TRATAMENTO DE ERROS DE ARREDONDAMENTO NA BIBLIOTECA TERRALIB

Jaudete Daltio[1], Marcus Vinícius Alvim Andrade[1] and Gilberto Ribeiro de Queiroz[2]
[1] UFV - Universidade Federal de Viçosa - Departamento de Informática, Campus da UFV, Viçosa, MG, Brasil, {jdaltio,marcus}@dpi.ufv.br; [2] INPE - Instituto Nacional de Pesquisas Espaciais, Caixa Postal 515, 12201 São José dos Campos, SP, Brasil,gribeiro@dpi.inpe.br

Abstract: In this paper we approach the problems of rounding errors in geometric algorithms. Particularly, we are interested in geometric algorithms used by TerraLib – a spatial component library used to construct small GIS. We describe some techniques that were used to (re)implement and adapt some TerraLib functions to produce robust (roundoff error free) functions.

Key words: Small GIS, TerraLib, Rounding Errors

1. INTRODUÇÃO

Sistemas de Informações Geográficas (SIGs) são sistemas automatizados usados para armazenar, analisar e manipular dados geográficos, isto é, dados que representam objetos e fenômenos em que a localização geográfica é uma característica inerente à informação e indispensável para analisá-la (Câmara et al., 1996). Estes sistemas envolvem problemas de várias áreas, como geometria computacional, computação gráfica, banco de dados, engenharia de software, dentre outras.

Inserido neste contexto encontra-se o projeto TerraLib, que vem sendo desenvolvido por meio de uma parceria entre o INPE (Instituto Nacional de Pesquisas Espaciais), a Tecgraf/PUC-Rio (Grupo de Computação Gráfica da Universidade Católica do Rio de Janeiro) e a FUNCATE (Fundação de

Ciência, Aplicações e Tecnologia Espaciais) (Câmara et al., 2000). A TerraLib consiste em uma biblioteca de classes que possibilita o desenvolvimento de aplicativos geográficos customizados. Ela encontra-se disponível na Internet (TerraLib, 2004) como sistema de código aberto, permitindo o desenvolvimento, de forma colaborativa, de aplicativos e ferramentas geográficas.

O núcleo da TerraLib contém, como qualquer SIG, um módulo geométrico responsável por dar suporte às operações geométricas requisitadas pelas camadas superiores do SIG, como sobreposições e consultas. Neste módulo são definidos a representação geométrica e os diversos algoritmos que realizam as manipulações geométricas. Uma das grandes dificuldades deste módulo está no tratamento dos erros de arredondamento (Yap and Dubé, 1995; Wallner et al., 2000), pois os valores numéricos manipulados por tais algoritmos estão associados a uma representação topológica e as variações causadas pelos erros de arredondamento podem gerar inconsistências entre os valores e as respectivas representações.

As variações nos resultados geradas pelos erros de arredondamento podem ser divididas em duas categorias: variações de valor e variações de decisão.

No primeiro caso, os erros de arredondamento causam "apenas" uma variação no valor numérico real do resultado e esta variação pode ser de certa forma controlada e/ou conhecida. Na verdade, em muitas aplicações estas variações são inerentes ao processo. O segundo caso ocorre quando certas decisões são tomadas baseadas em alguns valores obtidos ao longo do processamento numérico e as variações nos resultados causados pelos erros de arredondamento podem levar a tomadas de decisões erradas. Isto pode levar o programa a resultados imprevisíveis, inclusive podem impedir que o programa termine (isto é, levar a interrupção inesperada de um programa) (Hoffmann, 1989).

O objetivo deste trabalho é incorporar o tratamento de erros de arredondamento na biblioteca TerraLib. A proposta é analisar os módulos da TerraLib para identificar os predicados que são sensíveis às variações de decisão causadas pelos erros de arredondamento e eliminar ou tratar tais erros em cada um destes predicados. A intenção é minimizar a ocorrência de erros de arredondamento que levem de variações de decisão na TerraLib.

2. METODOLOGIA

Dentre as várias metodologias propostas na literatura para o tratamento dos erros de arredondamento (Agrawal, 1995; Yap and Dubé, 1995;

Hoffmann, 2001; Hopcroft and Kahn, 1992; Escardó, 2000), a mais simples consiste em estimar uma tolerância ε e supor que dois valores são iguais se a "diferença entre eles" é menor que ε. É fácil perceber que esta estratégia nem sempre produz o resultado desejado, pois através dela, pode-se considerar que dois objetos diferentes, porém "muito próximos", são iguais. Na verdade, esta estratégia não resolve o problema, apenas o "adia".

