Proceedings of the International Symposium on Spatial Data Handling, Zurich, Switzerland, 20-24 August 1984

27

CARTOGRAPHIC ERRORS SYMPTOMATIC OF UNDERLYING ALGEBRA PROBLEMS

Dr. Wm. Randolph Franklin

Electrical, Computer, and Systems Engineering Dept.
Rensselaer Polytechnic Institute
Troy, New York, USA, 12181

518-266-6077
Telex: 64 65 42, Answerback: RPITRO

Abstract

One problem that makes map overlay and other computer cartographic operations such as projection, rotation, and scaling so difficult is that of numerical errors. As database sizes grow, these initially trivial inaccuracies can cause topological inconsistencies that affect the overall integrity of the database. The usual solution is to represent the coordinates more accurately, as with a finer grid. However, this can never lead to a complete solution, and is actually part of the problem. This paper discusses the relationship of geometry to topology, and then examines a sequence of alternative models of computation, that allow ever more complex operations to be performed exactly, at the cost of ever greater complexity. The various models include integers, floating point "reals", big integers, finite precision rational numbers, exact rationals, rationals with either a square root or a general root operator appended, algebraic numbers, interval arithmetic, and lazy evaluation.

Introduction

A chief task of computer science and engineering is to design and implement systems that perform "naturally" according to the common-sense intuition of a user who, though ignorant of computers, is competent in his own speciality. This problem is now evident in computer cartography as larger databases, with tens of thousands of points, are processed by more complex algorithms implemented by thousands of lines of code, such as map overlaying. The major problem is that internal topological inconsistenies occur as a result of computational numerical inaccuracies. The errors can take the form of boundaries that intersect when they shouldn't, boundaries that do not meet at a vertex when they should, points that test to be inside a region when they should be outside, and so on. This problem is worse than just a "fuzz-factor" since, although users can accept small measurement inaccuracies, a database that is internally inconsistent represents an internal machine state that does

not correspond to any external state. This violation of the system's specifications constitutes a failure (Randell et al, 1978).

This problem of approximate calculations being counter intuitive also occurs in other areas such as Computer Aided Design, where it is even more severe since the geometry is three dimensional.

The usual solution is to increase the resolution of the coordinates by using more significant digits. This is equivalent to using a finer grid. Since the problems arise with close objects in the database, higher resolution would appear to reduce the problem, although it does not solve it completely. It appears possible that if only we had a sufficiently fine grid, we could solve the problem completely.

A theme of this paper is that the grid structure is not the solution, but is part of the problem. The visible problem is symptomatic of a deeper underlying problem in the number system that is used for the arithmetic calculations, and any solution must consider these fundamental issues.

Two more themes are presented here. First, it is impossible to separate the geometry from the topology since arbitrarily small geometry errors can later cause topology errors. Second, this problem can be mitigated either restricting the set of allowable operations or expanding the algebra in which the calculations are done, but neither approach is satisfactory, and the problem is still one of the more important unsolved problems in computer graphics today.

This present work builds upon, and is motivated by, a diverse body of previous research. Some other aspects of the necessity for high quality databases are described in (Chrisman, 1983). Some map overlay algorithms are presented by (D. White, 1977) and (Franklin, 1983b). Useful cartographic data structures are given by (Peucker and Chrisman, 1975) and (Franklin, 1983c), and the effect on special cases of changing the data structures is described in (Franklin, 1983a). The necessity for considering the underlying topological concepts is discussed by (M. White, 1983). Finally, an eventual direction for spatial data structure systems is given by (Peuquet, 1983).

In the rest of this paper, we will first study the problems of using integers in more detail, then consider the relationship of geometry to topology in an approximate spatial data structure. Next we will see that floating point numbers are even worse. After this we will see several other possible algebras and computation models that can solve the problem in varying degrees, at a proportionate implementation cost.

