CALCULATING AREA and ANGLE of a NEEDLE-LIKE TRIANGLE

W. Kahan
Sept. 23, 1986

Given the side-lengths a, b, c of a triangle, familiar trigonometric formulas determine its area

$$\Delta := \surd(s(s-a)(s-b)(s-c)), \text{ where } s := (a+b+c)/2,$$

and the angle

$$\begin{aligned} C &:= \arccos((a^2+b^2-c^2)/(2ab)) \\ &= 2\arctan\surd((s-a)(s-b)/(s(s-c))) \end{aligned}$$

opposite side c. However, rounding errors can undermine the relative accuracy of those formulas so badly as to render them computationally useless whenever the triangle is very like a needle: that is, whenever two sides add up to scarcely more than the third. For instance, when c is very tiny compared with a and b, which must then be very nearly equal, roundoff in s can be almost as big as c, and then $(s-a)$ and $(s-b)$ can be left relatively inaccurate after cancellation; then too, the argument of arccos(...) can be so close to 1 that roundoff causes a relatively big error in its tiny computed value of C. Another defect in the formulas above is that they fail to warn of certain situations, like $a=-3, b=4, c=2$, when the given data violate the constraints

$$0 \le a \le b+c \text{ and } 0 \le b \le c+a \text{ and } 0 \le c \le a+b$$

that ought to be satisfied by the sides of a triangle.

The purpose of this note is to exhibit formulas that give results correct to almost as many significant figures as are carried during the computation, regardless of the shape of the triangle, and warn when the data cannot be the side-lengths of a real triangle. These formulas work correctly on almost all computers and calculators, including all big IBM and DEC computers and all HP calculators. (All exceptions are machines that lack a guard digit for subtraction.) The formulas help to correct two common misconceptions about floating-point computation; the first is that cancellation is *always* bad news; the second is that a singularity, where small changes in input data cause drastic changes in the desired result, *always* degrades accuracy.

Besides exhibiting the formulas, this note outlines a proof that they are correct despite roundoff, and displays numerical results obtained on an HP-97 calculator into which they were programmed.

How to compute Δ :

First sort a, b, c so that $a \ge b \ge c$; this can be done at the cost of at most three comparisons. If $c < a-b$ then the data are not side-lengths of a triangle; otherwise attempt to compute

$$\Delta := \surd((a+(b+c))(c-(a-b))(c+(a-b))(a+(b-c)))/4.$$

Do not remove parentheses from this formula! If $\sqrt{}$(negative) is encountered then the side-lengths do not belong to a triangle.

How to compute C :

If necessary, swap a and b so that $a \geq b$. Then perform two more comparisons to decide whether and how to compute α thus:

$$\begin{array}{lll} \text{If } b \geq c \geq 0 & \text{then} & \alpha := c - (a - b); \\ \text{if } c \geq b \geq 0 & \text{then} & \alpha := b - (a - c); \end{array}$$

if neither, then the data are not side-lengths of a triangle. Otherwise attempt to compute

$$C := 2\arctan(\sqrt{}(\alpha((a-b)+c))/\sqrt{}(((a-c)+b)(a+(b+c)))).$$

Once again, do not remove parentheses; and if $\sqrt{}$(negative) is encountered then the side-lengths do not belong to a triangle. Moreover, $\arctan(positive/0) = arctan(+\infty) = +\pi/2 = 90°$, but $\arctan(0/0)$ must be regarded as indeterminate.

Why Cancellation cannot hurt:

It is not hard to prove that if p and q are two floating-point numbers, and if $1/2 \leq p/q \leq 2$, then $p - q$ is a floating-point number too, representable exactly in the computer, unless it underflows. But we shall ignore spurious over/underflow phenomena since we can evade them by scaling the data except in the most extreme cases. And we shall assume that when $p - q$ is exactly representable as a floating-point number then subtraction will deliver that value exactly, as it does on all but a few aberrant machines. Therefore cancellation in $p - q$ introduces no new error that was not already present in p and q. We shall find that no error was already present when cancellation occurs in our formulas for Δ and C, so cancellation cannot hurt their accuracy.

