[2.22] Methods for some CAS functions

The descriptions below are from a file on the TI web site at

ftp://ftp.ti.com/pub/graph-ti/calc-apps/info/92algorithms.txt

These descriptions probably apply to the original TI-92, and may not accurately describe the latest version of the 89/92+ CAS. Even so, they may be useful as a general indication of how the functions get their results.

arcLen()

arcLen(f(x),x,a,b) is merely $\int_a^b \sqrt{\left(\frac{d}{dx}(f(x))\right)^2 + 1}\, dx$

avgRC()

$$\text{avgRC}(f(x), x, h) = \frac{f(x+h)-f(x)}{h}$$

cFactor()
cFactor() is the same as factor() , except the domain is temporarily set to complex so that complex factors are sought and not rejected.

comDenom()
Term by term, comDenom(a/b+c/d) reduces (a*d+b*c)/(b*d) to lowest terms, and repeats this process for each additional term.

cSolve()
cSolve() is the same as solve() , except the domain is temporarily set to complex so that complex solutions are sought and not rejected.

cZeros()
cZeros(expr,var) is `expr▸list(cSolve(expr=0,var),var)`.

d() (symbolic differentiation)
d() repeatedly uses the chain rule together with the well-known formulas for the derivatives of sums, products, sinusoids, etc.

expand()
Polynomial expansion is done by repeated application of the distributive law, interleaved with use of the associative and commutative laws to collect similar terms.

factor()
The methods include the well-known factorizations of binomials and quadratics, together with methods described by Geddes, Czapor and Labahn, *Algorithms for Computer Algebra*, Kluwer Academic Publishers, Boston, 1992. Laguerre's method is used for approximate polynomial factorization. (Press et. al: *Numerical Recipes* Cambridge University Press, 1986.)

fMax()
See fMin()

fMin()
For fMin(expr,var), if d(expr,var) cannot be computed symbolically, a combination golden-section parabolic interpolation search is performed for one local minimum. (R. Brent, *Algorithms for Minimization without Derivatives*, Prentice-Hall, 1973.) Otherwise, fMin(expr,var) is determined by solve(d(expr,var)=0,var), filtered by attempting to determine the signs of higher-order derivatives at these candidates. Surviving candidates are compared with the limits of *expr* as *var* approaches +infinity and -infinity; and also with points where *expr* or its derivative is discontinuous.

integration (symbolic)
Antiderivatives are determined by substititutions, integration by parts, and partial-fraction expansion, much as described by J. Moses (*Symbolic Integration: The Stormy Decade*, Communications of the ACM, August 1971, Volume 14, Number 8, pp. 548-559). Antiderivatives are NOT computed by any of the Risch-type algorithms. Definite integrals are determined by subdividing the interval at detected singularities, then for each interval, computing the difference of the limit of an antiderivative at the upper and lower integration limits. Except in exact mode, *nINT()* is used were applicable when symbolic methods don't succeed.

limit()
Limits of indeterminate forms are computed by series expansions, algebraic transformations and repeated use of L'Hopital's rule when all else fails. Limits of determinate forms are determined by substitution.

nDeriv() (numeric derivative)

$$\text{nDeriv}(f(x), x, h) = \frac{f(x+h)-f(x-h)}{2h}$$

nInt() (numeric integration)
nInt() uses an adaptive 7-15 Gauss-Kronrod quadrature somewhat like the algorithm described in *Numerical Methods and Software*, by David Kahaner,Stephen Nash,Cleve B. Moler,George E. Forsythe and Michael A. Malcolm, Prentice Hall 1989.

nSolve() (numeric solve)
nSolve() uses a combination of bisection and the secant method, as described in Shampine and Allen's *Numerical Computing: An Introduction*, W.B. Saunders Co., 1973.

Product function $\Pi()$ *(symbolic)*
Iterated multiplication is used for modest numeric limits. Other products are determined by checking for telescoping products and simple patterns. Limits are used for infinite product limits.

propFrac() (proper fraction)
propFrac() uses polynomial division with remainder to determine the quotient polynomial and the numerator of the resulting fraction.

solve() (symbolic solve)
solve() uses factoring together with the inverses of functions such as ln(), sin(), etc. Except in exact mode, these techniques might be supplemented by heuristics to isolate solutions, followed by nSolve() to refine approximations.

Summation function Σ() (symbolic)
Iterated addition is used for modest numeric limits. Other sums are determined from formulas given in references such as "Standard Mathematical Tables and Formulae", CRC Press, together with tests for telescoping sums. Limits or special formulas are used for infinite summation limits.

taylor() (taylor series)
taylor(u,x,n,a) is Σ(limit(d(u,x,k),x,a)*(x-a)^k/k!,k,0,n), computed iteratively to avoid redundant computation of lower-order derivatives.

tCollect() (trig collect)
tCollect() repeatedly uses the well-known formulas for products and integer powers of sines and cosines, such as those given in the CRC *Standard Mathematical Tables and Formulae*.

tExpand() (trig expand)
tExpand() repeatedly uses the well-known formulas for the sine and cosine of multiple angles, angle sums and angle differences, such as those given in the CRC CRC *Standard Mathematical Tables and Formulae*.

zeros()
zeros(expr,var) is expr▸list(solve(expr=0,var),var)