É importante ressaltar que, na versão atual da TerraLib, as primitivas geométricas foram construídas utilizando-se tolerâncias ε para tentar contornar os problemas dos erros de arredondamento. A classe *TeGeometryAlgorithmsPrecision* foi projetada com o intuito de definir o valor ε e gerenciar as operações de comparação entre os objetos utilizando este valor.

Visto que a proposta da TerraLib é a produção de uma biblioteca que sirva de suporte para o desenvolvimento de SIGs, é essencial que ela satisfaça alguns requisitos de qualidade, como confiabilidade das funções disponibilizadas. Percebe-se, portanto, a necessidade de se adotar alguma metodologia mais sofisticada para tratar os erros de arredondamento.

Para se contornar por completo estes problemas a solução ideal é a utilização do paradigma de computação exata (Yap and Dubé, 1995), que consiste em representar de maneira exata todos os valores numéricos e determinar, também de maneira exata, todos os predicados geométricos. Este paradigma exige a utilização de uma representação numérica com precisão arbitrária (Hoffmann, 2001; Escardó, 2000), ou pelo menos, com uma precisão maior do que a fornecida pelas instruções de máquina. Isto geralmente leva a um alto consumo de memória e, principalmente, aumenta de forma considerável o tempo de processamento dos algoritmos, pois além do tempo necessário para a alocação de memória, as operações básicas como soma e multiplicação são implementadas em software.

Muitas vezes o tempo de resposta e a quantidade de memória necessária torna inviável o uso deste paradigma ao longo de todo o sistema, fazendo com que o uso da computação exata fique restrito àquelas partes do programa onde os erros de arredondamento podem levar o sistema a situações críticas. Para as demais, poderiam ser utilizadas outras estratégias como a aritmética intervalar (Agrawal and Requicha, 1994; Schmitt, 1975), a aritmética afim (Figueiredo and Stolfi, 1997; Andrade et al., 1994) e a computação simbólica (Zhang, 1996; Roy and Szpirglas, 1990).

3. A ESTRUTURA MODULAR DA TERRALIB

A estrutura modular da TerraLib pode ser representada de forma simplificada conforme Figura 1. O módulo responsável pela representação

dos dados espaciais inclui classes geométricas tradicionais usadas para dados geográficos, como pontos, linhas, polígonos, dentre outros. Esses dados são armazenados em um SGBD, responsável por gerenciá-los. A interface de cada classe é mantida tão mínima quanto possível, de forma que a introdução de novos algoritmos e ferramentas não afete o código já existente.

No projeto da TerraLib foi dada ênfase em manter um fraco acoplamento entre as classes do modelo dos dados, os algoritmos e as estruturas de dados (Câmara et al., 2000; TerraLib, 2004). O conversor de dados permite a exportação dos dados para formatos compatíveis com outros softwares, como o ShapeFile, MIF, JPEG, dentre outros.

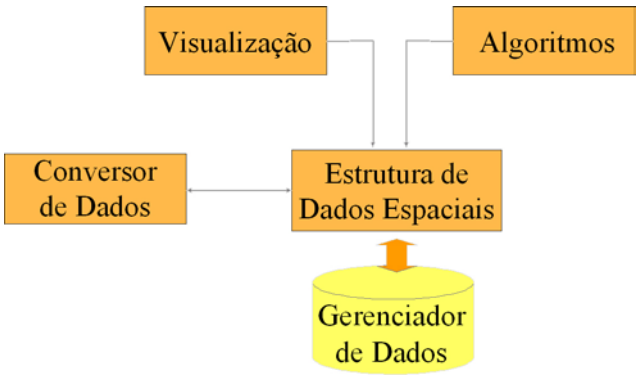


Figure 1. Estrutura Modular da TerraLib

Na implementação dos algoritmos foram combinados os paradigmas de orientação a objetos com a programação genérica (Austern, 1999). A orientação a objetos parte do princípio de agrupar elementos similares em classes e compartilhar as características comuns por meio de herança. A programação genérica parte do princípio de que grande parte dos algoritmos independem de uma implementação particular de uma estrutura de dados, apenas se baseiam em propriedades semânticas fundamentais de cada estrutura. Desta forma, o mesmo algoritmo pode ser adaptado para diferentes geometrias. Para fornecer um ambiente simples para visualização e uma interface que permita a prototipação rápida de idéias foi implementado um conjunto de classes GUI e de visualização, que são baseadas na biblioteca do software do público-domínio QT (QT Library, 2004).

