Some Improved Methods for The Determination of a Student's t-Quantile

by
James R Warren BSc MSc PhD PGCE

On rare occasions we are luckily enabled to discard futile custom and sterile evolution and acclaim a new tool that does the job ten million times better and twice as fast. Such is now our pleasure.

Readers who studied my paper "A Description of Program BELEM.BAS" may recollect my tactics for the determination of t ancillary to the statistical treatment of locational data.

Hill's Routine[1] offers a five-figure approximation for t given three or more degrees of freedom and an arbitrary probability. The process selects one of two series expansions for t according to a secondary criterion that is satisfied for most low probabilities and high degrees of freedom. When this criterion applies the second expansion proceeds very slowly relative to the first, a sluggishness exacerbated by the requirement for the determination of a negative normal deviate. Hill makes no recommendation for finding this deviate in his 1970 paper.

In this treatment I compare the efficiency of two methods of determining a negative normal deviate and, more importantly, I educe a general method for numerically finding g=f(x) for any continuous f(x) which I apply to determining the t-quantile for a given probability when degrees of freedom is fixed.

This latter process, which I call KEDGING, allows arbitrarily-high precision limited only by the capabilities of your equipment.

In particular, it gives at least seven orders of magnitude more accuracy than Hill's Process whilst taking half the time.

The Kedging Process

Kedging is an interpolative iteration based upon a coupling of a mechanism to determine f(x) for x linked to a Lagrangian interpolation which determines a revised x' value by interpolation of the f(x) values to a given f'(x). The primary mechanism for f(x) may be a numerical integration, a power series or any appropriate and efficacious algorithm.

In practice, some (non-critical) starting approximation of x is established by ancillary means and the error bound ε associated with this value.
f(x-ε), f(x) and f(x+ε) are then computed and the Lagrangian interpolation applied on

a four-point basis to fix the new x' associated with the known and constant f'(x).

This equivalates fitting a quadratic curve to the points and reading the fourth point at the required abscissa. It is now assumed that the new and much smaller ε is $|x-x'|$ as a new cycle is entered.

No more than two or three iterations of this loop suffice to generate a t which gives a probability estimate within 10^{-13} of the true feed probability for $0.25 \leq p \leq 0.000001$ and $3 \leq v \leq 30$. Two iterations are known to yield at least twelve-figure accuracy. Kedging will quickly drive itself into the ground even using QBASIC double-precision arithmetic so we need to incorporate safeguards against both division-by-zero in the Lagrangian interpolation (due to vanishment of ε) as well as against the associated effect of subcritical $|x-x'|$.

This interpolative iteration is realised as REFINERY* in the Program TTIMER.BAS of Appendix One and the Segments REFINERY* in Program New BELEM.BAS of Appendix Two.

I have called this process KEDGING by analogy to the technique of that name resorted to by the old time sailing masters if they fell becalmed in shallows. They would have the anchor rowed a few hundred meters ahead in a cutter and then dropped to the sea bed. A party aboard the ship would then haul on the anchor capstan until the craft drew up to and lifted dry the anchor. Repetition then secured another tedious increment of uncial progress. I think the mining students amongst you will prefer the illustration of the Victorian roller crusher whose orbiting raff wheel returned the coarse comminute to the mill for another pass between the cylinders until it was small enough to fall through the sieve.

PHASE I

The Formulaic Bases of the Tested Solutions

Hill's Routine

Hill's Routine, or Hill's Process, is coded in QBASIC in Segments T1, T2 and T3 of TTIMER.BAS and its logic will not algorismically be developed here.

T1 uses a 64-interval Romberg integration within Function TND to form the negative normal deviate whilst T2 applies UND to determine the deviate using a rapidly-convergent infinite series. T3, which refines the Hill's Process output t using kedging, also employs UND.

Hastings' Polynomial for the Normal Deviate

Functions TND and UND are both primed by this approximator which is embodied in Function QND. This polynomial is Object 26.2.23 on Page 933 of Abramowitz and Stegun[2] and may be written:

$$d = U - \frac{2.515517 + 0.802353U + 0.010328\ U^2}{1 + 1.432788U + 0.189269\ U^2 + 0.001308\ U^3} \qquad \textbf{Eqn.1}$$

where:-

$$U = \sqrt{\log_n \left(\frac{1}{p^2} \right)} \qquad \textbf{Eqn.2}$$

The error due to this approximation is ±0.00045.

Romberg Integration

Romberg integration of sixty-five Gaussian Density Function ordinates such that:-

$$F = e^{-\frac{1}{2}z^2} \qquad \textbf{Eqn.3}$$

is implicated by T1. The integration is applied on the interval between zero and the estimate of the *positive* normal deviate.

Segment ROMBERG implements the process that will not algorismically be developed here. Interested readers may glean sufficient insight into the algorithm from "Basic Numerical Methods" by Scraton[3].

The Normal Deviate by Finite Elaboration
of its Infinite Series

Several authorities give the following formula for the normal deviate. It is Object 26.2.10 in Abramowitz and Stegun:-

$$P(d) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n d^{2n+1}}{n!\ 2^n (2n+1)} \qquad \textbf{Eqn.4}$$

Empirical trials by the Author undertaken for several deviations d and several accuracies a (expressed as numbers of correct decimal places) established the following table of necessary and sufficient expansion terms m:-

		Precision a									
		0	1	2	3	4	5	6	7	8	9
Deviate d	1	0	0	2	3	4	4	5	6	7	8
	2	0	3	5	7	8	9	11	12	13	14
	3	5	10	12	14	15	17	19	20	21	23
	4	16	19	21	23	25	27	28	30	32	33
	5	28	31	33	35	37	39	41	43	44	46

Table A
The Number of Iterations to Determine a Normal Deviate

It can be seen from Table A that more than 46 iterations are never required to compute a normal deviate under meaningful circumstances.

The Author plotted these results and inspectionally fitted straight-line envelopes in order to develop the following empirical equation for pre-selection of the necessary and sufficient iterations m:-

$$m = \text{entier}(0.8 + d^{2.09414342\,149} + 0.921892385\,ad^{0.54816927\,818}) \qquad \textbf{Eqn.5}$$

This fitment deliberately overspecifies m but the number of superfluous iterations never exceeds three for one or more significant figures of accuracy.

Subroutine GND implements this process.

Computing The Probability Associated with a Quantile
Using The Finite Trigonometric Series Method

Objects 26.7.3 and 26.7.4 on Page 948 of Abramowitz and Stegun are finite trigonometric series for computing the probability associated with any t-quantile.

Let auxiliary variables θ and n be declared as:-

$$\theta = \text{ArcTan}\left(\frac{1}{\sqrt{\nu}}\right) \qquad \textbf{Eqn.6}$$

and:-

$$n = \text{entier}\left(\frac{\nu}{2} - 1\right) \qquad \textbf{Eqn.7}$$

Then 26.7.3 for odd degrees of freedom may be given in condensed form as:-

$$p = \frac{2}{\pi}\left[\theta + \text{Sin}\,\theta\left(\sum_{k=0}^{n}\frac{\prod_{j=1}^{k} 2j}{\prod_{i=1}^{k} 2i+1}.\text{Cos}\,\theta^{2k+1}\right)\right] \qquad \textbf{Eqn.8}$$

Whilst the sister series 26.7.4 for even degrees of freedom may be written:-

$$p = \text{Sin}\,\theta\left[1 + \sum_{k=1}^{n}\frac{\prod_{j=1}^{k} 2j-1}{\prod_{i=1}^{k} 2i}.\text{Cos}\,\theta^{2k}\right] \qquad \textbf{Eqn.9}$$

Equations Six, Seven, Eight and Nine are implemented in Function TPROB and compute the associated probability exactly.

Readers are reminded that analytic t's exist for one and two degrees of freedom rendering respective applications of Equations Eight and Nine redundant.

The Asymptotic Expansion to Approximate t

Object 26.7.5 on Page 949 of Abramowitz and Stegun is a nest of polynomial terms for an asymptotic expansion in approximation of a t-quantile to three-figure accuracy.

The Normal Deviate x_p is approximated using Function QND (the Hastings Polynomial) whereupon:-

$$t_p \approx x_p + \frac{g_1(x_p)}{\nu} + \frac{g_2(x_p)}{\nu^2} + \frac{g_3(x_p)}{\nu^3} + \frac{g_4(x_p)}{\nu^4} \qquad \textbf{Eqn.10}$$

and:-

$$g_1(x) = \frac{1}{4}(x^3 + x) \qquad \textbf{Eqn.10a}$$

$$g_2(x) = \frac{1}{96}(5 x^5 + 16 x^3 + 3x) \qquad \textbf{Eqn.10b}$$

$$g_3(x) = \frac{1}{384}(3 x^7 + 19 x^5 + 17 x^3 - 15x) \qquad \textbf{Eqn.10c}$$

$$g_4(x) = \frac{1}{92160}(79 x^9 + 776 x^7 + 1482 x^5 - 1920 x^3 - 945x) \qquad \textbf{Eqn.10d}$$

This system is implemented as Function TAPPROX which generates a convenient starter t for the kedging of Method Four.

Methodological Comparisons

Table One of Appendix Five embodies the interpreted Program TTIMER.BAS timings to compute t-quantiles corresponding to a standard set of twenty probabilities set by Subroutine ALASSIGN. These spread across the range from 0.25 to 0.0000001 and the source code commentary of Appendix One may be consulted for specifics.

Further timings were undertaken for determinations at ν=5,10,15,20,25 and 30 for these twenty probabilities giving 120 t-determinations. Such screen-presented timings are quoted in Table One with the screen-determined timings at point ν's: And separately in Appendix Four for file-committed material. Readers should note that whilst all timings are quoted to eight figures for analytic reasons no time is more accurate than to three figures. Henceforth the 120-value series of Appendix Four will be referred to as "The Gross Timings" and the percentage of a Method's elapsed time for that series relative to the time to elaborate Method Three (100%) as "The Gross Percentage".

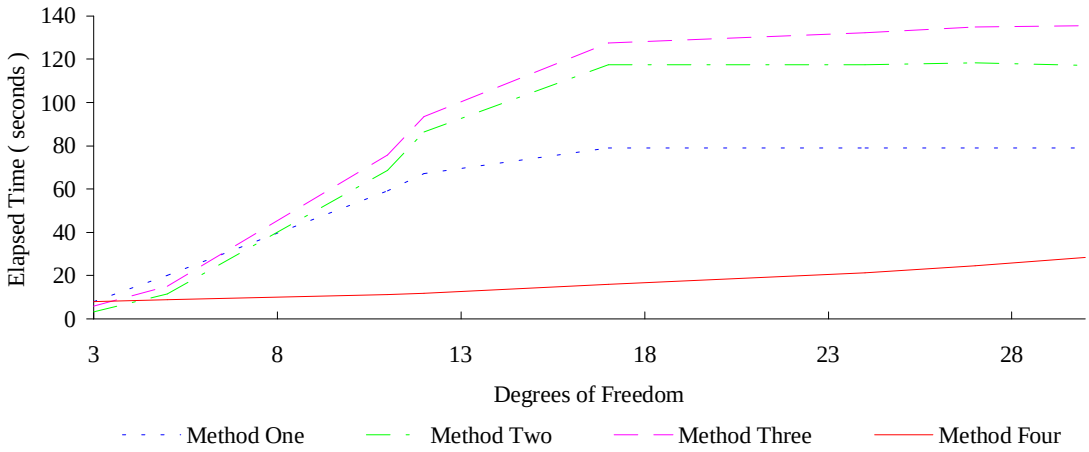
Method One

This technique is a straight five-figure Hill Approximation. When a normal deviate is needed its Hastings Approximation is kedged to twelve figures using a Romberg Integration of the Gaussian Ordinates.

Reference to the execution time graph of Figure One shows that whilst this is the costliest method for ν=3 a moderate linear climb is made to ν=12 when the cost converges to a constant level of 78.8 seconds at ν=17.

The Gross Time is 388.6484375 seconds, 64.67%.

THE GRAPH OF EXECUTION TIME VERSUS DEGREES OF FREEDOM
For Four Methods of t-Quantile Computation
Figure One



Method Two

This is the same Hill's Approximation as Method One except that Romberg is dropped in favor of the infinite series for the normal deviate truncated according to the scheme of Equation Five. These values are then kedged to twelve figures.

The Figure One curve shows this method to be the cheapest for $\nu=3$ and $\nu=4$ but there is a steep if unsteady cost climb to $\nu=17$ where elaboration times top out at 117.2 seconds.

The Gross Time is 536.90234375 seconds, 89.33%.

Astonishingly, the series summation is more expensive than Romberg for all $\nu>8$ and leads to solutions 48.7% more costly in time for all $\nu>16$.

Method Three

This is a twelve-figure technique based upon Method Two. The resulting five-figure Hill t-quantile is kedged until it becomes a twelve-figure approximation.

Figure One shows how it shadows the Method Two curve, naturally at a slightly higher and more expensive level. The final kedging overhead is not constant but rises slightly with increasing ν. This is the most expensive option for any $\nu>6$.

The two finite Hill Methods stabilise in terms of time cost at $\nu=17$ but that does not apply to Method Three (or Method Four).

Gross Time is 601 seconds, 100%.

Method Four

Method Four is not a Hill Process: It is almost pure kedging. It commences with a three-figure approximation to t using the Asymptotic Expansion

of Equation Ten, which is in turn fed an approximate normal deviate by the Hastings Polynomial. (This is a bit superfluous: kedging converges almost as quickly if the starter t is unity). The three-figure approximation is then kedged to twelve-figures employing the finite trigonometric series to supply reference probabilities.

Figure One shows that this is much the cheapest method for ν in the range 5 to 30 with only a gradual non-linear rise in cost.

Gross Time is 102.109375 seconds, 16.99%.

PHASE II

The series elaborations of Phase One are inefficient in several regards. Methods of improvement are indicated by consulting an empirical timing tariff for *interpreted* QBASIC upon my old 20MHz Tiko:-

OPERATION	ELAPSED TIME
Carcase	5.16015625
Brackets	0.01171875
Powers	19.5
Divisions	0.98828125
Multiplications	0.94140625
Additions	0.76953125
Subtractions	0.828125
Exponentiations of the Napierian Base	12.69140625
Logarithms to the Napierian Base	7.359375
Square Roots	0.828125
Cube Roots	19.44140625
Squares	19.71875
Cubes	19.73046875

Table B
Operational Timing Tariff

Each timing is in seconds for 5000 elaborations of the indicated

operation and is discounted for the execution time of the loop carcase, a matter of some 5.16 seconds.

It is very likely that the relative expense of these operations will be conserved both for *compiled* code and for enhanced clock speeds, and it is to these relativities that we shall appeal in planning enhancements.

First note that cardinal operations are about the same except that a division is about 18.6% longer than a multiplication. A square root identifies with a subtraction. Bracketing carries negligible overheads.

Secondly (except for square roots controlled by the intrinsic function) any raising to a power is at least twenty-two times as protracted as a multiplication confirming with some force the zipperhead dictum: "Do not power it: Times it".

The QBASIC trace indicated that solutions for t which implicated factorial computation became mired in the repeated multiplications, which developed the function ab initio when ever a new term of the convergent was built. Since each term employs a factorial based upon its sequential position the more efficient strategy is to augment a remembered product with a single multiplication.