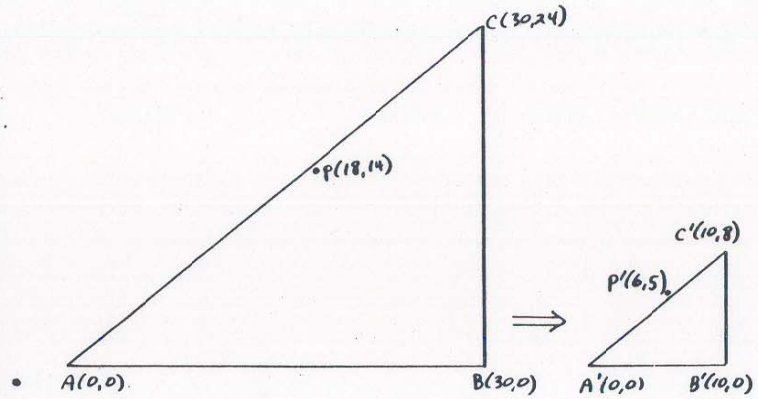

Figure 1: Scaling P Moves It Outside Triangle ABC

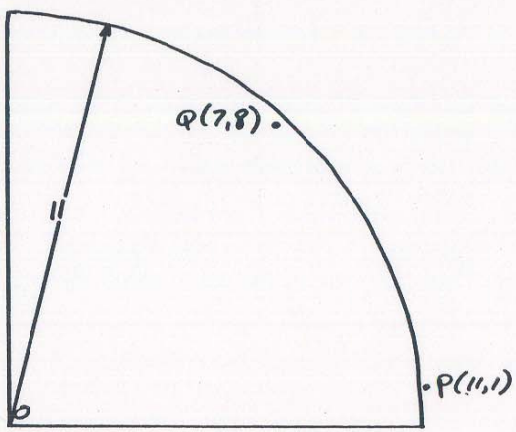

Figure 2: Rotating P Moves It Inside the Circle

Problems With Using Integers

Many spatial databases represent point coordinates with integers. This is equivalent to forcing all the coordinates to lie on the vertices of a grid when they are digitized. If there are problems with a loss of accuracy, for reasons to be described below, then the usual solution is to make the grid finer and more accurate. This appears to solve the problem (always except for a few nasty special cases that never seem to go away), though it actually only reduces it somewhat. It is a theme of this paper that the act of seeing a finer grid as a solution is itself part of the problem since it presupposes the form of the solution. Finally, as ever larger databases are being processed mechanically without any human intervention to enforce "common sense" constraints, then it is not sufficient to remove most of the computational problems; we must devise a system that handles them all.

Although the problems with using integers to represent map coordinates are obvious, it is worthwhile to recapitulate them. We will see where scaling or rotating an object can cause a point to move to the other side of a line. This is serious since point inclusion in a region is a fundamental operation in cartography. This leads into the general relationship of geometry and topology that is discussed next. First, consider the triangle ABC with A(0,0), B(30,0), C(30,24) as shown in Figure 1. Point P(18,14) is inside the triangle. Now, let us scale the triangle down by a factor of 3, and approximate any fractional coordinates by the closest integer. Triangle ABC becomes triangle A'B'C' with A'(0,0), B'(10,0), C'(10,8). However, P is transformed to P'(6,5), which is outside A'B'C'. Thus the scaling has moved a point from inside an object to outside it.

Rotating an object can also cause the same effect. Consider a circle of radius 11 centered on the origin, as shown in Figure 2. Let us rotate point P(11,1), which is outside the circle, in the positive direction by 45° to make point Q, and round off the result to the nearest integer. Q's location is (7.07, 8.48) which rounds to (7,8), which inside the circle.

There are also the problems that occur when the intersection of two lines is not an integer, so that the closest integral point is not on either of the two lines. Finally, as we use larger and larger integers, which is equivalent to using a finer grid for the data, overflowing the maximum legal integer becomes a problem.

Although the above errors are actually not too likely to occur anytime that a particular point is transformed, as ever larger data structures are processed, the probability of these errors occuring is large enough that the integrity of the whole database can be threatened.

Geometry Affects Topology

First we must distinguish between geometry and topology in this context:

Definitions

The TOPOLOGY a scene is the set of coincidence and inclusion relations in the scene. This includes whether a point lies on a line, and if not, which side of the line it is on, whether two lines intersect, and so on.

The GEOMETRY of a scene is the set of actual measurements of distances and coordinates. This includes the distance between two points, from a point to a line, and so on.

Although an actual cartographic database contains an assortment of heterogeneous data, we will consider only the points and lines here. The root of the problem is that we do not have a static scene, but one that is continuously being extended and modified. Thus a latent geometry error in a scene with the correct topology might cause a topology error after some future operation. For example, if we take an object, A, and rotate it 360 times by one degree, and then subtract the result from the original A, the result should be the empty set. This will probably not in fact occur. This is formalized in the following lemma.

Lemma

Assume that you are given four points in the plane, no three of which are collinear. You are allowed to extend an infinite line through any two points, and to create a new point at the intersection of any two lines.

Under these conditions, an arbitrary small change in the geometry of the input points can change the topology of the derived scene. That is, consider two sets of inputs, where one differs from the other by an arbitrarily small change in one point's coordinates. Then there is a sequence of constructions of new lines and points where some point is on one side of some line in the first case, but on the other side in the second case.