Dentre os módulos que compõem a estrutura da TerraLib, o mais afetado pelos problemas causados pelos erros de arredondamento é aquele responsável pela implementação dos algoritmos. Muitas decisões como testes de incidência, separação, tangência, etc. são deduzidos com base em cálculos numéricos realizados utilizando aritmética em ponto flutuante e tais cálculos, na sua grande maioria, são realizados por funções deste módulo.

Outro módulo que também é afetado pelos erros de arredondamento é o responsável pela representação de dados espaciais, pois é nele que são definidas as representações dos objetos e originalmente estas representações utilizam valores em ponto flutuantes.

Neste sentido, a proposta deste projeto é modificar os módulos de representação de dados espaciais e de algoritmos geométricos, incluindo métodos mais sofisticados para evitar e/ou tratar os erros de arredondamento. Nosso objetivo é analisar as diversas operações geométricas implementadas na TerraLib para determinar a estratégia mais indicada em cada caso para tratar as questões de erros de arredondamento.

4. TRABALHOS REALIZADOS

4.1 Representação de Dados Espaciais

O primeiro passo realizado neste trabalho foi a implementação de um novo módulo para a representação dos dados espaciais utilizando coordenadas homogêneas (Rezende and Stolfi, 1994) para a representação de pontos e retas. Como descrito a seguir, este método de representação possui inúmeras vantagens se comparado com as coordenadas cartesianas.

Em coordenadas homogêneas, um ponto do plano é representado pela tripla $[x, y, w]$, sendo que se $w \neq 0$ então este ponto corresponde ao ponto $(x/w, y/w)$ em coordenadas cartesianas. Se $w = 0$, o ponto corresponde ao ponto no infinito na direção do vetor (x, y).

Analogamente, uma reta é representada pelos seus coeficientes homogêneos $\langle A, B, C \rangle$ e, neste caso, ela consiste de todos os pontos $[x, y, w]$ tais que $A \cdot x + B \cdot y + C \cdot w = 0$. Esta representação permite decidir facilmente de que lado da reta está um determinado ponto. Dizemos então que a reta divide o plano em duas partes: os pontos $[x, y, w]$ tais que $A \cdot x + B \cdot y + C \cdot w > 0$ e os pontos $[x, y, w]$ tais que $A \cdot x + B \cdot y + C \cdot w < 0$.

Este tipo de representação possui diversas vantagens se comparada com a utilização de coordenadas cartesianas. Para os nossos propósitos, as duas principais vantagens são:

- permite uma representação exata (utilizando números inteiros) de pontos com coordenadas racionais que não podem ser representados exatamente com coordenadas cartesianas. Por exemplo, o ponto com coordenadas cartesianas (1/3, 1/3) pode ser representado por [1,1,3]. Isto é muito útil, pois evita diversos erros de arredondamento que causam enormes dificuldades para algoritmos geométricos.

- as operações geométricas básicas como obtenção do ponto de interseção entre dois segmentos, da reta passando por dois pontos, determinação da orientação (horária ou anti-horária) de três pontos, dentre outras, podem ser definidas de maneira simples e são todas baseadas na avaliação de determinantes de uma matriz 3x3, o que torna a implementação destas operações mais simples.

Para incorporar o conceito de coordenadas homogêneas na TerraLib, foi criada uma classe base denominada *TeHomog2D* e desta classe foram derivadas duas outras: *TeHCoord2D* e *TeHCoefficient* para representar os pontos e as retas respectivamente, conforme ilustrado na Figura 2.