The inefficient strategy implicates $(n^2-n)/2$ multiplications for each new $\Sigma i!$ whereas the efficient strategy incurs but one multiplication.

Further improvements can be made by substituting products for powers. Relative to the denominator of the Gaussian probability's infinite series we may also note that:-

$$2^n = 2 \times 2^{n-1} = 2 \times 2^n \qquad \textbf{Eqn.11}$$

and accordingly a new power of two may be computed by doubling its precursor.

Thirdly, the power of the deviate may be addressed by noting that each successive member of the series $d^1,d^3,d^5,d^7,...$ may be developed by multiplying its precursor by d^2, which is in turn $d \times d$, computed once and recorded in RAM. Therefore m powers become m products.

All these curtailments abridge only time without vitiating accuracy and are assimilated to Equation Four as:-

$$P(d) = \frac{1}{2} - \frac{1}{\sqrt{2\pi}} \sum_{n=1}^{m} \frac{-i \times d \times d^2}{(j+2) \times n! \times n \times 2 \times 2^n} \qquad \textbf{Eqn.12}$$

where appropriate entry values are defined for precursor functions.

Equation Twelve was implemented as Segment GND1.

The finite series for computing the Probability associated with the t-Quantile as given by Equations Eight and Nine is susceptible of similar improvements. Clearly the two product series under the summations are analogous to

factorials whilst the exponential cosine might also be reduced to a repeated product. If such improvements can be made timing improvements will extend to the pure kedging methodology of Method Four as well as the Hill's Process based techniques.

As implemented in Segment TPROB the summative component of Equations Eight and Nine was:-

$$G = \sum_{k=k_s}^{n} \frac{\prod_{i=1}^{k} 2i + i_m}{\prod_{i=1}^{k} 2i + j_m} . \mathrm{Cos}\theta^{2k+k_m} \qquad \textbf{Eqn.13}$$

This series was re-configured as:-

$$G = \sum_{k=1}^{n} \frac{u_{-} x \mathrm{Cos}(\theta)^2 \, \mathrm{x} \, \mathrm{d}_{-} \mathrm{x}(\mathrm{j}_{-} + 2)}{\mathrm{v}_{-} \mathrm{x}(\mathrm{i}_{-} + 2)} \qquad \textbf{Eqn.14}$$

where appropriate entry values are defined for precursor functions.
Equation Fourteen was implemented in Segment TPROB1.

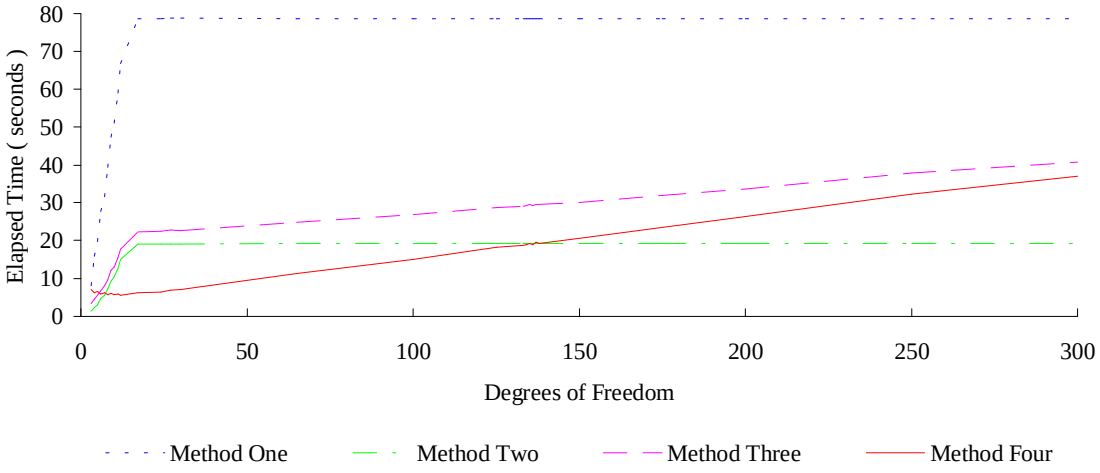
Methodological Comparisons

Table Three in Appendix Five embodies the interpreted Program TTIMER.BAS timings to compute t-quantiles corresponding to a standard set of twenty probabilities set by Subroutine ALASSIGN: These spread across the range from 0.25 to 0.0000001 and the source code commentary of Appendix One may be consulted for specifics.

The timings of Table Three were screen-presented at the indicated degrees of freedom and only three figure accuracy should be assumed, though more digits of precision are given in case they are analytically useful.

The three methods can be compared by reference to Figure Three "The Graph of Enhanced Execution Time Versus Degrees of Freedom".

THE GRAPH OF ENHANCED EXECUTION TIME VERSUS DEGREES OF FREEDOM
For Four Methods of t-Quantile Computation
Figure Three



Method One, which found the normal deviate via Romberg integration, is unimproved and clearly much the most expensive method at any ν, topping out at 78.65 seconds for 17 to 24 degrees of freedom. The Gross Time is 385.30859375 seconds, 355.92%.

Method Two is also Hill's Process but with the normal deviate found by series summation. This is the cheapest process for $\nu<8$ but grows steeply more expensive until $\nu=17$ where it tops out at some 19 seconds. The Gross Time is 89.41796875 seconds, 82.60%.

Method Three refines a Hill's Process estimate to twelve figure accuracy by kedging. It parallels Method Two to $\nu=17$ with a kedging overhead but when Method Two tops out a linear kedging penalty continues its shallow growth to and beyond $\nu=300$. The Gross Time is 108.2578125 seconds, 100%.

Method Four is pure kedging utilising the finite trigonometric series for the probability associated with the t-quantile. Figure Three shows how this method is the cheapest alternative in the region of principal interest for practical application of Student's t between $\nu=8$ and $\nu=134$. The cost minimum is at $\nu=12$. There is then a linear time cost rise from 6.21 seconds at $\nu=17$ to 36.97 seconds at $\nu=300$. It appears that extrapolation of the Method Three and Method Four curves would lead to a crossing around $\nu=360$ suggesting that preliminary Hill's Process approximation to provide a starter for kedging (rather than the 26.7.5 asymptotic expansion of Method Four) would give time savings for computing quantiles at $\nu>360$. In such regions, however, many practitioners are ready to apply The Gaussian Distribution to parametric problems. In any case, QBASIC elaboration founders due to overflow in the expansion of the trigonometric series when $\nu=325$ is reached. The Gross Time is 38.05859375 seconds, 35.16%.

PHASE III

Review of Equations Eight and Nine and especially of their summative kernel Equations Thirteen and Fourteen confirms that a pair of product series are generated whenever a t-associated probability is determined.

These product series grow rapidly and one or the other occasions overflow in QBASIC double-precision arithmetic when ν reaches about 320 and it is this overflow which accounts for Method Three founderings illustrated by Table Three in Appendix Five. Since the summative kernel involves the division of a pair of product series and since multiplication and division are commutative we can mitigate the rate of growth of the summation terms by dividing successive product series terms as they are computed and developing a composite product series as a product of the dividends.

This is equivalent to:-

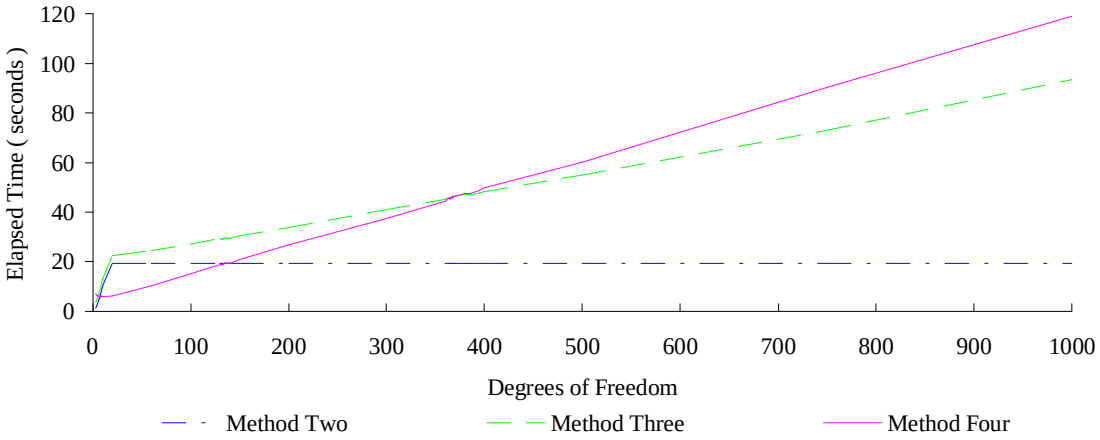
$$G = \sum_{k=k_s}^{n} \prod_{i=1}^{k} \frac{2i + i_m}{2i + j_m} \cdot \mathrm{Cos}^{2k+k_m} \qquad \textbf{Eqn.15}$$

which is implemented in Segment TPROB2.

This device does not improve either timings or accuracy but enables probabilities to be determined at least until ν=1511 and has been well tested in the TTIMER carcase up to ν=1000.

For various ν the results of this improvement are tabulated in Table Four of Appendix Five whose results are illustrated in Figure Four.

ENHANCED EXECUTION TIME VERSUS DEGREES OF FREEDOM
WITH PRODUCT SERIES PRE-DIVISION
For Three Methods of t-Quantile Computation
Figure Four



Method One

Method One cannot be enhanced in these terms and is discounted.

Method Two

Method Two is an ordinary five-figure Hill's Routine and tops out at about 19.28125 seconds at ν=20.

The Gross Time is 90.08203125 seconds, 82.50%.

Method Three

This is twelve-figure kedging started by Hill's Process and times rise steeply to ν=20 where a shallow quasi-linear climb supersedes until about 93.48 seconds are consumed for ν=1000.

The Gross Time is 109.19140625 seconds, 100%.

Method Four

This is twelve-figure kedging started by the three-figure Abramowitz and Stegun 26.7.5 asymptotic expansion.

It remains absolutely the cheapest method between ν=8 and ν=132 and is cheaper than the Hill-pilot kedge Method Three until ν=365.

Method Four is accordingly the most economical high-precision technique of t-quantile determination throughout the entire domain of prime interest between ν=8 and ν=365.

The Gross Time is 38.33984375, 35.11%.

A Hybrid Software Module

Further inspection of Figure Four shows that there is considerable noise on the timing curves around the Method Three-Method Four crossover and that

noise is likely to exist elsewhere but be masked by more distantly-separated sampling points. The figures in Table Four show considerable quantisation and I guess that both the noise and the quantisation are artifacts of electrical fluctuations in my computer system.

Above about forty degrees of freedom both the Method Three and Method Four curves *appear* to increase with a very shallow accelerative sweep suggestive of nlogn growth.

Nevertheless, rectilinearity can be assumed between ν=200 and ν=500 as a convenient basis for locating the crossover. Furthermore, 19.28125 seconds may be defended as the representative Method Two timing for ν>59.

Using the times tabulated in Table Four for Method Two and for ν=200 and ν=500 for Methods Three and Four we may develop the following equations:-

$$s_2 = 19.28125 \qquad \textbf{Eqn.16.2}$$

$$s_3 = 19.7552083\ 333 + 0.07013020\ 833\nu \qquad \textbf{Eqn.16.3}$$

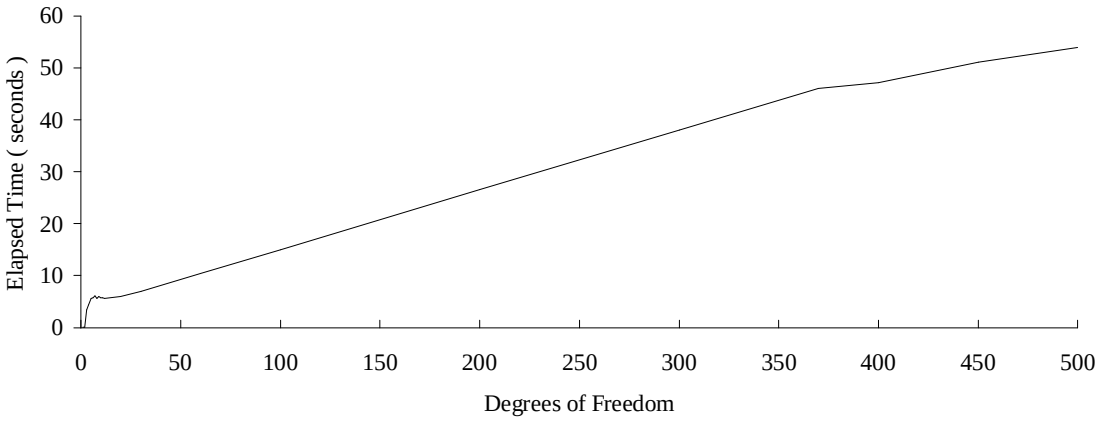
$$s_4 = 4.45833333\ \ 3333 + 0.11145833\ \ 333\nu \qquad \textbf{Eqn.16.4}$$

Simultaneous solutions of Equations 16.2 and 16.4 indicate that the Method Two-Method Four crossover is at ν=132.99065421 (ν=133) whilst simultaneous solutions of Equations 16.3 and 16.4 indicate that the Method Three-Method Four crossover is at ν=370.132325111 (ν=370).

Segment THYBRID of Program THYBRID.BAS employs these crossovers as the bases of selection rules for the most efficient t-quantile computation algorithm. If ν=1 or 2 then appropriate closed-form equations are solved whilst for ν=6 through 370 kedging started by 26.7.5 asymptotic expansion is used. Otherwise we revert to kedging started by Hill's Process.

Table Five of Appendix Five embodies the timings provided by this composite at various degrees of freedom and Figure Five shows the way in which timings are quasi-constant from ν=5 to 20 and rise linearly to about 46 seconds at ν=370 and then break to a shallower linear regime until time s=53.94 seconds at ν=500.

ELABORATION TIME VERSUS DEGREES OF FREEDOM
FOR TWENTY STUDENT'S t-QUANTILES
FOR THE MIXED METHOD ALGORITHM OF THYBRID.BAS
Figure Five



Conclusions

Hill's Process for the t-quantile which only ever aspired to five-figure accuracy is made redundant by my kedging technique which achieves arbitrary accuracy very quickly.

In particular, it achieves twelve figure accuracy in t-quantile estimation in 42.56% of the time required for a five-figure Hill's Process at thirty degrees of freedom. It is consistently cheaper than that basic Hill's Process throughout the eight to thirty degrees of freedom range in which t-quantiles have their greatest statistical interest, and consistently at least seven orders of magnitude more accurate.

With respect to comparable twelve-figure methods, kedging can be primed either using Hill's five-figure estimation algorithm or the Abramowitz and Stegun 26.7.5 asymptotic expansion. The Hill's-piloted method is only faster, however, for $\nu>370$ leaving 26.7.5-primed kedging the only practical choice between $\nu=3$ and $\nu=370$. In the five to thirty degrees of freedom range overall timings are 35.11% of those for the Hill's Process piloted algorithm. It is only in the regions of high ν that the superior initial accuracy of Hill's Process pays dividends in elaboration time.