Δ and C depend upon four factors each of the form $x \pm (y \pm z)$; if we prove that each factor is accurate to within a unit or two in its last significant digit carried, then we shall conclude easily that Δ and C must be accurate to within a few units in their last digits. The factors fall into two classes: Some are sums of two positive quantities each accurate to within a unit in its last digit; each such sum must obviously be accurate to within a unit or two in its last digit. The other factors are differences between two positive quantities each of which we shall show to be exact, so these factors will turn out to be accurate to within a unit in the last digit too. And then the factors must all be non-negative whenever the data satisfy the constraints that ought to be satisfied by the side-lengths of a triangle, as we shall now take for granted for the sake of argument.

Let us consider the factor $\alpha := c-(a-b)$ or $b-(a-c)$ according as $b \geq c$ or not. If $b \geq c$ then $c \leq b \leq a \leq b+c \leq 2b$, so $(a-b)$ must be exact. If $b < c$ then either $a < c$, in which case α is an accurate sum of two positive quantities, or else $a \geq c$ and then $b < c \leq a \leq b+c < 2c$, in which case $(a-c)$ must be exact. In all these cases, α must be accurate within a unit or two in its last digit, as claimed. Similar argumentation copes with the factor $(a-c)+b$ in the formula for C, and with the factor $c-(a-b)$ in the formula for Δ. All the other factors are sums of positive quantities.

When the data are not side-lengths of a real triangle, attempts to calculate $\surd(\ldots)$ will encounter a negative factor that will be correctly negative despite roundoff even if it is inaccurate.

Examples:
For each set of data a, b, c the values of Δ and C are exhibited as calculated on an HP-97 into which was programmed

$$\begin{aligned}
\Delta' &:= \surd(s(s-a)(s-b)(s-c)),\\
\Delta &:= \surd((a+(b+c))(c-(a-b))(c+(a-b))(a+(b-c)))/4,\\
C' &:= \arccos((a^2+b^2-c^2)/(2ab)),\\
C" &:= 2\arctan(\surd((s-a)(s-b))/\surd(s(s-c))) \text{ and}\\
C &:= 2\arctan(\surd(a((a-b)+c))/\surd(((a-c)+b)(a+(b+c)))).
\end{aligned}$$

The HP-97 carries ten significant decimal digits. The formulas for Δ and C were also programmed into an HP-71B , which carries twelve significant digits, to confirm that the final values of Δ and C (in degrees) were calculated correctly to at least nine significant digits on the HP-97.

a	b	c	Δ'	Δ	C' °	$C"$ °	C °
10	10	10	43.30127019	43.3012702*0*	60	60	59.9999999*8*
-3	4	2	*2.905*	Error	*151.045*	Error	Error
100000	99999.99995	0.00025	*15.8*	10	*Error*	1.*81*E-7	1.1459156E-7
100000	100000	1.00005	500*10.0*	50002.50003	*0*	5.73072E-4	5.7298644E-4
99999.99996	99999.99994	0.00003	*Error*	1.118033988	*0*	*Error*	1.2811726E-8
99999.99996	0.00003	99999.99994	*Error*	1.118033988	48.18968509	*Error*	48.18968510
10000	5000.000001	15000	*0*	612.372435*8*	180.00 *0*	180.00*0*	179.9985965
99999.99999	99999.99999	200000	*0*	Error	*180*	*180*	Error
5278.64055	94721.35941	99999.99996	*Error*	0	*Error*	*Error*	180
100002	100002	200004	0	0	*Error*	180	180
31622.77662	0.000023	31622.77661	0.*447*	0.327490458	*90*	*70.5*	64.22853824
31622.77662	0.015555	31622.77661	*246.18*	245.9460943	90.0*0*	89.9631*87*	89.9631515 *8*

Acknowledgements:
This note was prepared for an introductory class in Numerical Analysis. The accurate formula for C first appeared in the "Hewlett-Packard HP-15C Advanced Functions Handbook" (1982). The accurate formula for Δ first appeared in "Mathematics Written in Sand" in the Statistical Computing Section of the 1983 Proceedings of the American Statistical Association; that work had been supported by a grant from the U. S. Office of Naval Research, contract no. N 00014-76-C-0013.