The proof of this lemma is obvious.

One possible solution is to require that each coordinate calculation be performed in the same manner each time, that is, that exactly the same sequence of arithmetic calculations be used. Then at least the errors will be repeatable and the geometry and topology will together define some scene, although it may not be exactly the one that user has in mind. The problem with this is that the user will mentally apply certain geometry theorems that affect the topology. For example, see Figure 3 which shows the three medians of triangle ABC. Assume that the user gives the cartographic system the following sequence of commands:

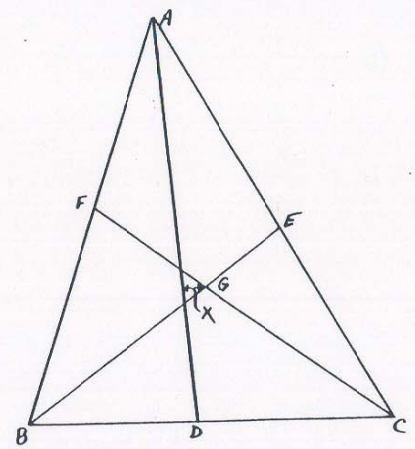

Figure 3: Calculating the Median, G, of Triangle ABC Using Floating Point Numbers

1. Input coordinates of A.
2. Input coordinates of B.
3. Input coordinates of C.
4. Let D = midpoint of BC.
5. Let E = midpoint of AC.
6. Let F = midpoint of AB.
7. Calculate equation of line through A and D.
8. Calculate equation of line through B and E.
9. Let G = intersection of AD and BE.
10. Calculate equation of line through C and F.
11. Calculate X = distance from G to CF.

Unless the cartographic system is a geometry theorem prover, it will not know know that X=0 exactly, and will calculate some small non-zero number. That it may calculate the same erroneous number each time is irrelevant since the topology is wrong and the math-literate but computer-naive user will expect X to be exactly 0.

Although this error may be small, a user who is unaware might then magnify it by some operation such as dropping a perpendicular from G to CF and then extending that line until it intersects something else at some random location.

For example, using the APL language on the IBM 3033 under the Michigan Terminal System at RPI, which does double precision calculations, if A=(0,0), B=(1,0), and C=(0,1), then the distance from G to CF is 1.388E-17.

Here is another example, in 3-D. Consider a cube with vertex coordinates at (±1, ±1, ±1). Most computers can calculate the equations of the faces so that:

1. Each face's four vertices lie exactly on the face.

2. Opposite faces are exactly parallel.

3. The angles at the face vertices are exactly 90°.

Thus the topology and geometry are correct. Now rotate this cube by some angle such as 20° around the X-axis, followed by 30° around the Y-axis and concluded by 40° around the Z-axis. The new vertex coordinates will not satisfy any of the three conditions for a cube stated above. We may try to relax the problem and be satisfied with a cube that is "close" to the rotation angles given, but it may be impossible to satisfy even the topology condition (1) above. Again limitations in geometry are causing topology errors.

Computer Floating Point Numbers

More cartographic database problems arise from computer floating point numbers. In fact, it is to avoid these complexities that integers are sometimes used. Floating point numbers are a model of the real number field which is defined by certain axioms, (Spiegel, 1963). In fact, for almost every real number axiom, there is a computer implementation for which it is violated:

Distributivity A*(B+C) = A*B + A*C

For a violation of this rule, consider a decimal floating number system which stores one decimal digit for each number. Calculations are performed exactly and then rounded to one digit. Thus 9 and 10 are exactly representable in this system, but 11 is not. Because of rounding, 2*4=10 but 2*8=20. Let A=2, B=6, and C=2. In this number system, distributivity is not satisfied, since A*(B+C)=20, but A*B+A*C = 10.

Associativity (A+B) + C = A + (B+C)

This is violated one any machine with fewer than 30 significant digits if A=1.E30, B=-1.E30, C=1.

Commutativity A+B = B+A

This can be violated on systems which can hold temporary numbers in registers at a higher precision than they can be stored in memory, and further allow a register - memory operation if the register is the first operand. When the second operand is stored in memory, it will lose precision. Four different machines with overlength floating point registers that violate commutativity are given in (Malcolm, 1972), (Gentleman and Marovich, 1974).

Reciprocal There is a Y such that X*Y = 1.

For example, in the APL system mentioned above, there is no multiplicative inverse for 3 since 1-3x(1/3) = 1.388E-17.