Hill's Process was essentially obsolete when it was published in 1970. My mathematical kedging centers upon Lagrangian Interpolation, a technique developed when kedging at sea was an essential recourse in extremis. The inventor of Lagrangian Interpolation, the Italian mathematician Joseph Louis Lagrange, died in 1813. Eighteenth-century methods can often furnish solutions to twenty-first century problems.

The timings presented are for *interpreted* QBASIC code run upon my

obsolete (but very trusty) Tiko 386-sx rated at 20MHz. When running on a modern 486, EXCEL furnishes the t-quantile to fifteen figures with sensible instantaneity. I would like to know which algorithm the MicroSoft programmers used. Their product is of course compiled.

NOTE:-

Certain timings cited in this report may vary slightly from those listed in the appendices.

This is because of minor coding enhancements which I install in pursuit of a policy of continuous improvement.

Notation

$_-$	(Subscript) The Serial Precursor of
a	The Number of Digits Precision in the Mantissa
d	Hastings' Approximate Normal Deviate
d	The Series Normal Deviate
d	The Normal Deviate
d_-	The Serial Precursor of the Normal Deviate
ε	The Error Bound
e	The Napierian Base
f(x)	The Function of Variable x
f'(x)	The Function of The Ultimate Estimate of x (a known constant)
g	A Secondary Function of x
$g_i(x)$	The ith. Secondary Function of x
G	The Trigonometric Series Summation
θ	The Trigonometric Function of ν
i	A Trigonometric Series Product Formation Auxiliary
i_-	The Serial Precursor of i
j	A Trigonometric Series Product Formation Auxiliary
j_-	The Serial Precursor of j
j_m	The Denominator Product Term Adjustor
k	A Trigonometric Series Product Formation Auxiliary
k	The Product Upper Bound
k_m	The Cosine Exponent Adjustor
k_s	The Trigonometric Series Summation Lower Bound
m	The Necessary and Sufficient Iterative Terms
ν	The Degrees of Freedom
n	A Series Term Serial Number
n	The Trigonometric Series Summation Limit
π	The Ludolphine Constant
p	The Probability (associated with ν and t)
P(d)	The Probability Associated with The Series Normal Deviate
s	A Timing
s_2	The Elaboration Time for Method Two
s_3	The Elaboration Time for Method Three

s_4	The Elaboration Time for Method Four
t	The Student's t-Quantile
t_p	An Approximation to the Quantile Associated with p
u_-	The Precursor Value of the Cosine Exponentiation
U	Hastings' Probability Function
v_-	The Precursor Value of the Denominator Product Term
x	The First Estimate of x
x	The Independent Variable
x'	The Second Estimate of x
x_p	The Approximate Normal Deviate Associated with p
z	The Normal Deviate (Standard Score)

References

1 "Algorithm 396:Student's t-Quantiles [S14]"
GW Hill
Communications of the ACM
V13 N10 pp619-620 October 1970

2 "Handbook of Mathematical Functions"
Edited by Milton Abramowitz and Irene A Stegun
Dover Publications of New York 1972
ISBN 0-486-61272-4

3 "Basic Numerical Methods"
RE Scraton 1984
Edward Arnold of London
ISBN 0-7131-3521-2

