www.sop.inria.fr/prisme/fiches/Arithmetique/index.html.en,1999
2. Marek Teichmann," Robust Predicates and Degeneracy" ,http://groups.csail.mit.edu/graphics/classes/6.838/S98/meetings/m12/pred/m12.html

Danke!

...für die Aufmerksamkeit!