These problems can be reduced by using a properly designed floating point standard, but cannot be entirely removed. The IEEE standard, designed by numerical analysts, is the best. In contrast, those systems dating from the 1960s can be quite poor.

Varying The Underlying Algebra

To avoid the above mentioned complexities of existing computer numbers, some other algebra may be used for the computations. The chosen algebra must be judged according to the criteria of simplicity and generality, i.e. what operations are possible and what figures may be represented. Several algebras will now be compared.

Big Integers

A big integer is an integer represented with as many digits as are necessary, perhaps 100 or more. Since the integers do satisfy the ring axioms, (Lipschutz, 1968), exactly, restricting coordinates to them means that the order of the calculation is not important since that distributivity, commutativity, and associativity are satisfied. There is also no problem with overflow, so that we can use as fine a grid as we wish.

However, big integers suffer the same problems with scaling and rotation as the usual integers. In addition, since most computer languages, with the exception of a few systems such as MACSYMA, (MIT, 1974), do not contain them, we must implement them ourselves. Following is a sample of what is necessary to add two big integers:

```
C>> Intadd - Add 2 big integers A and B with N elements each
C   to get C.
      SUBROUTINE INTADD(A, B, C, N)
      INTEGER A(N), B(N), C(N), N, I, CARRY
      I=N
      CARRY=0
1     C(I)=A(I)+B(I)+CARRY
      IF (C(I).GT.10000) THEN
         C(I)=C(I)-10000
         CARRY=1
      ELSE
         CARRY=0
      ENDIF
      I=I-1
      IF (I.GT.0) GO TO 1
      RETURN
      END
```

More details of big integers are described in the section on exact rationals, below.

Rationals Of Finite Precision

In this system, a coordinate is a quotient of a numerator and a denominator, both of which are integers of some maximum precision (number of digits). The big problem with rational numbers is that when two are combined by some arithmetic operation, the result will probably have as many digits as in the two operands combined. For example, 1/2+3/5 = 11/10. Thus approximations will soon be necessary, which causes all the problems that floating point numbers were subject to. Otherwise, the advantages and disadvantages are the same as for exact rationals below.

Since usual programming languages do not include rationals as a data type, they must be implemented. How to do this in Fortran is shown below. Here, one rational number is implemented as an array of two integers, with the first element storing the numerator, and the second the denominator. We assume that the numerator and denominator have no common factors greater than one, i.e. that the fraction is reduced. Another problem, is that none of the arithmetic operations, such as +, -, *, or /, nor any of the built-in functions are defined for rationals, so that the user must implement them also. The following shows an example, where two rationals, A=1/2, and B=2/3 are defined, and then added. GCD is a predefined function that calculates the greatest common divisor.

```
C>> Main program to test Ratadd.
      INTEGER A(2), B(2), C(2)
      DATA A/1,2/, B/2,3/
      CALL RATADD(A, B, C)
      PRINT 1, A, B, C
1     FORMAT(I5,'/'I5, ' + ', I5,'/',I5, ' = ', I5,'/',I5)
      STOP
      END
```



```
C>> RATADD - Add two rational numbers A and B to give C.
      SUBROUTINE RATADD (A, B, C)
      INTEGER A(2), B(2), C(2), I, GCD
      EXTERNAL GCD
C     Warning: the next two statements may overflow.
      C(1)=A(1)*B(1)+A(2)*B(1)
      C(2)=A(2)*B(2)
C     Reduce C to its lowest terms by removing common
C     factors. I assume that GCD is already defined and is
C     never 0.
      I=GCD(C(1), C(2))
      C(1)=C(1)/I
      C(2)=C(2)/I
      RETURN
      END
```

The process of defining all the low level operations that we are accustomed to take for granted can be very tedious. Writing programs that use our defined operations, can also be tedious, since brief expressions such as

```
D=B**2-4*A*C
```

become long strings of subroutine calls:

```
INTEGER R4(2), T1(2), T2(2)
DATA R4/4,1/
CALL RATMUL(A, C, T1)
CALL RATMUL(T1, R4, T2)
CALL RATMUL(B, B, T1)
CALL RATSUB(T1, T2, D)
```

The AUGMENT system, (Crary, 1974), (Crary, 1975), automates this last step. Once the user has defined his new data types and the names of the subroutines to manipulate them, AUGMENT can convert an infix expression containing these new types, such as above, into a Fortran program containing the proper sequence of subroutine calls. Of course, the user must still implement these subroutines that are being called.