Appendix One

Program TTIMER.BAS

```
'      PROGRAM TTIMER.BAS
'      A PROGRAM TO COMPARE THE SPEED OF SOME METHODS OF DETERMINATION
'      OF THE STUDENT'S t-QUANTILE FOR A RANGE OF PROBABILITIES AND
'      DEGREES OF FREEDOM
'
'      FOUR METHODS ARE TESTED:-
'         1   FIVE-FIGURE ACCURACY
'             HILL'S METHOD RESOURCES A NEGATIVE NORMAL DEVIATE DETERMINED BY
'             HASTINGS 26.2.23 REFINED BY AN INTERPOLATIVE ITERATION WITH
'             ORDINATES ERECTED BY ROMBERG INTEGRATION
'         2   FIVE-FIGURE ACCURACY
'             HILL'S METHOD RESOURCES A NEGATIVE NORMAL DEVIATE DETERMINED BY
'             HASTINGS 26.2.23 REFINED BY AN INTERPOLATIVE ITERATION WITH
'             ORDINATES ERECTED BY USE OF THE CONVERGENT SERIES FOR
'             GAUSSIAN PROBABILITY
'         3   TWELVE-FIGURE ACCURACY
'             HILL'S APPROXIMATION IS DETERMINED RESOURCING A NEGATIVE
'             NORMAL DEVIATE DETERMINED BY HASTINGS 26.2.23 REFINED BY AN
'             INTERPOLATIVE ITERATION WITH ORDINATES ERECTED BY USE OF THE
'             CONVERGENT SERIES FOR GAUSSIAN PROBABILITY.
'             HILL'S APPROXIMATION IS THEN ITSELF REFINED AGAINST THE
'             TARGET PROBABILITY USING AN INTERPOLATIVE ITERATION WHOSE
'             ORDINATES ARE ERECTED USING THE FINITE TRIGONOMETRIC SERIES
'             FOR THE PROBABILITY ASSOCIATED WITH THE STUDENT'S t-QUANTILE
'         4   FIFTEEN-FIGURE ACCURACY
'             THE SERIES APPROXIMATOR FOR THE t-QUANTILE 26.7.5 IS APPLIED
'             TO DEVELOP NEAR THREE-FIGURE ACCURACY.
'             THIS ESTIMATE IS THEN REFINED USING AN ITERPOLATIVE ITERATION
'             WHOSE ORDINATES ARE ERECTED USING THE FINITE TRIGONOMETRIC
'             SERIES FOR THE PROBABILITY ASSOCIATED WITH THE STUDENT'S
'             t-QUANTILE
'
'      WRITTEN BY:-
'
'            JAMES R WARREN BSc MSc PhD PGCE
'            "SOUTHGATE"
'            31 VICTORIA AVENUE
'            BLOXWICH
'            WS3 3HS
'            UNITED KINGDOM
'
'            14 OCTOBER 1996
'
'      THIS PROGRAM IS WRITTEN IN MICROSOFT QBASIC
'
' VARIABLE TYPE DEFAULTS
      DEFDBL A-H, O-R, T-Z
      DEFSTR S
      DEFINT I-K, M-N
      DEFLNG L
' SEGMENT DECLARATIONS ( THEMATIC GROUPAGE )
      DECLARE SUB ALASSIGN (NA, AL())
      DECLARE SUB CAPTION (ME, IL(), SB())
      DECLARE SUB HEAD (ME)
      DECLARE SUB TAIL (ME, IL(), SB(), NP, TR)
      DECLARE SUB TIMELOG (ME, IL(), SB(), NA, AL(), JL, JU, JS)
      DECLARE SUB GETTER ()
      DECLARE SUB NOTE (S)
      DECLARE SUB WARBLE ()
      DECLARE SUB ROMBERG (NT, CF, FLB, FUB, RI)
      DECLARE FUNCTION GAUSSIAN (Z)
      DECLARE SUB GND (AL, DV, PV)
      DECLARE SUB GND1 (AL, DV, PV)
      DECLARE FUNCTION FAC (N)
      DECLARE FUNCTION QND (AL)
      DECLARE FUNCTION TND (AL)
      DECLARE FUNCTION UND (AL)
      DECLARE FUNCTION T1 (JD, AL)
      DECLARE FUNCTION T2 (JD, AL)
      DECLARE FUNCTION T3 (JD, AL)
      DECLARE FUNCTION T4 (JD, AL)
      DECLARE SUB REFINERY1 (TL, UV, PV)
      DECLARE SUB REFINERY2 (TL, UV, PV)
      DECLARE SUB REFINERY3 (JD, TL, UV, PV)
      DECLARE SUB REFINERY4 (JD, TL, UV, PV)
      DECLARE SUB SAMEX (IW, M, X(), TL)
      DECLARE SUB LAGRANGIAN (N, XL(), FL(), XT, FT)
      DECLARE FUNCTION TAPPROX (AL, JD)
      DECLARE FUNCTION TPROB (T, JD)
      DECLARE FUNCTION TPROB1 (T, JD)
      DECLARE FUNCTION TPROB2 (T, JD)
```

```
      DECLARE FUNCTION PROD (K, MD)
' COMMON VARIABLES
      COMMON SHARED PI, HP, IA, SA, IV, SOU
' STATIC ARRAY DEFINITIONS
      DIM AL(25), XL(10), FL(10), IL(5), SB(30)
' DYNAMIC ARRAY DEFINITIONS
'     ( none )
' DEVICE ATTRIBUTIONS
      SCREEN 12: WINDOW (1, 1)-(640, 480)
' LOGICAL UNIT, TRANSPUT MODE AND DEVICE SETTINGS
      IV = 2: SXV = ".TTI"
      SP = "C:\QBASIC\QBFILES\"
      SOU = SP + "TTIMER" + SXV
'     SOU = "CON"
' FORMAT DEFINITIONS
'     ( none )
' NUMERICAL CONSTANT DEFINITIONS
      PI = 3.141592653589793#: HP = PI / 2
' STRING CONSTANT DEFINITIONS
'     ( none )
' TEXT VARIABLE DEFINITIONS
'     ( none )
'
' ** THE ALGORITHM **
'
      ALASSIGN NA, AL()
      JL = 5: JU = 30: JS = 5
      CAPTION ME, IL(), SB()
      TIMELOG ME, IL(), SB(), NA, AL(), JL, JU, JS
      CLOSE IV
      WARBLE
      END

      SUB ALASSIGN (NA, AL())
' A SUBROUTINE TO FILL THE ARRAY AL() WITH THE TWENTY VALUES OF INTEGRAL PROBABILITY:
' 0.25, 0.1, 0.05, 0.025, 0.01, 0.005, 0.0025, 0.001, 0.0005, 0.00025, 0.0001,
' 0.00005, 0.000025, 0.00001, 0.000005, 0.0000025, 0.000001, 0.0000005, 0.00000025,
' 0.0000001
'     ARGUMENTS:
'        NA     THE NUMBER OF INTEGRAL PROBABILITIES
'        AL()   THE ARRAY  OF INTEGRAL PROBABILITIES
'
      NA = 20
      AL(1) = .25: II = 1: IC = 1
      FOR I = 1 TO 19
         IF I = IC THEN
            AL(I + 1) = AL(I) / 2.5: II = II + 1: IC = 3 * II - 2
         ELSE
            AL(I + 1) = AL(I) / 2
         END IF
      NEXT I
      END SUB

      SUB CAPTION (ME, IL(), SB())
' A SUBROUTINE TO DEFINE THE EXPLANITORY CAPTION BLOCKS FOR THE AVAILABLE
' METHODS OF t-QUANTILE DETERMINATION
'     ARGUMENTS:
'        ME     THE METHOD SERIAL NUMBER
'        IL()   THE LINE FINISH NUMBER
'        SB()   THE ARRAY OF CAPTION LINES
'
      IL(0) = 0: IL(1) = 4: IL(2) = 9: IL(3) = 18: IL(4) = 25
      SB(1) = "   1   FIVE-FIGURE ACCURACY"
      SB(2) = "       HILL'S METHOD RESOURCES A NEGATIVE NORMAL DEVIATE DETERMINED BY"
      SB(3) = "       HASTINGS 26.2.23 REFINED BY AN INTERPOLATIVE ITERATION WITH"
      SB(4) = "       ORDINATES ERECTED BY ROMBERG INTEGRATION"
      SB(5) = "   2   FIVE-FIGURE ACCURACY"
      SB(6) = "       HILL'S METHOD RESOURCES A NEGATIVE NORMAL DEVIATE DETERMINED BY"
      SB(7) = "       HASTINGS 26.2.23 REFINED BY AN INTERPOLATIVE ITERATION WITH"
      SB(8) = "       ORDINATES ERECTED BY USE OF THE CONVERGENT SERIES FOR"
      SB(9) = "       GAUSSIAN PROBABILITY"
      SB(10) = "   3   TWELVE-FIGURE ACCURACY"
      SB(11) = "       HILL'S APPROXIMATION IS DETERMINED RESOURCING A NEGATIVE"
      SB(12) = "       NORMAL DEVIATE DETERMINED BY HASTINGS 26.2.23 REFINED BY AN"
      SB(13) = "       INTERPOLATIVE ITERATION WITH ORDINATES ERECTED BY USE OF THE"
      SB(14) = "       CONVERGENT SERIES FOR GAUSSIAN PROBABILITY."
      SB(15) = "       HILL'S APPROXIMATION IS THEN ITSELF REFINED AGAINST THE"
      SB(16) = "       TARGET PROBABILITY USING AN INTERPOLATIVE ITERATION WHOSE"
      SB(17) = "       ORDINATES ARE ERECTED USING THE FINITE TRIGONOMETRIC SERIES"
      SB(18) = "       FOR THE PROBABILITY ASSOCIATED WITH THE STUDENT'S t-QUANTILE"
      SB(19) = "   4   FIFTEEN-FIGURE ACCURACY"
```

```
       SB(20) = "       THE SERIES APPROXIMATOR FOR THE t-QUANTILE 26.7.5 IS APPLIED"
       SB(21) = "       TO DEVELOP NEAR THREE-FIGURE ACCURACY."
       SB(22) = "       THIS ESTIMATE IS THEN REFINED USING AN ITERPOLATIVE ITERATION"
       SB(23) = "       WHOSE ORDINATES ARE ERECTED USING THE FINITE TRIGONOMETRIC"
       SB(24) = "       SERIES FOR THE PROBABILITY ASSOCIATED WITH THE STUDENT'S"
       SB(25) = "       t-QUANTILE"
       END SUB

       FUNCTION FAC (N)
' A FUNCTION TO COMPUTE THE FACTORIAL OF N
'      ARGUMENTS:
'         N      THE NUMBER WHOSE FACTORIAL IS REQUIRED
       F = 1: FOR I = 1 TO N: F = F * I: NEXT I
       FAC = F
       END FUNCTION

       FUNCTION GAUSSIAN (Z)
' A FUNCTION TO SUPPLY THE GAUSSIAN PROBABILITY DENSITY FUNCTION
' OF THE NORMAL DEVIATE Z.
' ( NOTE: THE SCALING CONSTANT IS SUPPLIED INDEPENDENTLY AS CF )
'      ARGUMENT:
'         Z      THE NORMAL DEVIATE
'
       GAUSSIAN = EXP(-.5 * Z ^ 2)
       END FUNCTION

       SUB GETTER
' A SUBROUTINE TO ACCEPT A KEYSTROKE AS SA AND TO YIELD ITS ASCII CODE AS IA
' ( SA AND IA ARE COMMON SHARED )
       DO
          SA = INKEY$
       LOOP UNTIL SA <> ""
       IA = ASC(SA)
       END SUB

       SUB GND (AL, DV, PV)
' A SUBROUTINE TO APPROXIMATE A GAUSSIAN PROBABILITY AL FROM THE DEVIATE DV.
' THIS ROUTINE SUMMATES THE FIRST M NECESSARY AND SUFFICIENT TERMS OF A
' RAPIDLY CONVERGENT SERIES.
'      ARGUMENTS:
'         AL     THE GAUSSIAN PROBABILITY
'         DV     THE ( POSITIVE ) NORMAL DEVIATE
'         PV     THE REQUIRED MANTISSA PRECISION
' ( PI IS COMMON SHARED )
'
       M = INT(.8 + DV ^ 2.09414342149# + .921892385# * PV * DV ^ .5481692781800001#)
       I = -1: J = I: AL = 0#
       FOR N = 0 TO M: I = -I: J = J + 2: AL = AL + I * DV ^ J / (FAC(N) * J * 2 ^ N): NEXT N
       AL = .5# + (1 / SQR(2 * PI)) * AL
       AL = 1# - AL
       END SUB

       SUB GND1 (AL, DV, PV)
' A SUBROUTINE TO APPROXIMATE A GAUSSIAN PROBABILITY AL FROM THE DEVIATE DV.
' THIS ROUTINE SUMMATES THE FIRST M NECESSARY AND SUFFICIENT TERMS OF A
' RAPIDLY CONVERGENT SERIES.
'      ARGUMENTS:
'         AL     THE GAUSSIAN PROBABILITY
'         DV     THE ( POSITIVE ) NORMAL DEVIATE
'         PV     THE REQUIRED MANTISSA PRECISION
' ( PI IS COMMON SHARED )
'
       M = INT(.8 + DV ^ 2.09414342149# + .921892385# * PV * DV ^ .5481692781800001#)
       AL = DV
       I = 1: J = 1: FA = 1#: RN = 1#: D = DV: DW = DV * DV
       FOR N = 1 TO M: I = -I: J = J + 2: D = D * DW: FA = FA * N: RN = 2 * RN: AL = AL + I *
D / (J * FA * RN): NEXT N
       AL = .5# - (1 / SQR(2 * PI)) * AL
       END SUB

       SUB HEAD (ME)
' A SUBROUTINE TO PRINT THE OUTPUT HEADER FOR A METHOD TIMING
'      ARGUMENTS:
'         ME     THE METHOD SERIAL NUMBER
' ( IV IS COMMON SHARED )
'
       CLS
       IF SOU = "CON" THEN
            COLOR 3: PRINT "METHOD"; ME: PRINT "--------": PRINT
          ELSE
             PRINT #IV, "METHOD"; ME: PRINT #IV, "--------": PRINT #IV,
```

```
      END IF
      END SUB

      SUB LAGRANGIAN (N, XL(), FL(), XT, FT)
' A SUBROUTINE TO COMPUTE A LAGRANGIAN INTERPOLATE FT BEING A FUNCTIONAL VALUE
' AT ABSCISSOR XT FROM KNOWLEDGE OF THE N POLYNOMIAL CO-ORDINATES (XL(),FL())
'      ARGUMENTS:
'         N      THE NUMBER OF KNOWN CO-ORDINATES
'         XL()   THE ARRAY  OF KNOWN X CO-ORDINATES
'         FL()   THE ARRAY  OF KNOWN Y CO-ORDINATES
'         XT     THE ABSCISSAL VALUE AT WHICH FT IS TO BE ESTIMATED
'         FT     THE INTERPOLATED FUNCTION OF XT
'
      FT = 0#
         FOR I = 1 TO N
            A = 1#
               FOR J = 1 TO I - 1
                  A = A * (XT - XL(J)) / (XL(I) - XL(J))
               NEXT J
               FOR J = I + 1 TO N
                  A = A * (XT - XL(J)) / (XL(I) - XL(J))
               NEXT J
            FT = FT + FL(I) * A
         NEXT I
      END SUB

      SUB NOTE (S)
' A SUBROUTINE TO SOUND A NOTE UPON THE COMPUTER SPEAKER
'      ARGUMENT:
'         S     THE NOTE SPECIFIER STRING "IN$NL" e.g. "2B0506"
'                  I   THE OCTAVE NUMBER        ( 0-6 )
'                  N$  THE NOTE LETTER          ( ABCDEFG )
'                  N   THE NOTE NUMBER          ( 0-84 )
'                  L   THE LENGTH OF THE NOTE   ( 1-64 )
'
      I = VAL(MID$(S, 1, 1)): N$ = MID$(S, 2, 1)
      N = VAL(MID$(S, 3, 2)): L = VAL(MID$(S, 5, 2))
      PLAY "O" + STR$(I) + "N" + STR$(N) + "L" + STR$(L) + "X" + VARPTR$(N$)
      END SUB

      FUNCTION PROD (K, MD)
' A FUNCTION TO COMPUTE THE PRODUCT SERIES 2*I+MD BETWEEN 1 AND K
'      ARGUMENTS:
'         K      THE UPPER BOUND OF THE PRODUCT SERIES
'         MD     THE ADDITIVE MODIFIER
'
      P = 1#: FOR I = 1 TO K: P = P * (2 * I + MD): NEXT I
      PROD = P
      END FUNCTION

      FUNCTION QND (AL)
' A FUNCTION TO APPROXIMATE A NORMAL DEVIATE.
' THIS ROUTINE EMPLOYS A POLYNOMIAL APPROXIMATOR DUE TO HASTINGS
' TAKEN FROM 26.2.23 ON PAGE 933 OF "HANDBOOK OF MATHEMATICAL FUNCTIONS"
' BY ABRAMOWITZ AND STEGUN ( ISBN 0-486-61272-4 ).
'      ARGUMENT:
'         AL     THE INTEGRAL PROBABILITY
' ( PI IS COMMON SHARED )
'
      DIM C(6)
      C(1) = 2.515517: C(2) = .802853: C(3) = .010328
      C(4) = 1.432788: C(5) = .189269: C(6) = .001308
      U = SQR(LOG(1 / AL ^ 2))
      U1 = C(1) + C(2) * U + C(3) * U ^ 2
      U2 = 1 + C(4) * U + C(5) * U ^ 2 + C(6) * U ^ 3
      QND = U - U1 / U2
      END FUNCTION

      SUB REFINERY1 (TL, UV, PV)
' A SUBROUTINE TO IMPROVE AN APPROXIMATE NORMAL DEVIATE
'      ARGUMENTS:
'         PI     THE LUDOLPHINE CONSTANT
'         TL     THE REQUIRED ( PLUS OR MINUS ) TOLERANCE
'         UV     THE ESTIMATED NORMAL DEVIATE
'         PV     THE PROBABILITY ATTACHING TO THE TRUE DEVIATE
'
      DIM XL(6), FL(6)
      PV = .5# - PV
      M = 3: N = 7: TM = .00045#: CF = 1 / SQR(2 * PI)
      DO
         XL(1) = UV - TM: XL(2) = UV: XL(3) = UV + TM
```

```
          FOR I = 1 TO M
             ROMBERG N, CF, 0, XL(I), FL(I)
          NEXT I
          UW = UV
          SAMEX IW, M, FL(), .000000000001#
          IF IW = 0 THEN LAGRANGIAN M, FL(), XL(), PV, UV
          TM = ABS(UV - UW)
       LOOP UNTIL TM < TL OR IW = 1
       PV = .5# - PV
       END SUB

       SUB REFINERY2 (TL, UV, PV)
' A SUBROUTINE TO IMPROVE AN APPROXIMATE NORMAL DEVIATE.
' THIS SUBROUTINE UTILISES THE RAPIDLY-CONVERGENT SERIES
' OF THE GND1 ROUTINE
'      ARGUMENTS:
'         TL     THE REQUIRED ( PLUS OR MINUS ) TOLERANCE
'         UV     THE ESTIMATED NORMAL DEVIATE
'         PV     THE PROBABILITY ATTACHING TO THE TRUE DEVIATE
'
       DIM XL(6), FL(6)
       M = 3: TM = .00045#
       DO
          XL(1) = UV - TM: XL(2) = UV: XL(3) = UV + TM
          FOR I = 1 TO M
             GND1 FL(I), XL(I), 12#
          NEXT I
          UW = UV
          SAMEX IW, M, FL(), .000000000001#
          IF IW = 0 THEN LAGRANGIAN M, FL(), XL(), PV, UV
          TM = ABS(UV - UW)
       LOOP UNTIL TM < TL OR IW = 1
       PV = .5# - PV
       END SUB

       SUB REFINERY3 (JD, TL, UV, PV)
' A SUBROUTINE TO IMPROVE AN APPROXIMATE STUDENT'S t QUANTILE.
' THE PROBABILITIES ASSOCIATED WITH t QUANTILES ARE COMPUTED BY
' THE FINITE SERIES TECHNIQUE OF FUNCTION TPROB2
'      ARGUMENTS:
'         JD     THE DEGREES OF FREEDOM
'         TL     THE REQUIRED ( PLUS OR MINUS ) TOLERANCE
'         UV     THE ESTIMATED t QUANTILE
'         PV     THE PROBABILITY ATTACHING TO THE TRUE DEVIATE
' ( PI IS COMMON SHARED )
'
       DIM XL(6), FL(6)
       M = 3: TM = .00005#
       DO
          XL(1) = UV - TM: XL(2) = UV: XL(3) = UV + TM
          FOR I = 1 TO M
             FL(I) = TPROB2(XL(I), JD)
          NEXT I
          UW = UV
          SAMEX IW, M, FL(), 9.9999999999999999D-12
          IF IW = 0 THEN LAGRANGIAN M, FL(), XL(), 1 - PV, UV
          TM = ABS(UV - UW)
       LOOP UNTIL TM < TL OR IW = 1
       END SUB

       SUB REFINERY4 (JD, TL, UV, PV)
' A SUBROUTINE TO IMPROVE AN APPROXIMATE STUDENT'S t QUANTILE.
' THE PROBABILITIES ASSOCIATED WITH t QUANTILES ARE COMPUTED BY
' THE FINITE SERIES TECHNIQUE OF FUNCTION TPROB2
'      ARGUMENTS:
'         JD     THE DEGREES OF FREEDOM
'         TL     THE REQUIRED ( PLUS OR MINUS ) TOLERANCE
'         UV     THE ESTIMATED t QUANTILE
'         PV     THE PROBABILITY ATTACHING TO THE TRUE DEVIATE
' ( PI IS COMMON SHARED )