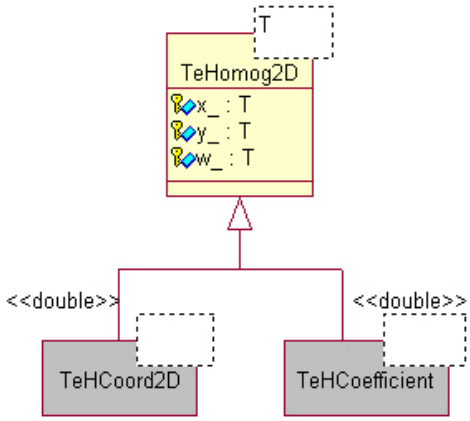


Figure 2. Classes introduzidas na TerraLib

4.2 Algoritmos

Há diversas operações na biblioteca TerraLib (Queiroz, 2003) suscetíveis a erros de arredondamento onde podem ocorrer variações de decisão. Dentre estas podemos destacar os operadores topológicos que testam as relações de dois objetos geométricos (*iguais, disjuntos, tocam-se, cruzam-se*), a análise da orientação de três pontos, determinação da posição relativa entre um ponto e um segmento, verificação se uma linha é ou não fechada, a verificação se um ponto pertence a um determinado polígono ou se ele está em um segmento.

Dentre as operações listadas acima, uma das mais utilizadas pela TerraLib é a operação que fornece a orientação de três pontos, que pode ser realizada computando-se o sinal de um determinante. Esse operador denominado de *TeCCW (Counter Clockwise Test)* retorna o sentido de

rotação em torno dos três pontos: anti-horário, horário ou se os três pontos estiverem alinhados.

Este operador é utilizado em diversas funções, como na verificação para cada segmento de uma linha poligonal se um ponto encontra-se sobre ele *(TeIsOnLine)*, na determinação do envoltório convexo de um conjunto de pontos *(ConvexHull)*, no operador que determina se um polígono é ou não convexo *(TeIsConvex)* e nas funções booleanas que verificam a posição relativa de um ponto em relação a um segmento *(IsAbove, IsBeleow, IsOn)*.

Outra importante aplicação deste operador está na verificação da existência de pontos de interseção entre segmentos de retas *(TeIntersection)*. Dados dois segmentos A e B, formados pelos pontos P_1P_2 e P_3P_4, respectivamente, verificar se eles se interceptam consiste em testar se os pontos P_1 e P_2 estão de lados opostos do segmento formado por P_3P_4 e também se P_3 e P_4 estão de lados opostos do segmento formado por P_1P_2, que nada mais é que avaliar o sinal do determinante de uma matriz 3x3 formada pelas coordenadas homogêneas dos três pontos em questão.

Portanto, é fácil perceber que a obtenção do sinal do determinante de uma matriz é uma operação largamente utilizada na TerraLib. Por isso, esta operação foi a primeira a ser tratada com o objetivo de se evitar erros de arredondamento.

Outra operação muito importante em SIGs que trabalham com representação vetorial é a determinação das interseções entre segmentos. Isso porque operações como união, interseção, diferença e as operações que avaliam o relacionamento topológico são baseadas na determinação destes pontos de interseção.

4.2.1 Obtenção exata do sinal do determinante de uma matriz

A solução adotada se baseia no trabalho de Shewchuk (1997) que explora recursos da computação exata. Embora o uso deste tipo de solução aumente o tempo de execução e a quantidade de memória utilizada, seu uso em algoritmos isolados não representa uma perda significativa de eficiência.

Este algoritmo explora a idéia de que muitas vezes é possível retornar respostas corretas sem terminar a computação exata. No teste da orientação somente o sinal do determinante, e não o seu valor propriamente dito, é necessário. Assim, o algoritmo inicialmente verifica o valor do determinante usando aritmética em ponto flutuante, e somente se a análise do erro indicar que o sinal desse resultado não é confiável então é realizado um teste exato.

A execução do algoritmo computa inicialmente o determinante da forma tradicional, através de operações como multiplicações e adições, e calcula uma seqüência de até quatro resultados (*A, B, C* e *D*), onde *A* é o resultado em ponto flutuante simples, obtido através das operações padrão. Caso esse resultado não seja suficientemente exato, segue-se computando uma

seqüência de aproximações sucessivamente mais exatas (*B* e *C*). Isto é feito através da expansão dos números utilizados em diferentes pontos da equação e realizando-se o cálculo exato das operações sob esses números. O algoritmo termina somente quando a exatidão do sinal é assegurada. Para reduzir o tempo da computação, cada aproximação reusa uma computação precedente menos exata.