Finally, a useful rational addition routine would need to scale the numbers to prevent overflow, and this would introduce the very errors that we are trying to avoid. Such a scaling rational addition routine could be:

```
C>> RATADD - Add two rational numbers without overflowing.
      SUBROUTINE RATADD (A, B, C)
      INTEGER A(2), B(2), C(2), I, GCD, LIMIT, M, SCALE
      EXTERNAL GCD
      DATA LIMIT/32767/
C     Scale numbers if necessary to prevent overflow.
C     This introduces errors, and will change a large number
C     to infinity.
      M = MAX( ABS(A(1)), ABS(A(2)), ABS(B(1)), ABS(B(2))
      IF (M.GT.LIMIT) THEN
         SCALE=LIMIT/M
         A(1)=A(1)/SCALE
```

```
    A(2)=A(2)/SCALE
    B(1)=B(1)/SCALE
    B(2)=B(2)/SCALE
  ENDIF
  C(1)=A(1)*B(1)+A(2)*B(1)
  C(2)=A(2)*B(2)
  Reduce C to its lowest terms by removing common
  factors. I assume that GCD is already defined.
  I=GCD(C(1), C(2))
  C(1)=C(1)/I
  C(2)=C(2)/I
  RETURN
  END
```

xact Rationals

By this we mean representing a number as a numerator ivided by a denominator, where both are represented by big ntegers with as many digits as necessary. For example, we ight have

$$A = \frac{123456789123456789}{8346351293274672736}$$

Since we cannot represent such large integers directly n most languages, we must implement them also. The usual echnique is to use an array to store each integer, with bout 4 digits stored in each element. Then the numerator of would be an array AN(5) with

```
AN(1)=12
AN(2)=3456
AN(3)=7891
AN(4)=2345
AN(5)=6789
```

The denominator of A would be another array, AD. In ummary, representing a point with exact rationals would equire using 2 exact rational numbers. Each exact rational n turn would be the quotient of 2 big integers. Each big nteger would be an array of normal sized integers.

This number field has many advantages:

a) Since it is a true mathematical field, distributivity nd the other properties hold, so that the order of alculation is immaterial and mathematical identities hold.

b) Objects can be scaled up and down by any rational raction.

c) If two lines are intersected, then the new point lies n them. Likewise, if a plane equation is calculated from hree points, then they will fall on the plane.

d) Objects can be rotated by any angle whose trig unctions (e.g. sine and cosine) are rational.

Against this are a few deficiencies:

a) This representation is not simple since the number of digits in a number doubles with each operation.

b) Certain figures which are constructible in classical geometry (i.e. with ruler and compass) are not realizable in rational numbers. For example, as shown in Figure 4, the length of the diagonal of a square cannot be used as a coordinate. Even the pentagram shown in Figure 5 cannot be represented. We do not require that it be regular, which would certainly require a Sqrt(5), but merely that some coordinates be assigned to the vertices so that the given collinearity and coincidence relationships obtain. This is impossible unless the whole figure collapses to a single line.

The proof is that the collinearities in the pentagram are projective geometry properties, and the figure is constrained such that the projective cross-ratio of ABCD is constant. Since it is irrational if the pentagram is regular, then it is always irrational. This means that at least one of A, B, C, or D has an irrational coordinate.

c) Objects cannot be rotated by angles such as 30° or 45°.

The question of rotation is not totally trivial. Define a "trig-rational" angle as one whose sine and cosine are rational, i.e. the angles that are usable with rationals. Although no angle, not a multiple of 90°, that is itself rational or even algebraic, will be trig-rational, the trig-rational angles are dense in the interval [0,360), so that we may find a representable angle arbitrarily close to the desired one:

Lemma

Trig-rational angles are dense in [0,360). (This means that there is a trig-rational angle arbitrarily close to any angle T such that 0<=T<360, (Spiegel, 1963).)

Proof 1

Choose one such angle, such as T=Arcsine(0.6). Since T is irrational, there do not exist M and N, M≠N, such that M*T=N*T. As N->infinity and the angles (N*T Mod 360) are added to [0,360), the largest interval between consecutive angles tends to zero, (Knuth, 1973). Thus there is a trig-rational angle arbitrarily close to any angle.

Proof 2

The angle T such that

$$\tan T = \frac{2K+1}{2K(K+1)}$$

for integer K is trig-rational. As K -> infinity, T->0. Thus some multiple of T is arbitrarily close to our target angle.