'
       DIM XL(6), FL(6)
       M = 3: TM = .001#
       DO
          XL(1) = UV - TM: XL(2) = UV: XL(3) = UV + TM
          FOR I = 1 TO M
             FL(I) = TPROB2(XL(I), JD)
          NEXT I
          UW = UV
          SAMEX IW, M, FL(), 9.9999999999999999D-12
          IF IW = 0 THEN LAGRANGIAN M, FL(), XL(), 1 - PV, UV
```

```
         TM = ABS(UV - UW)
      LOOP UNTIL TM < TL OR IW = 1
      END SUB

      SUB ROMBERG (NT, CF, FLB, FUB, RI)
' A SUBROUTINE TO PERFORM A ROMBERG INTEGRATION OF FUNCTION XF()
' BETWEEN FLB AND FUB.
' CF IS THE INTEGRAL SCALING CONSTANT ( UNITY IF ABSENT ).
' ( NOTE: SEGMENT ROMBERG IS CURRENTLY CONFIGURED FOR THE GAUSSIAN PROBABILITY INTEGRAL )
'      ARGUMENTS:
'         NT     THE NUMBER OF TRAPEZOIDAL INTEGRATIONS TO BE PERFORMED ( NT < 10 )
'         CF     THE INTEGRAL SCALING CONSTANT
'         FLB    THE INTEGRAL LOWER BOUND
'         FUB    THE INTEGRAL UPPER BOUND
'         RI     THE ROMBERG INTEGRAL
'
      DIM WA(257), WB(8, 9)
      NS = NT - 1: MI = 2 ^ NS: MJ = MI + 1: DS = (FUB - FLB) / MI
      WA(1) = GAUSSIAN(FLB): WA(MJ) = GAUSSIAN(FUB)
      FOR I = 2 TO MI: WA(I) = 2 * GAUSSIAN(FLB + (I - 1) * DS): NEXT I
      FOR J = 0 TO NS
         II = 2 ^ J: TT = 0!: FOR I = 1 TO MJ STEP II: TT = TT + WA(I): NEXT I
         WB(0, NT - J) = .5# * II * DS * TT
      NEXT J
      FOR J = 1 TO NS
         JJ = 2 ^ (2 * J)
         FOR I = J + 1 TO NT: WB(J, I) = (JJ * WB(J - 1, I) - WB(J - 1, I - 1)) / (JJ - 1):
NEXT I
      NEXT J
      RI = CF * WB(NS, NT)
      END SUB

      SUB SAMEX (IW, M, X(), TL)
' A SUBROUTINE TO WARN OF EQUIVALENCY BETWEEN ANY ARRAYED X()
'      ARGUMENTS:
'         IW     THE EQUIVALENCY INDICATOR SWITCH
'                   0   NO EQUIVALENCY DETECTED
'                   1   EQUIVALENCY DETECTED
'         M      THE NUMBER OF VALUES X()
'         X()    THE ARRAY OF VALUES FOR INTERCOMPARISON
'         TL     THE TOLERANCE
'
      IW = 0
      FOR I = 1 TO M
         FOR J = 1 TO I - 1
            IF X(I) < X(J) + TL AND X(I) > X(J) - TL THEN IW = 1
         NEXT J
         FOR J = I + 1 TO M
            IF X(I) < X(J) + TL AND X(I) > X(J) - TL THEN IW = 1
         NEXT J
      NEXT I
      END SUB

      FUNCTION T1 (JD, AL)
' A FUNCTION TO SUPPLY STUDENT'S t.
' THIS FUNCTION IS ADAPTED FROM BY JAMES R WARREN FROM
' THE ORIGINAL ALGOL SOURCE OF SEGMENT STUDENT'S t-QUANTILE
' BY GW HILL ( CACM V13 N10 PAGE 620 OCTOBER 1970 )
'      ARGUMENTS:
'         JD     THE DEGREES OF FREEDOM
'         AL     THE INTEGRAL PROBABILITY
' ( HP IS COMMON SHARED )
'
      AL1 = 2 * AL
      SELECT CASE JD
         CASE 1
            P = AL1 * HP: T1 = COS(P) / SIN(P)
         CASE 2
            T1 = SQR(2 / (AL1 * (2 - AL1)) - 2)
         CASE ELSE
            A = 1 / (JD - .5)
            B = 48 / A ^ 2
            C = ((20700 * A / B - 98) * A - 16) * A + 96.36
            D = ((94.5 / (B + C) - 3) / B + 1) * SQR(A * HP) * JD
            X = D * AL1: Y = X ^ (2 / JD)
            IF Y > .05 + A THEN
                  X = TND(.5 * AL1): Y = X ^ 2
                  IF JD < 5 THEN C = C + .3 * (JD - 4.5) * (X + .6)
                  C = (((.05 * D * X - 5) * X - 7) * X - 2) * X + B + C
                  Y = (((((.4 * Y + 6.3) * Y + 36) * Y + 94.5) / C - Y - 3) / B + 1) * X
                  Y = A * Y ^ 2
```

```
                     IF Y > .002 THEN
                           Y = EXP(Y) - 1
                        ELSE
                           Y = .5 * Y ^ 2 + Y
                     END IF
                ELSE
                     Y = ((1 / (((JD + 6) / (JD * Y) - .089 * D - .822) * (JD + 2) * 3) + .5 /
(JD + 4)) * Y - 1) * (JD + 1) / (JD + 2) + 1 / Y
             END IF
             T1 = SQR(JD * Y)
       END SELECT
       END FUNCTION

       FUNCTION T2 (JD, AL)
' A FUNCTION TO SUPPLY STUDENT'S t.
' THIS FUNCTION IS ADAPTED FROM BY JAMES R WARREN FROM
' THE ORIGINAL ALGOL SOURCE OF SEGMENT STUDENT'S t-QUANTILE
' BY GW HILL ( CACM V13 N10 PAGE 620 OCTOBER 1970 )
'      ARGUMENTS:
'         JD     THE DEGREES OF FREEDOM
'         AL     THE INTEGRAL PROBABILITY
' ( HP IS COMMON SHARED )
'
       AL1 = 2 * AL
       SELECT CASE JD
          CASE 1
             P = AL1 * HP: T2 = COS(P) / SIN(P)
          CASE 2
             T2 = SQR(2 / (AL1 * (2 - AL1)) - 2)
          CASE ELSE
             A = 1 / (JD - .5)
             B = 48 / A ^ 2
             C = ((20700 * A / B - 98) * A - 16) * A + 96.36
             D = ((94.5 / (B + C) - 3) / B + 1) * SQR(A * HP) * JD
             X = D * AL1: Y = X ^ (2 / JD)
             IF Y > .05 + A THEN
                   X = UND(.5 * AL1): Y = X ^ 2
                   IF JD < 5 THEN C = C + .3 * (JD - 4.5) * (X + .6)
                   C = (((.05 * D * X - 5) * X - 7) * X - 2) * X + B + C
                   Y = (((((.4 * Y + 6.3) * Y + 36) * Y + 94.5) / C - Y - 3) / B + 1) * X
                   Y = A * Y ^ 2
                      IF Y > .002 THEN
                            Y = EXP(Y) - 1
                         ELSE
                            Y = .5 * Y ^ 2 + Y
                      END IF
                ELSE
                   Y = ((1 / (((JD + 6) / (JD * Y) - .089 * D - .822) * (JD + 2) * 3) + .5 /
(JD + 4)) * Y - 1) * (JD + 1) / (JD + 2) + 1 / Y
             END IF
             T2 = SQR(JD * Y)
       END SELECT
       END FUNCTION

       FUNCTION T3 (JD, AL)
' A FUNCTION TO SUPPLY STUDENT'S t.
' THIS FUNCTION IS ADAPTED FROM BY JAMES R WARREN FROM
' THE ORIGINAL ALGOL SOURCE OF SEGMENT STUDENT'S t-QUANTILE
' BY GW HILL ( CACM V13 N10 PAGE 620 OCTOBER 1970 )
'      ARGUMENTS:
'         JD     THE DEGREES OF FREEDOM
'         AL     THE INTEGRAL PROBABILITY
' ( HP IS COMMON SHARED )
'
       AL1 = 2 * AL
       SELECT CASE JD
          CASE 1
             P = AL1 * HP: T3 = COS(P) / SIN(P)
          CASE 2
             T3 = SQR(2 / (AL1 * (2 - AL1)) - 2)
          CASE ELSE
             A = 1 / (JD - .5)
             B = 48 / A ^ 2
             C = ((20700 * A / B - 98) * A - 16) * A + 96.36
             D = ((94.5 / (B + C) - 3) / B + 1) * SQR(A * HP) * JD
             X = D * AL1: Y = X ^ (2 / JD)
             IF Y > .05 + A THEN
                   X = UND(.5 * AL1): Y = X ^ 2
                   IF JD < 5 THEN C = C + .3 * (JD - 4.5) * (X + .6)
                   C = (((.05 * D * X - 5) * X - 7) * X - 2) * X + B + C
                   Y = (((((.4 * Y + 6.3) * Y + 36) * Y + 94.5) / C - Y - 3) / B + 1) * X
```

```
                  Y = A * Y ^ 2
                     IF Y > .002 THEN
                           Y = EXP(Y) - 1
                        ELSE
                           Y = .5 * Y ^ 2 + Y
                     END IF
               ELSE
                  Y = ((1 / (((JD + 6) / (JD * Y) - .089 * D - .822) * (JD + 2) * 3) + .5 /
(JD + 4)) * Y - 1) * (JD + 1) / (JD + 2) + 1 / Y
            END IF
            T = SQR(JD * Y)
            REFINERY3 JD, .00000001#, T, AL
            T3 = T
      END SELECT
      END FUNCTION

      FUNCTION T4 (JD, AL)
' A FUNCTION TO SUPPLY STUDENT'S t.
' THIS FUNCTION REFINES A POLYNOMIAL APPROXIMATION USING
' AN ITERATIVE INTERPOLATION
'     ARGUMENTS:
'        JD     THE DEGREES OF FREEDOM
'        AL     THE INTEGRAL PROBABILITY
' ( HP IS COMMON SHARED )
'
      AL1 = 2 * AL
      SELECT CASE JD
         CASE 1
            P = AL1 * HP: T4 = COS(P) / SIN(P)
         CASE 2
            T4 = SQR(2 / (AL1 * (2 - AL1)) - 2)
         CASE ELSE
            T = TAPPROX(AL, JD)
            REFINERY4 JD, .00000001#, T, AL
            T4 = T
      END SELECT
      END FUNCTION

      SUB TAIL (ME, IL(), SB(), NP, TR)
' A SUBROUTINE TO PRINT THE RESULTS OF A METHOD TIMING
'     ARGUMENTS:
'        ME     THE METHOD SERIAL NUMBER
'        IL()   THE ARRAY OF LINE FINISH NUMBERS
'        SB()   THE ARRAY OF CAPTION LINES
'        NP     THE NUMBER OF QUANTILES COMPUTED
'        TR     THE TIME TO COMPUTE NP QUANTILES ( SECONDS )
' ( IV AND SOU ARE COMMON SHARED )
'
      IF SOU = "CON" THEN
            COLOR 15
            FOR K = IL(ME - 1) + 1 TO IL(ME): PRINT SB(K): NEXT K
            PRINT
            PRINT "THIS TIME TEST IS FOR INTERPRETED QBASIC CODE"
            PRINT : COLOR 2
            PRINT "EXECUTION TIME FOR "; NP; " t-QUANTILES = ";
            COLOR 14: PRINT TR; : COLOR 2
            PRINT " SECONDS"
            WARBLE
            GETTER
         ELSE
            FOR K = IL(ME - 1) + 1 TO IL(ME): PRINT #IV, SB(K): NEXT K
            PRINT #IV,
            PRINT #IV, "THIS TIME TEST IS FOR INTERPRETED QBASIC CODE"
            PRINT #IV,
            PRINT #IV, "EXECUTION TIME FOR "; NP; " t-QUANTILES = "; TR; " SECONDS"
            PRINT #IV, : PRINT #IV, : PRINT #IV,
      END IF
      END SUB

      FUNCTION TAPPROX (AL, JD)
' A FUNCTION TO APPROXIMATE THE STUDENT'S t-QUANTILE.
' THIS ROUTINE EMPLOYS A SERIES APPROXIMATOR TAKEN FROM 26.7.5 ON PAGE 949
' OF "HANDBOOK OF MATHEMATICAL FUNCTIONS" BY ABRAMOWITZ AND STEGUN
' ( ISBN 0-486-61272-4 )
'     ARGUMENTS:
'        AL     THE INTEGRAL PROBABILITY
'        JD     THE DEGREES OF FREEDOM
'
      X = QND(AL)
      G1 = .25 * (X ^ 3 + X)
      G2 = .01041666666# * (5 * X ^ 5 + 16 * X ^ 3 + 3 * X)
```

```
      G3 = .00260416666# * (3 * X ^ 7 + 19 * X ^ 5 + 17 * X ^ 3 - 15 * X)
      G4 = .00001085069# * (79 * X ^ 9 + 776 * X ^ 7 + 1482 * X ^ 5 - 1920 * X ^ 3 - 945 * X)
      TAPPROX = X + G1 / JD + G2 / JD ^ 2 + G3 / JD ^ 3 + G4 / JD ^ 4
      END FUNCTION

      SUB TIMELOG (ME, IL(), SB(), NA, AL(), JL, JU, JS)
' A SUBROUTINE TO TIME THE EXECUTION AND DISPLAY THE RESULT FOR THE ME
' AVAILABLE t-QUANTILE DETERMINATION METHODS
'      ARGUMENTS:
'         ME     THE METHOD SERIAL NUMBER
'         IL()   THE ARRAY OF LINE FINISH NUMBERS
'         SB()   THE ARRAY OF CAPTION LINES
'         NA     THE NUMBER OF INTEGRAL PROBABILITIES
'         AL()   THE ARRAY  OF INTEGRAL PROBABILITIES
'         JL     THE START  DEGREES OF FREEDOM
'         JU     THE FINISH DEGREES OF FREEDOM
'         JS     THE DEGREES OF FREEDOM STEP INTERVAL
' ( IV AND SOU ARE COMMON SHARED )
'
      NP = NA * ((JU - JL) / JS + 1)
      OPEN "O", IV, SOU
' BLOCK FOR METHOD 1
      HEAD 1
      TI = TIMER
      FOR J = JL TO JU STEP JS: FOR I = 1 TO NA: TT = T1(J, AL(I)): NEXT I: NEXT J
      TJ = TIMER
      TR = TJ - TI
      TAIL 1, IL(), SB(), NP, TR
' BLOCK FOR METHOD 2
      HEAD 2
      TI = TIMER
      FOR J = JL TO JU STEP JS: FOR I = 1 TO NA: TT = T2(J, AL(I)): NEXT I: NEXT J
      TJ = TIMER
      TR = TJ - TI
      TAIL 2, IL(), SB(), NP, TR
' BLOCK FOR METHOD 3
      HEAD 3
      TI = TIMER
      FOR J = JL TO JU STEP JS: FOR I = 1 TO NA: TT = T3(J, AL(I)): NEXT I: NEXT J
      TJ = TIMER
      TR = TJ - TI
      TAIL 3, IL(), SB(), NP, TR
' BLOCK FOR METHOD 4
      HEAD 4
      TI = TIMER
      FOR J = JL TO JU STEP JS: FOR I = 1 TO NA: TT = T4(J, AL(I)): NEXT I: NEXT J
      TJ = TIMER
      TR = TJ - TI
      TAIL 4, IL(), SB(), NP, TR
' MODULE TERMINATION
      END SUB

      FUNCTION TND (AL)
' A FUNCTION TO APPROXIMATE A NEGATIVE NORMAL DEVIATE.
' THIS ROUTINE EMPLOYS A POLYNOMIAL APPROXIMATOR DUE TO HASTINGS
' TAKEN FROM 26.2.23 ON PAGE 933 OF "HANDBOOK OF MATHEMATICAL FUNCTIONS"
' BY ABRAMOWITZ AND STEGUN ( ISBN 0-486-61272-4 ).
' SEGMENT REFINERY1 DUE TO JAMES R WARREN IS AN ITERATED COUPLE OF
' ROMBERG INTEGRATION AND FOUR-POINT LAGRANGIAN INTERPOLATION
' WHICH TAKES THE APPROXIMATION TO TWELVE-FIGURE ACCURACY
'      ARGUMENT:
'         AL     THE INTEGRAL PROBABILITY
' ( PI IS COMMON SHARED )
'
      UV = QND(AL)
      REFINERY1 .00000001#, UV, AL
      TND = -UV
      END FUNCTION

      FUNCTION TPROB (T, JD)
' A SUBROUTINE TO DETERMINE THE PROBABILITY ASSOCIATED WITH STUDENT'S t
' BY FINITE SERIES SUMMATION.
' DEGREES OF FREEDOM MUST BE THREE OR MORE.
'      ARGUMENTS:
'         T      THE STUDENT'S t QUANTILE
'         JD     THE DEGREES OF FREEDOM
' ( PI IS COMMON SHARED )
'
      N = INT(JD / 2 - 1): KS = 1 - 2 * (JD / 2 - INT(JD / 2))
      IM = -KS: JM = 1 - KS: KM = 1 - KS
      G = 0#: TH = ATN(T / SQR(JD))
```

```
      FOR K = KS TO N: G = G + COS(TH) ^ (2 * K + KM) * PROD(K, IM) / PROD(K, JM): NEXT K
      SELECT CASE KS
         CASE 0
            P = (2 / PI) * (TH + SIN(TH) * G)
         CASE 1
            P = SIN(TH) * (1 + G)
      END SELECT
      TPROB = (P + 1) / 2
      END FUNCTION

      FUNCTION TPROB1 (T, JD)
' A SUBROUTINE TO DETERMINE THE PROBABILITY ASSOCIATED WITH STUDENT'S t
' BY FINITE SERIES SUMMATION.
' DEGREES OF FREEDOM MUST BE THREE OR MORE.
'      ARGUMENTS:
'         T      THE STUDENT'S t QUANTILE
'         JD     THE DEGREES OF FREEDOM
' ( PI IS COMMON SHARED )
'
      N = INT(JD / 2 - 1): KS = 1 - 2 * (JD / 2 - INT(JD / 2))
      G = 0#: TH = ATN(T / SQR(JD))
      RD = 1#: RN = 1#: RC = COS(TH): R2 = RC * RC
      I = 1 - KS: J = -KS
      IF KS = 0 THEN RT = RC ELSE RT = 1#
      FOR K = 1 TO N
         I = I + 2: J = J + 2: RD = RD * J: RN = RN * I: RT = RT * R2: G = G + RT * RD / RN
      NEXT K
      SELECT CASE KS
         CASE 0
            P = (2 / PI) * (TH + SIN(TH) * (COS(TH) + G))
         CASE 1
            P = SIN(TH) * (1 + G)
      END SELECT
      TPROB1 = (P + 1) / 2
      END FUNCTION

      FUNCTION TPROB2 (T, JD)
' A SUBROUTINE TO DETERMINE THE PROBABILITY ASSOCIATED WITH STUDENT'S t
' BY FINITE SERIES SUMMATION.
' DEGREES OF FREEDOM MUST BE THREE OR MORE.
'      ARGUMENTS:
'         T      THE STUDENT'S t QUANTILE
'         JD     THE DEGREES OF FREEDOM
' ( PI IS COMMON SHARED )
'
      N = INT(JD / 2 - 1): KS = 1 - 2 * (JD / 2 - INT(JD / 2))
      G = 0#: TH = ATN(T / SQR(JD))
      RD = 1#: RP = 1#: RC = COS(TH): R2 = RC * RC
      I = 1 - KS: J = -KS
      IF KS = 0 THEN RT = RC ELSE RT = 1#
      K = 0
      DO
         K = K + 1: I = I + 2: J = J + 2: RD = J / I: RP = RP * RD: RT = RT * R2: G = G + RP
* RT
      LOOP UNTIL K = N
      SELECT CASE KS
         CASE 0
            P = (2 / PI) * (TH + SIN(TH) * (COS(TH) + G))
         CASE 1
            P = SIN(TH) * (1 + G)
      END SELECT
      TPROB2 = (P + 1) / 2
      END FUNCTION

      FUNCTION UND (AL)
' A FUNCTION TO APPROXIMATE A NEGATIVE NORMAL DEVIATE.
' THIS ROUTINE EMPLOYS A POLYNOMIAL APPROXIMATOR DUE TO HASTINGS
' TAKEN FROM 26.2.23 ON PAGE 933 OF "HANDBOOK OF MATHEMATICAL FUNCTIONS"
' BY ABRAMOWITZ AND STEGUN ( ISBN 0-486-61272-4 ).
' SEGMENT REFINERY2 DUE TO JAMES R WARREN IS AN ITERATED COUPLE OF
' SERIES SUMMATION AND FOUR-POINT LAGRANGIAN INTERPOLATION
' WHICH TAKES THE APPROXIMATION TO TWELVE-FIGURE ACCURACY
'      ARGUMENT:
'         AL     THE INTEGRAL PROBABILITY
' ( PI IS COMMON SHARED )
'
      UV = QND(AL)
      REFINERY2 .0000000000001#, UV, AL
      UND = -UV
      END FUNCTION
```

```
       SUB WARBLE
' A SUBROUTINE TO SOUND A SUCCESSION OF FOUR NOTE CALLS
' ON THE COMPUTER SPEAKER
       NOTE "3C0506": NOTE "4D0509": NOTE "5E0512": NOTE "6F0515"
       END SUB
```