A Distillation Program:

Given $N > 0$ and $x_1, x_2, \ldots, x_N$, we wish to replace them by N' and $x'_1, x'_2, \ldots, x'_{N'}$ in such a way as nearly minimizes N' yet keeps $x'_1 + x'_2 + \ldots + x'_{N'} = x_1 + x_2 + \ldots + x_N$ exactly, using only standard WP operations upon WP variables. We shall use a DP addition " $Z := p+q$ " of WP variables p and q to provide $Z = z+\zeta = p+q$ exactly while $z = [z+\zeta]$ rounded to WP ; this can be realized using only WP arithmetic as described earlier.

The first step is to sort $\{x_j\}$ by magnitude to ensure that $0 < |x_1| \le |x_2| \le \ldots \le |x_N|$. The sorting procedure used here should depend upon the provenance of the data: if $\{x_j\}$ is in no special order then a Heap-sort is appropriate; otherwise a Bubble-sort or Merge-sort may run faster.

The Distillation process proper consists of repeated summations alternating in direction from small-to-big, then big-to-small. Each summation leaves $\Sigma_j x_j$ unchanged. Each summation starts with $x_1 + x_2 + \ldots + x_{kl}$ and $x_{kr} + \ldots + x_{N-1} + x_N$ already distilled, and tracks changes in kl and kr . Here is the code:

```
Sort {$x_j$} so that $0 < |x_1| \le |x_2| \le \ldots \le |x_N|$ ; ... omit zeros
kl := 1 ;  kr := N ;
while  kl < kr  do {                                  ... Forward pass
     j := kl ;  p := $x_j$ ;  k := 0 ;
     for  i = j+1 to N  do {
          if  i>kr & j-1>k  then { $x_j$ := p ;  j := j+1 ;  p := $x_i$ }
          else { q := $x_i$ ;
                   Z := p+q exactly; ... = $z+\zeta$  with  $z = [z+\zeta]$
                   if  $\zeta = 0$  then  k := j
                   else { $x_j$ := $\zeta$ ;  if  z ≠ q  then  k := j ;
                            j := j+1 ;  if  k = 0  then  kl := j  ;
                   p := z } };
     kr := k ;  if  kl > 1 then  kl := kl-1 ;
     if  p = 0  then  N := j-1  else {  $x_j$ := p ;  N := j } ;
     if  kl < kr  then {                              ... Backward pass
          j := kr ;  p := $x_j$ ;  k := N+1 ;
          for i = j-1 to 1 step -1 do {
               if i<kl & j+1<k then { $x_j$ := p ; j := j-1 ; p := $x_i$ }
               else { q := $x_i$ ;
                        Z := p+q exactly; ... = $z+\zeta$  with  $z = [z+\zeta]$
                        if  $\zeta$ ≠ q  then  k := j ;
                        if  $\zeta = 0$  then  p := z
                        else { $x_j$ := z ;  j := j-1 ;  p := $\zeta$ ;
                                 if  k > N  then  kr := j } } } ;
          kl := k ;  if kr < N then kr := kr+1 ;
          if p = 0 then  j: = j+1  else  $x_j$: = p ;
          if  j > 1 then { j:=j-1 ; N:=N-j ; kl:=kl-j ; kr:=kr-j ;
                              for i = 1 to N  do  $x_i$ := $x_{i+j}$ } } }.
```

Implementation of " $Z := p+q$ " to produce exact	:	if $|p| < |q|$ then swap(p, q) ; ... so $|p| \ge |q|$ $z := p+q$; ... rounded to WP $\zeta := (p-z)+q$; ... turns out to be EXACT