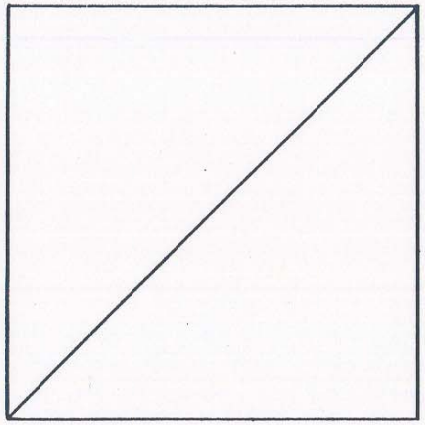

Figure 4: The Diagonal of a Square Cannot Be Used With Rational Coordinates

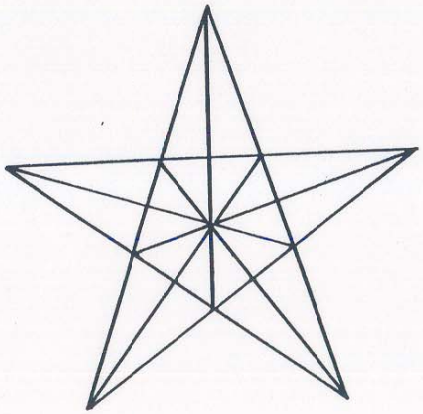

Figure 5: A Pentagram is Not Representable In Rational Coordinates

Example

We wish to calculate a trig-rational angle close to 45°.

Begin with the traditional method of generating a triple of integers (A,B,C) that satisfy Pythagoras' theorem, that is, pick (X,Y) integers, and let:

```
A = X**2 - Y**2
B = 2 X*Y
C = X**2 + Y**2
```

Since tan 45° = 1, then we want A~B (A approximately equal to B), i.e.

```
X**2-2XY-Y**2 ~ 0
```

Therefore

$X \sim Y(1+\sqrt{2})$

Since X and Y must be integers, the closer the approximation is, the closer T will be to 45°. For instance, if X=12, Y=5, then tan(T)=110/120 and T=44.76^{o}. If X=1207 and Y=500, then tan(T)= 1206849/1207000 so that T=44.996^{o}.

The weakness with this is that $8*T \neq 360^{o}$ so that if the user rotates his object eight times successively by our approximation to 45°, then the resulting object will not be exactly the same as the original. Thus this algebra fails the test of producing intuitively correct results.

Thus we must expand the field to allow the common angles.

Rationals Plus Square Root

This field allows all expressions E produced by this grammar:

```
E = Q |
E = Sqrt(E) |
E = E op E
Q = any rational number
op = "+" | "-" | "*" | "/"
```

This expanded system has some powerful advantages:

a) We can now represent any coordinate that can be created by a traditional ruler and compass construction. Since these are the operations that are intuitively reasonable, we can exactly represent all intuitively reasonable constructions.

b) We can represent 30° and 45° and all angles

obtainable by bisection or addition. These are the angles 3*K/(2**L) degrees for positive integers K, L.

On the other hand:

a) Combining two numbers can give an expression whose simplest representation is as long as the sum of the lengths of the two operands. That is, as for rational numbers, there is frequently no simplification. These expressions may contain nested square roots. Thus we need an algebraic manipulation system, such as MACSYMA (MIT, 1974) or REDUCE (Hearn, 1973) to implement our cartographic system.

For example 3°, the smallest integral angle expressible here has as its simplest representation:

$$\cos 3^o = \frac{(\sqrt{6}-\sqrt{2})(\sqrt{5}-1) + (\sqrt{6}+2)(\sqrt{10+2\sqrt{5}})}{16}$$

b) It is difficult to determine the simplest properties of these numbers, such as whether they are zero, positive, or negative. For example, if

$$X = \sqrt{3+2\sqrt{2}} - \sqrt{3-2\sqrt{2}} - 2$$

then is X=0 ? If we evaluate the expression to sufficient precision using approximations, then we may suspect that X=0. To prove it, we must find the polynomial with rational coefficients that has X as one of its roots, find an interval [A,B] that separates the root from all the extraneous ones, and test whether the polynomial has a zero root in [A,B]. In this case, since there the expression has four square roots, the polynomial would probably be of degree 16.

c) We still cannot represent angles such as 1° that are formed by trisection. In order to obtain them, we must generalize further:

Rationals Plus A General Root Operation

Compared with the previous case, our expressions are even more complicated, but we can represent more of the angles, such as those obtained by trisection. For example,

$$\cos 3^o = 4(\cos 1^o)^3 - 3\cos 1^o$$

which can be solved to give a complicated, though explicit, formula for $\cos 1^o$ in terms of $\cos 3^o$, which is itself complicated.