Appendix Two

Program New BELEM.BAS

```
'      PROGRAM BELEM.BAS
'      A PROGRAM TO COMPUTE THE DESCRIPTIVE STATISTICS OF THE MAGNITUDES
'      IN CONTEMPORANEOUS CLUSTERED OR CLASSIFIED ENTITIES LOCATED IN A
'      PLANAR CARTESIAN GRID SYSTEM.
'      THE PROGRAM ALSO COMPUTES THE SPATIAL MASS CENTROIDS AND DYNAMIC RADII
'      OF SUCCESSIVE CLUSTERS AND ERECTS PLOTTABLE BELEMNOIDS WHICH SUMMARISE
'      MAGNITUDE MIGRATION DIRECTION AND MAGNITUDE MEAN AND STANDARD DEVIATION
'      FOR EACH CLUSTER.
'      CONFIDENCE LIMITS ARE QUOTED FOR THE MEAN OF THE ENTITIES IN EACH CLUSTER.
'      THE PROGRAM IS CONFIGURED FOR BATCH WORKING AFTER INTERACTIVE FILENAME
'      PROMPTING.
'      CONSULT THE PAPER "A DESCRIPTION OF PROGRAM BELEM.BAS"
'      FOR DATA FORMATTING DETAILS.
'      THESE FILES BEAR THE EXTENSION SHOWN:-
'
'            THE DATA       INPUT  FILE      .CSV
'            THE STATISTICS OUTPUT FILE      .BEL
'
'      WRITTEN BY:-
'
'         JAMES R WARREN BSc MSc PhD PGCE
'         "SOUTHGATE"
'         31 VICTORIA AVENUE
'         BLOXWICH
'         WS3 3HS
'         UNITED KINGDOM
'
'         8 NOVEMBER 1996
'
'      THIS PROGRAM IS WRITTEN IN MICROSOFT QBASIC
'
' VARIABLE TYPE DEFAULTS
       DEFDBL A-H, O-R, T-Z
       DEFSTR S
       DEFINT I-K, M-N
       DEFLNG L
' SEGMENT DECLARATIONS
       DECLARE SUB CENTROID (J, N, P(), X(), Y(), CX, CY, DS)
       DECLARE SUB CONFIDENCE (J, N, D, AL, AM, VS, NS, DL, DU)
       DECLARE SUB DATAIN (SW, CB, CC, EM, AO, D, AL, K, NM, SK(), IP(), SN(), AE(), AN(),
PP(), V())
       DECLARE SUB DATAOUT (K, CB, AX, SW, SJ(), SK(), V())
       DECLARE SUB DIATESSARON (J, CK, PI, AM, VP, TX, CX, CY, V())
       DECLARE SUB FINALE ()
       DECLARE SUB GETTER ()
       DECLARE SUB GF (NDC, N, XM(), XF(), AM, VP, VS, VQ, VK)
       DECLARE SUB GND1 (AL, DV, PV)
       DECLARE SUB LAGRANGIAN (N, XL(), FL(), XT, FT)
       DECLARE SUB MOMENTAL (N, P(), AM, VP, VS, VQ, VR)
       DECLARE SUB NOTE (S)
       DECLARE SUB PICK (KT, I, L, R, S, IY, IX, MK, SF)
       DECLARE SUB PLACE (KT, I, L, R, S, IY, IX, IU, MK, JF, SF, IB)
       DECLARE FUNCTION QND (AL)
       DECLARE SUB REFINERY2 (TL, UV, PV)
       DECLARE SUB REFINERY3 (JD, TL, UV, PV)
       DECLARE SUB REFINERY4 (JD, TL, UV, PV)
       DECLARE SUB RESULTHETA (K, PI, TA())
       DECLARE SUB ROTATE (TH, XO, YO, XI, YI, XR, YR)
       DECLARE SUB SAMEX (IW, M, X(), TL)
       DECLARE SUB STANDARD (CB, CC, PI, PH, K, CK, DR(), AX, EM, AO, V())
       DECLARE FUNCTION TAPPROX (AL, JD)
       DECLARE SUB THETA (J, PI, TA(), V())
       DECLARE FUNCTION THYBRID (JD, AL)
       DECLARE FUNCTION TPROB2 (T, JD)
       DECLARE FUNCTION UND (AL)
' COMMON VARIABLES
       COMMON SHARED IA, SA
       COMMON SHARED PI, HP
       COMMON SHARED SC, SM, SCR
       COMMON SHARED IU, IV, SP, SF, SXV, SXW
       COMMON SHARED SAP, SB, SE, SI, SL, SLF, SMP, SN, SO, SRF, ST, SU, SV, SX, SY
' STATIC ARRAY DEFINITIONS
'      ( none )
' DYNAMIC ARRAY DEFINITIONS
'      CURRENT NUMBER OF CLUSTERS                          NCL=11
'      CURRENT NUMBER OF DATA PER CLUSTER                  NDC=100
'      CURRENT NUMBER OF OUTPUT STATISTICS PER CLUSTER     NOS=21
       NCL = 11: NDC = 100: NOS = 21
       REDIM SK(NCL), DR(NCL)
       REDIM TA(NCL)
       REDIM SN(NDC), AE(NDC), AN(NDC)
```

```
       REDIM P(NDC), X(NDC), Y(NDC)
       REDIM V(NDC, NOS)
       REDIM IP(NCL, NDC), PP(NCL, NDC)
       REDIM SJ(NOS)
       REDIM NR(NCL), CL(NCL), CU(NCL)
' DEVICE ATTRIBUTIONS
       SCREEN 12: WINDOW (1, 1)-(640, 480)
' LOGICAL UNIT, EXTENSION AND PATHNAME SETTINGS
       IU = 1: IV = 2
       SXV = ".CSV": SXW = ".BEL"
       SP = "C:\QBASIC\QBFILES\"
' FORMAT DEFINITIONS
'      ( none )
' NUMERICAL CONSTANT DEFINITIONS
       PH = 1.6180339887495#
       PI = 3.141592653589793#
       HP = PI / 2#
' STRING CONSTANT DEFINITIONS
       SC = ":": SM = ",": SCR = CHR$(13) + CHR$(10)
' TEXT VARIABLE DEFINITIONS
       SAP = "APEX       ": SB = "BELEMNOID ": SE = "CENTROID ": SI = "ITEMS ": SL = "CLUSTER
"
       SLF = "LEFT FOOT  ": SMP = "SAMPLE ": SN = "NAME ": SO = "LOWER ": SRF = "RIGHT FOOT "
       ST = "CONFIDENCE LIMIT": SU = "UPPER ": SV = "VALUES ": SX = "X": SY = "Y"
'
' ** THE ALGORITHM **
'
       SZ = "ENTER THE FILENAME ( WITHOUT EXTENSION ):"
       PLACE 4, 0, 0, 0, SZ, 15, 16, 1, 2, 1, SF, 0
       PICK 4, 0, 0, 0, SF, 15, 59, 3, SA8
       CLS
       PLACE 4, 0, 0, 0, "PLEASE WAIT", 15, 36, 1, 14, 1, SF, 0
       DATAIN SW, CB, CC, EM, AO, D, AL, K, NM, SK(), IP(), SN(), AE(), AN(), PP(), V()
       FOR J = 1 TO K
          FOR I = 1 TO V(J, 2)
             P(I) = PP(J, IP(J, I)): X(I) = AE(IP(J, I)): Y(I) = AN(IP(J, I))
          NEXT I
          CENTROID J, FIX(V(J, 2)), P(), X(), Y(), V(J, 3), V(J, 4), V(J, 5)
          DR(J) = V(J, 5)
       NEXT J
       FOR J = 1 TO K - 1
          THETA J, PI, TA(), V()
       NEXT J
       RESULTHETA K, PI, TA()
       AX = 0#
       FOR J = 1 TO K
          FOR I = 1 TO V(J, 2): P(I) = PP(J, IP(J, I)): NEXT I
          MOMENTAL FIX(V(J, 2)), P(), V(J, 6), V(J, 7), V(J, 8), V(J, 9), V(J, 10)
          CONFIDENCE J, FIX(V(J, 2)), D, AL, V(J, 6), V(J, 8), NS, V(J, 12), V(J, 13)
          V(J, 11) = NS: AX = AX + V(J, 6)
       NEXT J
       AX = AX / K
       STANDARD CB, CC, PI, PH, K, CK, DR(), AX, EM, AO, V()
       FOR J = 1 TO K
          DIATESSARON J, CK, PI, V(J, 6), V(J, 7), TA(J), V(J, 3), V(J, 4), V()
       NEXT J
       DATAOUT K, CB, AX, SW, SJ(), SK(), V()
       FINALE
       END

       SUB CENTROID (J, N, P(), X(), Y(), CX, CY, DS)
' A SUBROUTINE TO COMPUTE THE MASS CENTROID AND THE DYNAMIC RADIUS OF
' A SET OF VALUES IN A CARTESIAN CO-ORDINATE SYSTEM.
' THE CENTROID FORMULAE ARE TAKEN FROM PAGE 116 OF
' "GEOGRAPHICAL DATA:SOURCES, PRESENTATION AND ANALYSIS" BY HUGH MATTHEWS
' AND IAN FOSTER ( ISBN 0-19-913328-X ).
' THE FORMULA FOR DYNAMIC RADIUS IS TAKEN FROM PAGE 313 OF "LOCATIONAL METHODS"
' BY HAGGETT, CLIFF AND FREY ( ISBN 0-7131-5956-1 )
'      ARGUMENTS:
'         J      THE CLUSTER NUMBER
'         N      THE NUMBER OF CLUSTER MAGNITUDES
'         P()    THE ARRAY OF VALUE MAGNITUDES
'         X()    THE ARRAY OF MAGNITUDE X CO-ORDINATES
'         Y()    THE ARRAY OF MAGNITUDE Y CO-ORDINATES
'         CX     THE ARRAY OF CENTROID  X CO-ORDINATES
'         CY     THE ARRAY OF CENTROID  Y CO-ORDINATES
'         DS     THE ARRAY OF DYNAMIC RADII
'
       PS = 0#: XS = 0#: YS = 0#: DS = 0#
       FOR I = 1 TO N
          PS = PS + P(I): XS = XS + P(I) * X(I): YS = YS + P(I) * Y(I)
```

```
      NEXT I
      CX = XS / PS: CY = YS / PS
      FOR I = 1 TO N
         DS = DS + P(I) * ((X(I) - CX) ^ 2 + (Y(I) - CY) ^ 2)
      NEXT I
      DS = SQR(DS / PS)
      END SUB

      SUB CONFIDENCE (J, N, D, AL, AM, VS, NS, DL, DU)
' A SUBROUTINE TO COMPUTE FOR A DATA CLUSTER THE NORMATIVE SAMPLE SIZE NR(J) AND
' THE UPPER AND LOWER CONFIDENCE LIMITS CL(J) AND CU(J)
'      ARGUMENTS:
'         J      THE CLUSTER NUMBER
'         N      THE NUMBER OF CLUSTER MAGNITUDES
'         D      THE DESIRED ACCURACY ( UNITS )
'         AL     THE CONFIDENCE INTERVAL ( PROBABILITY )
'         AM     THE ARITHMETIC MEAN           OF THE CLUSTER MAGNITUDES
'         VS     THE SAMPLE STANDARD DEVIATION OF THE CLUSTER MAGNITUDES
'         NS     THE NORMATIVE SAMPLE SIZE
'         DL     THE LOWER CONFIDENCE LIMIT
'         DU     THE UPPER CONFIDENCE LIMIT
'
      JD = N - 1: TT = VS * THYBRID(JD, (1 - AL) / 2)
      NS = INT((TT / D) ^ 2) + 1
      A = TT / SQR(N): DL = AM - A: DU = AM + A
      END SUB

      SUB DATAIN (SW, CB, CC, EM, AO, D, AL, K, NM, SK(), IP(), SN(), AE(), AN(), PP(), V())
' A SUBROUTINE TO LOAD THE SCALAR AND LOCATIONAL DATA FOR A MAGNITUDE CLUSTER
' MIGRATION ANALYSIS
'      ARGUMENTS:
'         SW     THE MAGNITUDE UNIT NAME
'         CB     THE REPRESENTED MAGNITUDE OF THE STANDARD BELEMNOID AXIS ( MAGNITUDE UNITS )
'         CC     THE STANDARD BELEMNOID AXIS LENGTH ( MAP UNITS )
'         EM     THE HIGHEST EASTING
'         AO     THE LOWEST NORTHING
'         D      THE DESIRED ACCURACY ( UNITS )
'         AL     THE CONFIDENCE INTERVAL ( PROBABILITY )
'         K      THE NUMBER OF CLUSTERS
'         NM     THE NUMBER OF DATA OBJECTS
'         SK()   THE ARRAY OF CLUSTER NAMES
'         IP()   THE ARRAY OF PICKING LABELS
'         SN()   THE ARRAY OF DATA OBJECT NAMES
'         AE()   THE ARRAY OF DATA OBJECT EASTINGS
'         AN()   THE ARRAY OF DATA OBJECT NORTHINGS
'         PP()   THE ARRAY OF MAGNITUDES
'         V()    THE ARRAY OF OUTPUT STATISTICS
' ( IU, SP, SF AND SXV ARE COMMON SHARED )
'
      OPEN "I", IU, SP + SF + SXV
      INPUT #IU, SW
      INPUT #IU, CB, CC
      INPUT #IU, EM, AO
      INPUT #IU, D, AL
      INPUT #IU, K, NM
      FOR I = 1 TO K
         INPUT #IU, SK(I), V(I, 2)
         FOR J = 1 TO V(I, 2): INPUT #IU, IP(I, J): NEXT J
      NEXT I
      FOR I = 1 TO NM: INPUT #IU, SN(I): NEXT I
      FOR I = 1 TO NM: INPUT #IU, AE(I): NEXT I
      FOR I = 1 TO NM: INPUT #IU, AN(I): NEXT I
      FOR I = 1 TO K: FOR J = 1 TO NM: INPUT #IU, PP(I, J): NEXT J: NEXT I
      CLOSE IU
      END SUB

      SUB DATAOUT (K, CB, AX, SW, SJ(), SK(), V())
' A SUBROUTINE TO SAVE THE SCALAR AND LOCATIONAL SUMMARY STATISTICS FOR
' A MAGNITUDE CLUSTER MIGRATION ANALYSIS
'      ARGUMENTS:
'         K      THE NUMBER OF CLUSTERS
'         CB     THE REPRESENTED MAGNITUDE OF THE STANDARD BELEMNOID AXIS ( MAGNITUDE UNITS )
'         AX     THE MEAN OF THE MAGNITUDE MEANS
'         SW     THE MAGNITUDE UNIT NAME
'         SJ()   THE ARRAY OF DEFINITIONAL CAPTIONS
'         SK()   THE ARRAY OF CLUSTER NAMES
'         V()    THE ARRAY OF OUTPUT STATISTICS
' ( IV, SP, SF, SM, SCR, SXW AND ALL CLICHES ARE COMMON SHARED )
'
' CAPTION DEFINITION PHASE
      S2 = "  "
```

```
       SJ(1) = SL + SN: SJ(2) = SI + "IN " + SL: SJ(3) = SL + SE + "EASTING": SJ(4) = SL + SE
+ "NORTHING"
       SJ(5) = SL + "DYNAMIC RADIUS": SJ(6) = SV + "ARITHMETIC MEAN": SJ(7) = SV + "POPULATION
SD"
       SJ(8) = SV + SMP + "SD": SJ(9) = SV + "SKEWNESS": SJ(10) = SV + "KURTOSIS": SJ(11) =
"NORMATIVE " + SMP + "SIZE"
       SJ(12) = SO + ST: SJ(13) = SU + ST: SJ(14) = SB + SE + S2 + SX: SJ(15) = SB + SE + S2 +
SY: SJ(16) = SB + SLF + SX