Caso essa exatidão não seja obtida no nível *C,* o determinante exato *D* é calculado. Este resultado exato pode possuir até dezesseis componentes, mas o uso da eliminação dos componentes nulos faz com que o comprimento de dois a seis componentes seja mais comum na prática. Na verdade, *A, B,* e *C* são "valores lógicos" usados para testar a exatidão do resultado antes de continuar. Em grande parte das aplicações, a maioria das chamadas ao algoritmo terminará com uma aproximação ponto flutuante *A*, que é computada sem lançar mão de recursos da aritmética exata.

4.2.2 Interseção entre Segmentos

Atualmente está sendo implementada uma versão exata de um algoritmo para obter a interseção entre um conjunto de segmentos. Esta solução é baseada no algoritmo de *Bentley e Ottman* (Preparata and Shamos, 1989; Berg et al., 2000; Rezende and Stolfi, 1994), que consiste em imaginar a existência de uma reta vertical r que varre o plano da esquerda para a direita, identificando e tratando alguns eventos. Cada um destes eventos serve de gatilho para que um conjunto de operações seja realizado. São definidos três tipos de eventos: extremo esquerdo (vértice mais à esquerda do segmento), extremo direito (vértice mais à direita do segmento) e cruzamento (ponto de interseção entre dois segmentos).

Uma agenda de eventos *E* é utilizada para armazenar os eventos, que são mantidos ordenados crescentemente pelo valor da abscissa. O algoritmo utiliza também uma lista *L* para indicar os segmentos ativos a cada instante, que devem ser tratados a cada evento. Um segmento é dito ativo quando ele é interceptado pela reta de varredura.

Na prática, a varredura consiste em percorrer a agenda de eventos e para cada elemento desta lista executar seu conjunto de operações. Na determinação do ponto de interseção entre dois segmentos a TerraLib utiliza atualmente a representação paramétrica dos segmentos, implementando a resolução de um sistema de equações. Com o uso de coordenadas homogêneas este algoritmo de determinação do ponto de interseção pode ser implementado de modo mais simples e mais robusto, permitindo que os casos degenerados sejam tratados mais facilmente como, por exemplo, quando dois segmentos são paralelos. Os pontos de interseção obtidos e os dois segmentos que os geraram são inseridos numa lista que, ao final do processo, conterá todas as interseções existentes entre os segmentos. A

implementação em desenvolvimento visa tratar também casos especiais, como segmentos verticais e colineares, que apresentam problemas na implementação convencional do algoritmo de varredura.

O algoritmo supõe que os segmentos de entrada são definidos por pontos (representando seus vértices) cujas coordenadas são números reais com 3 casas decimais. Inicialmente, faz-se a conversão destes números para coordenadas inteiras, multiplicando-os por 1000 para eliminar a parte decimal. Isto faz com que estes valores se tornem relativamente grandes, sendo muito comum que, durante o processamento (após diversas multiplicações e somas) possam ocorrer situações de *overflow* (quando uma operação produz um valor numérico maior do que o maior valor representável). Para evitar a ocorrência de *overflows* e *underflows* adotamos algumas estratégias, descritas abaixo.

As operações básicas foram sobrecarregadas de forma a identificar a possível ocorrência destas situações e disparar exceções, não permitindo que elas ocorram. No tratamento destas exceções, inicialmente faz-se a translação dos pontos para o menor valor de *X* dentre as abscissas e o menor valor de *Y* dentre as ordenadas dos vértices, o que reduz consideravelmente os valores das coordenadas dos pontos. Outra estratégia adotada é a utilização do método da bisseção, que consiste em reduzir os dois segmentos analisados dividindo-os à metade e descartando o trecho de cada segmento que não contém o ponto interseção, até que o tamanho dos segmentos seja pequeno o suficiente para não gerar um *overflow*.

5. PRÓXIMOS PASSOS

O andamento do projeto será focado na implementação de funções que podem ser resolvidas através da aplicação dos algoritmos já trabalhados (orientação e interseção), que são utilizados como predicados para tomadas de decisão dentro de outras funções.

Com base nesta análise, os próximos passos incluem uma implementação robusta para os algoritmos de determinação de envoltório convexo e da orientação de polígonos. Além disso, pretendemos (re) implementar de maneira exata os relacionamentos topológicos: *touch, in, cross, overlap, disjoint*, comumente aplicados em outras funções. A construção destes operadores é baseada na análise de uma matriz de 9 interseções, que analisa para cada dois objetos (ponto, linha e polígono) as interseções existentes entre interior, fronteira e exterior. Desta forma, grande parte do seu esforço computacional está relacionado ao problema da interseção.