Algebraic Numbers

These are the roots of polynomials with rational coefficients. The advantage is that we can represent any rational angle, since if angle T=N/D degrees where N and D are integers, then performing a D-section of the N degrees is equivalent to solving a D-th degree polynomial.

The problem is that, due to the unsolvability of general polynomial equations, the only means of representation of a number is as the following pair:

- a polynomial, plus
- an interval [A,B] that isolates one of the roots.

This field is clearly too complicated for everyday use.

Other Possible Computation Models

Some possible solutions do not involve varying the underlying algebra, and can in fact be superimposed on top of any of the models described in the last section . They include:

Interval Arithmetic

This accepts that the calculation will be only approximate, and stores the possible range for each number. Thus instead of 2, we might have [1.9,2.1]. Then, instead of

(3-2)*4 = 4

we would have

([2.9,3.1] - [1.9,2.1]) * [3.9,4.1] = [3.12, 4.1]

Notice how the interval can grow. If we can determine when the interval will be large enough that the topology becomes uncertain, then we know that we will be safe until then.

The implementation of interval arithmetic is almost as complicated as rational numbers, since the user must first define each number as an array of the lower and upper bounds, and then write all the low level routines. For example, the addition routine could be:

```
C       Intadd: Interval arithmetic addition routine.
        SUBROUTINE INTADD (A, B, C)
        REAL A(2), B(2), C(2)
C       Combine the lower bounds.
        C(1)=A(1)+B(1)
C       Combine the upper bounds.
        C(2)=A(2)+B(2)
        RETURN
        END
```

Interval arithmetic packages has been studied for some time. One implementation, which uses AUGMENT to compile into a Fortran program, is described in (Yohe, 1977).

Lazy Evaluation

Instead of calculating any numbers, we might just store the expression that, when evaluated, would give the number. Thus if the program contained these lines:

```
A=1.
B=2.
C=-3.
D=B**2-4*A*C
```

instead of storing 16 for D, the computer would store 4**2-4*1*(-3). It would defer evaluation until an answer was needed, and then try to simplify the expression before evaluating it. Implementing lazy evaluation in Fortran would be difficult; it would be better to build upon Lisp or a similar system. Since we need to evaluate expressions only to determine topology, we can defer some work until then. Nevertheless, the choice of underlying algebra still affects us at that point. In the end, lazy evaluation has no advantages.

Summary

This paper presents a challenge in the question of the best method for performing calculations with coordinates when processing spatial data. Existing computer numbers violate all the axioms of the number systems that they are designed to model, with the resulting geometric inaccuracies causing topological errors wherein the database violates its design specifications. These problems can be corrected by any one of several other models of computation, with the more complicated models allowing more powerful operations to be performed exactly. However, in these models, a coordinate becomes an expression of arbitrary size, whose complexity can double with every operation since it generally doesn't simplify. In addition, using such a system often requires first implementing it from the ground up, beginning with subroutines for addition and the elementary functions. More research is needed to determine a system of computation that strikes a useful medium between consistency and simplicity.

Acknowledgement

This material is based upon work supported by the National Science Foundation under grant no. ECS 80-21504.

References

(Chrisman, 1983) N.R. Chrisman. "The Role of Quality Information in the Long-Term Functioning of a Geographic Information System", Proc. Sixth International Symposium on Automated Cartography (Auto-Carto Six), vol. 1, (16-21 October 1983), Ottawa, Canada, pp. 303-312.

(Crary, 1974) F.D. Crary. The Augment Precompiler, I. User Information, MRC Technical Summary Report #1469, Mathematics Research Center, University of Madison-Wisconsin, (December 1974, revised April 1976).

(Crary, 1975) F.D. Crary. The Augment Precompiler, II. Technical Documentation MRC Technical Summary Report #1470, Mathematics Research Center, University of Madison-Wisconsin, (October 1975).

(Franklin, 1983a) W.R. Franklin. "Rays - New Representation for Polygons and Polyhedra", Computer Graphics and Image Processing 22, (1983), pp. 327-338.

(Franklin, 1983b) W.R. Franklin. "A Simplified Map Overlay Algorithm", Harvard Computer Graphics Conference, Cambridge, MA, (31 July - 4 August 1983), to appear.

(Franklin, 1983c) W.R. Franklin. "Adaptive Grids For Geometric Operations", Proc. Sixth International Symposium on Automated Cartography (Auto-Carto Six), vol. 2, (16-21 October 1983), Ottawa, Canada, pp. 230-239.

(Gentleman and Marovich, 1974) W.M. Gentleman and S.B. Marovich. "More on ALgorithms That Reveal Properties of Floating Point Arithmetic Units", Comm. ACM 17, (5), (May 1974), pp. 276-277.