       SJ(17) = SB + SLF + SY: SJ(18) = SB + SAP + SX: SJ(19) = SB + SAP + SY: SJ(20) = SB +
SRF + SX: SJ(21) = SB + SRF + SY
' DATA RECORDING PHASE
       OPEN "O", IV, SP + SF + SXW
       PRINT #IV, SF
       PRINT #IV,
       PRINT #IV, SJ(1) + SM + "STANDARD";
       FOR J = 1 TO K: PRINT #IV, SM + SK(J); : NEXT J
       PRINT #IV, SCR
       FOR I = 2 TO 21
          PRINT #IV, SJ(I);
          FOR J = 0 TO K: PRINT #IV, SM; V(J, I); : NEXT J
          PRINT #IV,
       NEXT I
       PRINT #IV, "Standard Belemnoid Mu = ";
       IF CB <= 0# THEN
             PRINT #IV, AX;
          ELSE
             PRINT #IV, CB;
       END IF
       PRINT #IV, SW
       CLOSE IV
       END SUB

       SUB DIATESSARON (J, CK, PI, AM, VP, TX, CX, CY, V())
' A SUBROUTINE TO COMPUTE THE VERTEX CO-ORDINATES OF A CLUSTER BELEMNOID
'      ARGUMENTS:
'         J      THE CLUSTER NUMBER
'         CK     THE SCALING CONSTANT
'         PI     THE LUDOLPHINE CONSTANT
'         AM     THE ARITHMETIC MEAN OF THE CLUSTER MAGNITUDES
'         VP     THE POPULATION STANDARD DEVIATION OF THE CLUSTER MAGNITUDES
'         TX     THE CLUSTER DIRECTIONAL ANGLE
'         CX     THE CLUSTER CENTROID X CO-ORDINATE
'         CY     THE CLUSTER CENTROID Y CO-ORDINATE
'         V()    THE ARRAY OF OUTPUT STATISTICS
'
       AU = AM / CK: AS2 = VP / (2 * CK): H = AS2 * TAN(PI / 5)
       V(J, 14) = CX: V(J, 15) = CY
       V(J, 16) = CX - AS2: V(J, 17) = CY - H
       V(J, 18) = CX: V(J, 19) = CY + AU
       V(J, 20) = CX + AS2: V(J, 21) = CY - H
       FOR I = 16 TO 20 STEP 2
          II = I + 1
          ROTATE TX, CX, CY, V(J, I), V(J, II), XR, YR
          V(J, I) = XR: V(J, II) = YR
       NEXT I
       END SUB

       SUB FINALE
' A SUBROUTINE TO SOUND THE PROGRAM TERMINATION SIGNAL
'
       NOTE "3C0506"
       NOTE "4D0509"
       NOTE "5E0512"
       NOTE "6F0515"
       END SUB

       SUB GETTER
' A SUBROUTINE TO ACCEPT A KEYSTROKE AS SA AND TO YIELD ITS ASCII CODE AS IA
' ( SA AND IA ARE COMMON SHARED )
       DO
          SA = INKEY$
       LOOP UNTIL SA <> ""
       IA = ASC(SA)
       END SUB

       SUB GF (NDC, N, XM(), XF(), AM, VP, VS, VQ, VK)
' A SUBROUTINE TO COMPUTE THE ARITHMETIC MEAN, POPULATION STANDARD DEVIATION, SAMPLE STANDARD
DEVIATION,
' SKEWNESS AND KURTOSIS OF A GROUPED FREQUENCY DISTRIBUTION OF CLASS-INTERVAL MID-POINTS XM(N)
AND
' CLASS INTERVAL FREQUENCIES XF(N).
```

```
' THIS SUBROUTINE WILL ALSO COMPUTE THOSE DESCRIPTIVE STATISTICS FOR A SIMPLE DATA SET IF THE
VALUES
' ARE ASSIGNED TO XM(N) AND THE PAIRED XF(N) EACH SET TO UNITY.
' THIS METHOD IS BASED UPON THAT GIVEN IN "QUANTITATIVE GEOGRAPHY" BY COLE AND KING ( SBN 471-
16475-5 ).
' THE FORMULA FOR THE SAMPLE STANDARD DEVIATION OF A GROUPED FREQUENCY DISTRIBUTION IS TAKEN
FROM
' PAGE 70 OF "STATISTICAL ANALYSIS" BY YA-LUN CHOW ( ISBN 0-03-089422-0 ).
'      ARGUMENTS:
'         NDC    THE MAXIMUM NUMBER OF DATA PER CLUSTER
'         N      THE NUMBER OF CLASS INTERVALS
'         XM()   THE ARRAY OF CLASS INTERVAL MID-POINTS
'         XF()   THE ARRAY OF CLASS INTERVAL FREQUENCIES
'         AM     THE ARITHMETIC MEAN
'         VP     THE POPULATION STANDARD DEVIATION
'         VS     THE SAMPLE     STANDARD DEVIATION
'         VQ     THE SKEWNESS
'         VK     THE KURTOSIS
'
       REDIM W(NDC + 3, 4)
       N1 = N + 1: N2 = N + 2: N3 = N + 3
       FS = 0#: FOR I = 1 TO N: FS = FS + XF(I): NEXT I
       FOR I = 1 TO N
          FOR J = 1 TO 4
             W(I, J) = XF(I) * XM(I) ^ J
             W(N1, J) = W(N1, J) + W(I, J)
          NEXT J
       NEXT I
       FOR J = 1 TO 4: W(N2, J) = W(N1, J) / FS: NEXT J
       W(N3, 2) = W(N2, 2) - W(N2, 1) ^ 2
       W(N3, 3) = W(N2, 3) - 3 * W(N2, 1) * W(N2, 2) + 2 * W(N2, 1) ^ 3
       W(N3, 4) = W(N2, 4) - 4 * W(N2, 1) * W(N2, 3) + 6 * W(N2, 1) ^ 2 * W(N2, 2) - 3 * W(N2,
1) ^ 4
       AM = W(N1, 1) / FS
       VP = SQR(W(N3, 2))
       DS = 0#: FOR I = 1 TO N: DS = DS + XF(I) * (XM(I) - AM) ^ 2: NEXT I
       VS = SQR((FS * W(N1, 2) - W(N1, 1) ^ 2) / (FS * (FS - 1)))
       VQ = W(N3, 3) / VP ^ 3
       VK = W(N3, 4) / VP ^ 4
       ERASE W
       END SUB

       SUB GND1 (AL, DV, PV)
' A SUBROUTINE TO APPROXIMATE A GAUSSIAN PROBABILITY AL FROM THE DEVIATE DV.
' THIS ROUTINE SUMMATES THE FIRST M NECESSARY AND SUFFICIENT TERMS OF A
' RAPIDLY CONVERGENT SERIES.
'      ARGUMENTS:
'         AL     THE GAUSSIAN PROBABILITY
'         DV     THE ( POSITIVE ) NORMAL DEVIATE
'         PV     THE REQUIRED MANTISSA PRECISION
' ( PI IS COMMON SHARED )
'
       M = INT(.8 + DV ^ 2.09414342149# + .921892385# * PV * DV ^ .5481692781800001#)
       AL = DV
       I = 1: J = 1: FA = 1#: RN = 1#: D = DV: DW = DV * DV
       FOR N = 1 TO M: I = -I: J = J + 2: D = D * DW: FA = FA * N: RN = 2 * RN: AL = AL + I *
D / (J * FA * RN): NEXT N
       AL = .5# - (1 / SQR(2 * PI)) * AL
       END SUB

       SUB LAGRANGIAN (N, XL(), FL(), XT, FT)
' A SUBROUTINE TO COMPUTE A LAGRANGIAN INTERPOLATE FT BEING A FUNCTIONAL VALUE
' AT ABSCISSOR XT FROM KNOWLEDGE OF THE N POLYNOMIAL CO-ORDINATES (XL(),FL())
'      ARGUMENTS:
'         N      THE NUMBER OF KNOWN CO-ORDINATES
'         XL()   THE ARRAY  OF KNOWN X CO-ORDINATES
'         FL()   THE ARRAY  OF KNOWN Y CO-ORDINATES
'         XT     THE ABSCISSAL VALUE AT WHICH FT IS TO BE ESTIMATED
'         FT     THE INTERPOLATED FUNCTION OF XT
'
       FT = 0#
          FOR I = 1 TO N
             A = 1#
                FOR J = 1 TO I - 1
                   A = A * (XT - XL(J)) / (XL(I) - XL(J))
                NEXT J
                FOR J = I + 1 TO N
                   A = A * (XT - XL(J)) / (XL(I) - XL(J))
                NEXT J
             FT = FT + FL(I) * A
          NEXT I
```

```
      END SUB

      SUB MOMENTAL (N, P(), AM, VP, VS, VQ, VR)
' A SUBROUTINE TO COMPUTE THE MOMENTAL DESCRIPTIVE STATISTICS OF
' THE SIMPLE DATA DISTRIBUTION XM().
' THE METHOD IS TAKEN FROM PAGE 30 OF "STATISTICS IN GEOGRAPHY"
' ( SECOND EDITION ) BY DAVID EBDON ( ISBN 0-631-13688-6 ).
'      ARGUMENTS:
'         N      THE NUMBER OF DATA
'         P()    THE ARRAY OF DATA VALUES
'         AM     THE ARITHMETIC MEAN
'         VP     THE POPULATION STANDARD DEVIATION
'         VS     THE SAMPLE     STANDARD DEVIATION
'         VQ     THE MOMENTAL SKEWNESS
'         VR     THE MOMENTAL KURTOSIS
'
      AM = 0#: FOR I = 1 TO N: AM = AM + P(I): NEXT I: AM = AM / N
      VP = 0#: VS = 0#: VQ = 0#: VR = 0#
      FOR I = 1 TO N
         XD = P(I) - AM
         XX = XD * XD: VP = VP + XX
         XX = XX * XD: VQ = VQ + XX
         XX = XX * XD: VR = VR + XX
      NEXT I
      VS = SQR(VP / (N - 1)): VP = SQR(VP / N)
      VQ = VQ / (N * VP ^ 3): VR = VR / (N * VP ^ 4)
      END SUB

      SUB NOTE (S)
' A SUBROUTINE TO SOUND A NOTE UPON THE COMPUTER SPEAKER
'      ARGUMENT:
'         S     THE NOTE SPECIFIER STRING "IN$NL" e.g. "2B0506"
'                  I   THE OCTAVE NUMBER        ( 0-6 )
'                  N$  THE NOTE LETTER          ( ABCDEFG )
'                  N   THE NOTE NUMBER          ( 0-84 )
'                  L   THE LENGTH OF THE NOTE   ( 1-64 )
'
      I = VAL(MID$(S, 1, 1)): N$ = MID$(S, 2, 1)
      N = VAL(MID$(S, 3, 2)): L = VAL(MID$(S, 5, 2))
      PLAY "O" + STR$(I) + "N" + STR$(N) + "L" + STR$(L) + "X" + VARPTR$(N$)
      END SUB

      SUB PICK (KT, I, L, R, S, IY, IX, MK, SF)
' A SUBROUTINE TO OBTAIN A VARIABLE OF TYPE KT AT SCREEN POSITION IY,IX
'      ARGUMENTS:
'         KT     THE DATUM TYPE CHOICE
'                   1   SHORT INTEGER
'                   2   LONG  INTEGER
'                   3   DOUBLE PRECISION REAL
'                   4   STRING
'         I      THE SHORT INTEGER
'         L      THE LONG INTEGER TO BE OBTAINED  ( OPTION )
'         R      THE REAL         TO BE OBTAINED  ( OPTION )
'         S      THE STRING       TO BE OBTAINED  ( OPTION )
'         IY     THE STARTING SCREEN ROW
'         IX     THE STARTING SCREEN COLUMN
'         MK     THE PRINTING COLOR
'         SF     THE PROPER ( OR DEFAULT ) PRINTING FORMAT
'                ( "SAnn" TRUNCATES A STRING TO nn CHARACTERS )
'
      SDEL = CHR$(0) + CHR$(83)
      COLOR MK: LOCATE IY, IX: IO = I: LO = L: RO = R: SOL = S: L1 = LEN(SF)
      SELECT CASE KT
         CASE 1
            IF LEN(STR$(I)) > L1 THEN L1 = LEN(STR$(I))
         CASE 2
            IF LEN(STR$(L)) > L1 THEN L1 = LEN(STR$(L))
         CASE 3
            IF LEN(STR$(R)) > L1 THEN L1 = LEN(STR$(R))
         CASE 4
            L1 = LEN(S)
      END SELECT
      PRINT SPACE$(L1)
      DO
         LOCATE IY, IX: PRINT "."
         FOR II = 1 TO 10: NEXT II
         LOCATE IY, IX: PRINT " "
         SA = INKEY$
      LOOP UNTIL SA <> ""
      IA = ASC(SA)
      IF IA <> 13 THEN
```

```
         SCON = "": IT = IX: LOCATE IY, IX
            DO
               IF SA = SDEL THEN
                     IT = IT - 1
                     LOCATE IY, IT: PRINT SPACE$(1)
                     LOCATE IY, IT: SCON = LEFT$(SCON, LEN(SCON) - 1)
                  ELSE
                     SELECT CASE KT
                        CASE 1 TO 3
                           IF IA > 47 AND IA < 58 OR IA = 46 OR IA = 45 THEN
                                 SCON = SCON + SA
                                 PRINT SA;
                                 IT = IT + 1
                              ELSE
                                 IF IA <> 13 THEN NOTE "2B0506"
                           END IF
                        CASE 4
                           IF IA > 31 AND IA < 127 THEN
                                 SCON = SCON + SA
                                 PRINT SA;
                                 IT = IT + 1
                              ELSE
                                 IF IA <> 13 THEN NOTE "2B0506"
                           END IF
                     END SELECT
               END IF
               GETTER
            LOOP UNTIL IA = 13
            SELECT CASE KT
               CASE 1
                  I = INT(VAL(SCON) + .5)
               CASE 2
                  L = INT(VAL(SCON) + .5)
               CASE 3
                  R = VAL(SCON)
            END SELECT
         ELSE
            I = IO: L = LO: R = RO
      END IF
      L1 = LEN(SF): IF LEN(SCON) > L1 THEN L1 = LEN(SCON)
      LOCATE IY, IX: PRINT SPACE$(L1): LOCATE IY, IX