Visamos também, posteriormente, implementar os operadores união, interseção e diferença entre polígonos, amplamente utilizados para a

construção de consultas em mapas e operações de sobreposição. Nestes algoritmos, o cálculo das interseções entre os objetos é de extrema importância, pois os pontos de interseção devem ser determinados numa das suas primeiras etapas de seu processamento.

Posteriormente, prevê-se o estabelecimento de uma análise comparativa do desempenho alcançado pela implementação robusta desenvolvida em relação a implementação atual da TerraLib, analisando-se, para diversos casos, as operações reimplementadas.

AGRADECIMENTOS

Este trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, entidade governamental brasileira promotora do desenvolvimento científico e tecnológico através do projeto TerraUFV número 552435/2002-3.

6. REFERÊNCIAS BIBLIOGRÁFICAS

Agrawal, A., 1995, "A General Approach To The Design Of Robust Algorithms For Geometric Modeling", *PhD Thesis*, Comp. Science, Southern California Univ.

Agrawal, A., and Requicha, A. A. G., 1994, "A Paradigm For The Robust Design Of Algorithms For Geometric Modeling", *Computer Graphics Forum*.

Andrade, M. V. A., Comba, J. L. D., and Stolfi, J., 1994, "Affine Arithmetic" (Extended Abstract). *Presented At Interval'94*, St. Petersburg (Russia).

Austern, M. H., 1999, "Generic Programming and STL", *Using and Extending the C++ Standard Template Library*.

Berg, M., Kreveld, M. Van, Overmars, M., and Schwarzkopf, O., 2000, "Computational Geometry, Algorithms And Applications", *Springer Verlag*.

Câmara, G. et al, 2000, "Terralib: Technology In Support of GIS Innovation". *II Workshop Brasileiro De Geoinformática*, Geoinfo2000.

Câmara, G., Casanova, M., Hemerly, A., Magalhães, G., and Medeiros, C., 1996, "Anatomia de Sistemas de Informação Geográfica", *X Escola Brasileira da Computação*.

Escardó, M. H., 2000, "Exact Numerical Computation", *Issac'00*, Acm Press. Tutorial.

Figueiredo, L. H., and Stolfi, J., 1997, "Self-Validated Numerical Methods And Applications", *Brazilian Mathematics Colloquium Monograph*, Impa, Rio de Janeiro.

Hoffmann, C. M., 1989, "The problems of accuracy and robustness in geometric computation", *IEEE Computer*.

Hoffmann, C. M., 2001, "Robustness In Geometric Computations", *Technical Report*, Computer Science Dept., Purdue University.

Hopcroft, J. E., and Kahn, P. J., 1992, "A Paradigm For Robust Geometric Algorithms", *Algorithmica*.

Preparata, F., and Shamos, M., 1989, "Computational Geometry: An Introduction", *Springer-Verlag*.

QT Library home page, 2004, http://www.trolltech.com/products/qt/qtlibrary.html.

Queiroz, G. R., 2003, "Algoritmos Geométricos para Bancos de Dados Geográficos: Da Teoria à Prática na Terralib", *Tese de Mestrado*, Inpe.

Rezende, P. J., and Stolfi, J., 1994, "Fundamentos de Geometria Computacional", *IX Escola de Computação*.

Roy, M. F., and Szpirglas, A., 1990, "Complexity of Computation of Real Algebraic Numbers". *Journal of Symbolic Computation 10*.

Schmitt, G., 1975, "Some Considerations Using Interval Analysis In Adjustment Computations XVI General Assembly of The Internat", *Union of Geodesy and Geophysics, Internat*.

Shewchuk, J. R., 1997, "Adaptive precision floating-point arithmetic and fast robust geometric predicates", *Discrete Comput. Geom.*

TerraLib home page, 2004, http://www.terralib.org.

Wallner, J., Krasauskas, R., and Pottmann, H., 2000, "Error Propagation In Geometric Computations".

Yap, C. K., and Dubé, T., 1995, "The Exact Computation Paradigm", Du and F.K. Hwang, Editors, *Computing In Euclidean Geometry*, World Scientic Press, Singapore.

Zhang, J., 1996, "Symbolic Computation On Complex Polynomial Solution of Differential Equations", *Journal of Symbolic Computation*.