(Hearn, 1973) A.C. Hearn. Reduce 2 User's Manual, University of Utah, Computer Science Department, (1973).

(Knuth, 1973) D.E. Knuth. The Art of Computer Programming, Volume 1: Fundamental Algorithms, (1973), Addison-Wesley.

(Lipschutz, 1968) S. Lipschutz. Theory and Problems of Linear ALgebra, Schaum's Outline Series, McGraw-Hill, New York, (1968).

(Malcolm, 1972) M.A. Malcolm. "Algorithms to Reveal Properties of FLoating Point Arithmetic", Comm. ACM 15, (11), (November 1972), pp. 949-951.

(MIT, 1974) Massachusetts Institute of Technology, The Mathlab Group, Project MAC. Macsyma Reference Manual, (1974).

(Peucker and Chrisman, 1975) T.K. Peucker, and N. Chrisman. "Cartographic Data Structures", The American Cartographer 2 (1), (1975), pp. 55-69.

(Peuquet, 1983) D.J. Peuquet. "The Application of Artificial Intelligence Techniques to Very Large Geographic Databases", Proc. Sixth International Symposium on Automated Cartography (Auto-Carto Six), vol. 1, (16-21 October 1983), Ottawa, Canada, pp. 419-420.

(Randell et al, 1978) B. Randell, P.A. Lee, and P.C. Treleaven. "Reliability Issues in Computing System Design", ACM Computing Surveys 10, (2), (June 1978), pp. 123-165.

(Sedgewick, 1983) R. Sedgewick. Algorithms, (1983), Addison-Wesley Series in Computer Science.

(Spiegel, 1963) M.R. Spiegel. Theory and Problems of Advanced Calculus, Schaum's Outline Series, McGraw-Hill, New York, (1963).

(D. White, 1977) D. White. "A New Method of Polygon Overlay", An Advanced Study Symposium on Topological Data Structures for Geographic Information Systems, Oct. 16th through 21st, 1977, Laboratory for Computer Graphics and Spatial Analysis, Harvard University, Cambridge, MA, USA, 02138, (1977).

(M. White, 1983) M. White. "Tribulations of Automated Cartography and how Mathematics Helps", Proc. Sixth International Symposium on Automated Cartography (Auto-Carto Six), vol. 1, (16-21 October 1983), Ottawa, Canada, p. 408-314.

(Yohe, 1977) J.M. Yohe. The Interval Arithmetic Package, MRC Technical Summary Report #1755, Mathematics Research Center, University of Madison-Wisconsin, (June 1977).

ISARITHMIC MAPS AND GEOGRAPHICAL DISAGGREGATION

Dr. Erik Wallin
Lund University Computing Center
Box 783
220 07 Lund
Sweden

Introduction

In Sweden there are some 2 600 parishes. They constitute the most common areal sub-division for statistics on the Swedish socio-economic terrain. Some of the problems encountered when trying to display such socio-economic features in thematic, isarithmic maps are due to the fact that the form and the size of these parishes varies considerably. As shown in figure 1, high concentration of very small parishes is found in some parts of Sweden, whereas almost the whole northern part, especially in the mountain area, have a few number of very large parishes. This has severe effects on both the accuracy and the relevance of isoline construction in certain areas, when using traditional methods.

A series of thematic maps, based on parish data, were produced in 1974 to display some of the characteristics of the regional development in Sweden during the 60's and 70's. The isoline construction problem was 'solved' by simply masking the areas with too low density of parishes. When a new series of thematic maps were to be produced in 1982, almost ten years later, a more elaborated solution had to be found. An ad hoc-solution was found that might be of a more general interest. The method is based on a disaggregation and intrapolation technique, rather than aggregation and interpolation techniques commonly practiced. In this paper some of the details of the proposed method will be highlighted in relation to a discussion of the isarithmic technique in more general terms.

Isometric and Isoplethic Maps

Thematic mapping with the isarithmic technique presuppose the existence of a continuous spatial function having well-defined and unambiguous values for arbitrary points within a given territory -- the area to be mapped.

Most available isarithmic techniques, i.e. the interpolation and smoothing algorithms found in program packages like GD3, NAG/Graph or UNIRAS, presuppose that we have at our disposal a sample $(x,y,z)_k$ from a proposed continuous spatial function $z=f(x,y)$. The parameters of the proposed function are often estimated solely by taking the actual sample into account. To use these methods, we are, as it were, almost forced to pretend that we are dealing with point-related data and the