      SELECT CASE KT
         CASE 1
            PRINT USING SF; I
         CASE 2
            PRINT USING SF; L
         CASE 3
            PRINT USING SF; R
         CASE 4
            LOCATE IY, IX: PRINT SPACE$(LEN(SCON)): LOCATE IY, IX
            IF SOL <> "" AND SCON = "" THEN SCON = SOL
            IF LEFT$(SF, 2) = "SA" THEN
                  S = LEFT$(SCON, VAL(MID$(SF, 3)))
               ELSE
                  S = SCON
            END IF
            PRINT S
      END SELECT
      END SUB

      SUB PLACE (KT, I, L, R, S, IY, IX, IU, MK, JF, SF, IB)
' A SUBROUTINE TO PLACE A VARIABLE OF TYPE KT AT SCREEN POSITION IY, IX
' OR ALTERNATIVELY PLACE THE VARIABLE WITHIN A PADDED REPORT FILE
'      ARGUMENTS:
'         KT     THE DATUM TYPE CHOICE
'                   1   SHORT INTEGER
'                   2   LONG INTEGER
'                   3   DOUBLE PRECISION REAL
'                   4   STRING
'         I      THE SHORT INTEGER TO BE PRINTED  ( OPTION )
'         L      THE LONG  INTEGER TO BE PRINTED  ( OPTION )
'         R      THE REAL          TO BE PRINTED  ( OPTION )
'         S      THE STRING        TO BE PRINTED  ( OPTION )
'         IY     THE STARTING SCREEN ROW
'         IX     THE STARTING SCREEN COLUMN
'         IU     THE LOGICAL UNIT NUMBER
'                   1   PRINT TO THE SCREEN
'                   2   PRINT TO A REPORT FILE
'         MK     THE NOMINAL PRINTING COLOR
'         JF     THE LINE FEED SUPPRESSOR SWITCH
'                   0   FOLLOW WITH A LINE FEED
```

```
'                  1   DO NOT FOLLOW WITH A LINE FEED
'         SF     THE REQUIRED PRINTING FORMAT
'         IB     THE NUMBER OF FORWARD PADDING SPACES
'
       IF SF = "" THEN SF = "#########"
       COLOR MK
       SELECT CASE IU
          CASE 1
             LOCATE IY, IX
                SELECT CASE KT
                   CASE 1
                      PRINT USING SF; I
                   CASE 2
                      PRINT USING SF; L
                   CASE 3
                      PRINT USING SF; R
                   CASE 4
                      PRINT S
                END SELECT
          CASE 2
             IF IB > 0 THEN PRINT #IU, SPACE$(IB);
             IF JF = 1 THEN SCC = ";" ELSE SCC = CHR$(13) + CHR$(10)
                SELECT CASE KT
                   CASE 1
                      PRINT #IU, USING SF; I; SCC
                   CASE 2
                      PRINT #IU, USING SF; L; SCC
                   CASE 3
                      PRINT #IU, USING SF; R; SCC
                   CASE 4
                      PRINT #IU, S
                END SELECT
       IF JF = 0 THEN PRINT #LU,
       END SELECT
       END SUB

       FUNCTION QND (AL)
' A FUNCTION TO APPROXIMATE A NORMAL DEVIATE.
' THIS ROUTINE EMPLOYS A POLYNOMIAL APPROXIMATOR DUE TO HASTINGS
' TAKEN FROM 26.2.23 ON PAGE 933 OF "HANDBOOK OF MATHEMATICAL FUNCTIONS"
' BY ABRAMOWITZ AND STEGUN ( ISBN 0-486-61272-4 ).
'       ARGUMENT:
'          AL     THE INTEGRAL PROBABILITY
' ( PI IS COMMON SHARED )
'
       DIM C(6)
       C(1) = 2.515517: C(2) = .802853: C(3) = .010328
       C(4) = 1.432788: C(5) = .189269: C(6) = .001308
       U = SQR(LOG(1 / AL ^ 2))
       U1 = C(1) + C(2) * U + C(3) * U ^ 2
       U2 = 1 + C(4) * U + C(5) * U ^ 2 + C(6) * U ^ 3
       QND = U - U1 / U2
       END FUNCTION

       SUB REFINERY2 (TL, UV, PV)
' A SUBROUTINE TO IMPROVE AN APPROXIMATE NORMAL DEVIATE.
' THIS SUBROUTINE UTILISES THE RAPIDLY-CONVERGENT SERIES
' OF THE GND1 ROUTINE
'       ARGUMENTS:
'          TL     THE REQUIRED ( PLUS OR MINUS ) TOLERANCE
'          UV     THE ESTIMATED NORMAL DEVIATE
'          PV     THE PROBABILITY ATTACHING TO THE TRUE DEVIATE
'
       DIM XL(6), FL(6)
       M = 3: TM = .00045#
       DO
          XL(1) = UV - TM: XL(2) = UV: XL(3) = UV + TM
          FOR I = 1 TO M
             GND1 FL(I), XL(I), 12#
          NEXT I
          UW = UV
          SAMEX IW, M, FL(), .000000000001#
          IF IW = 0 THEN LAGRANGIAN M, FL(), XL(), PV, UV
          TM = ABS(UV - UW)
       LOOP UNTIL TM < TL OR IW = 1
       PV = .5# - PV
       END SUB

       SUB REFINERY3 (JD, TL, UV, PV)
' A SUBROUTINE TO IMPROVE AN APPROXIMATE STUDENT'S t QUANTILE.
' THE PROBABILITIES ASSOCIATED WITH t QUANTILES ARE COMPUTED BY
```

```
' THE FINITE SERIES TECHNIQUE OF FUNCTION TPROB2
'      ARGUMENTS:
'         JD     THE DEGREES OF FREEDOM
'         TL     THE REQUIRED ( PLUS OR MINUS ) TOLERANCE
'         UV     THE ESTIMATED t QUANTILE
'         PV     THE PROBABILITY ATTACHING TO THE TRUE DEVIATE
' ( PI IS COMMON SHARED )
'
       DIM XL(6), FL(6)
       M = 3: TM = .00005#
       DO
          XL(1) = UV - TM: XL(2) = UV: XL(3) = UV + TM
          FOR I = 1 TO M
             FL(I) = TPROB2(XL(I), JD)
          NEXT I
          UW = UV
          SAMEX IW, M, FL(), 9.999999999999999D-12
          IF IW = 0 THEN LAGRANGIAN M, FL(), XL(), 1 - PV, UV
          TM = ABS(UV - UW)
       LOOP UNTIL TM < TL OR IW = 1
       END SUB

       SUB REFINERY4 (JD, TL, UV, PV)
' A SUBROUTINE TO IMPROVE AN APPROXIMATE STUDENT'S t QUANTILE.
' THE PROBABILITIES ASSOCIATED WITH t QUANTILES ARE COMPUTED BY
' THE FINITE SERIES TECHNIQUE OF FUNCTION TPROB2
'      ARGUMENTS:
'         JD     THE DEGREES OF FREEDOM
'         TL     THE REQUIRED ( PLUS OR MINUS ) TOLERANCE
'         UV     THE ESTIMATED t QUANTILE
'         PV     THE PROBABILITY ATTACHING TO THE TRUE DEVIATE
' ( PI IS COMMON SHARED )
'
       DIM XL(6), FL(6)
       M = 3: TM = .001#
       DO
          XL(1) = UV - TM: XL(2) = UV: XL(3) = UV + TM
          FOR I = 1 TO M
             FL(I) = TPROB2(XL(I), JD)
          NEXT I
          UW = UV
          SAMEX IW, M, FL(), 9.999999999999999D-12
          IF IW = 0 THEN LAGRANGIAN M, FL(), XL(), 1 - PV, UV
          TM = ABS(UV - UW)
       LOOP UNTIL TM < TL OR IW = 1
       END SUB

       SUB RESULTHETA (K, PI, TA())
' A SUBROUTINE TO COMPUTE THE CENTRAL TENDENCY OF
' THE K-1 CLUSTER DIRECTIONAL ANGLES TA().
' THE VECTORIAL RESULTANT MEAN DIRECTIONAL ANGLE IS ASSIGNED TO TA(K)
'      ARGUMENTS:
'         K      THE NUMBER OF CLUSTERS
'         PI     THE LUDOLPHINE CONSTANT
'         TA()   THE ARRAY OF DIRECTIONAL ANGLES
'
       J = 1: TC = 0#: TS = 0#
       FOR I = 1 TO K - 1
          TC = TC + COS(TA(I)): TS = TS + SIN(TA(I))
       NEXT I
       IF TC >= 0# THEN
          IF TS >= 0# THEN
                J = 0
             ELSE
                J = 2
          END IF
       END IF
       TA(K) = J * PI + ATN(TS / TC)
       END SUB

       SUB ROTATE (TH, XO, YO, XI, YI, XR, YR)
' A SUBROUTINE TO ROTATE A PAIR OF CARTESIAN CO-ORDINATES (X,Y)
' DEXTRALLY ABOUT XO,YO THROUGH ANGLE TH
'      ARGUMENTS:
'         TH     THE ANGLE OF ROTATION ( RADIANS )
'         XO     THE ABSCISSAL CENTER OF ROTATION
'         YO     THE ORDINAL   CENTER OF ROTATION
'         XI     THE UNROTATED X CO-ORDINATE
'         YI     THE UNROTATED Y CO-ORDINATE
'         XR     THE ROTATED   X CO-ORDINATE
'         YR     THE ROTATED   Y CO-ORDINATE
```

```
'
      AC = COS(TH): AI = SIN(TH): X = XI - XO: Y = YI - YO
      XR = XO + X * AC + Y * AI
      YR = YO + Y * AC - X * AI
      END SUB

      SUB SAMEX (IW, M, X(), TL)
' A SUBROUTINE TO WARN OF EQUIVALENCY BETWEEN ANY ARRAYED X()
'      ARGUMENTS:
'         IW     THE EQUIVALENCY INDICATOR SWITCH
'                   0   NO EQUIVALENCY DETECTED
'                   1   EQUIVALENCY DETECTED
'         M      THE NUMBER OF VALUES X()
'         X()    THE ARRAY OF VALUES FOR INTERCOMPARISON
'         TL     THE TOLERANCE
'
      IW = 0
      FOR I = 1 TO M
         FOR J = 1 TO I - 1
            IF X(I) < X(J) + TL AND X(I) > X(J) - TL THEN IW = 1
         NEXT J
         FOR J = I + 1 TO M
            IF X(I) < X(J) + TL AND X(I) > X(J) - TL THEN IW = 1
         NEXT J
      NEXT I
      END SUB

      SUB STANDARD (CB, CC, PI, PH, K, CK, DR(), AX, EM, AO, V())
' A SUBROUTINE TO COMPUTE THE VERTEX CO-ORDINATES OF THE STANDARD BELEMNOID
'      ARGUMENTS:
'         CB     THE REPRESENTED MAGNITUDE OF THE STANDARD BELEMNOID AXIS ( MAGNITUDE UNITS )
'         CC     THE STANDARD BELEMNOID AXIS LENGTH ( MAP UNITS )
'         PI     THE LUDOLPHINE CONSTANT
'         PH     THE RATIO OF PHIDIAS
'         K      THE NUMBER OF CLUSTERS
'         CK     THE SCALING CONSTANT
'         DR()   THE ARRAY OF DYNAMIC RADII
'         AX     THE MEAN OF MAGNITUDE MEANS
'         EM     THE HIGHEST EASTING
'         AO     THE LOWEST NORTHING
'         DT()   THE ARRAY OF BELEMNOID VERTEX CO-ORDINATES
'         V()    THE ARRAY OF OUTPUT STATISTICS
'
      IF CB <= 0# THEN
            MOMENTAL K, DR(), AM, VP, VS, VQ, VR
            AU = AM * (1 + 1 / PH ^ 2): CK = AX / AU
         ELSE
            AU = CC: AM = CC / (1 + 1 / PH ^ 2): CK = CB / CC
      END IF
      AG = SQR(2 * AM ^ 2 * (1 - COS(2 * PI / 5)))
      HS = (AG / 2) * TAN(PI / 5)
      V(0, 14) = EM - AG / 2: V(0, 15) = AO + HS
      V(0, 16) = EM - AG: V(0, 17) = AO
      V(0, 18) = EM - AG / 2: V(0, 19) = AO + HS + AU
      V(0, 20) = EM: V(0, 21) = AO
      END SUB

      FUNCTION TAPPROX (AL, JD)
' A FUNCTION TO APPROXIMATE THE STUDENT'S t-QUANTILE.
' THIS ROUTINE EMPLOYS A SERIES APPROXIMATOR TAKEN FROM 26.7.5 ON PAGE 949
' OF "HANDBOOK OF MATHEMATICAL FUNCTIONS" BY ABRAMOWITZ AND STEGUN
' ( ISBN 0-486-61272-4 )
'      ARGUMENTS:
'         AL     THE INTEGRAL PROBABILITY
'         JD     THE DEGREES OF FREEDOM
'
      X = QND(AL)
      G1 = .25 * (X ^ 3 + X)
      G2 = .01041666666# * (5 * X ^ 5 + 16 * X ^ 3 + 3 * X)
      G3 = .00260416666# * (3 * X ^ 7 + 19 * X ^ 5 + 17 * X ^ 3 - 15 * X)
      G4 = .00001085069# * (79 * X ^ 9 + 776 * X ^ 7 + 1482 * X ^ 5 - 1920 * X ^ 3 - 945 * X)
      TAPPROX = X + G1 / JD + G2 / JD ^ 2 + G3 / JD ^ 3 + G4 / JD ^ 4
      END FUNCTION

      SUB THETA (J, PI, TA(), V())
' A SUBROUTINE TO COMPUTE A DIRECTIONAL ANGLE TA(J)
'      ARGUMENTS:
'         J          THE CLUSTER NUMBER
'         PI         THE LUDOLPHINE CONSTANT
'         TA()       THE ARRAY OF DIRECTIONAL ANGLES
'         V(J,3)     THE ROW OF CENTROID X CO-ORDINATES
```

```
'         V(J,4)    THE ROW OF CENTROID Y CO-ORDINATES
'
       TX = 0#: IF V(J, 4) >= V(J + 1, 4) THEN TX = PI
       TA(J) = TX + ATN((V(J + 1, 3) - V(J, 3)) / (V(J + 1, 4) - V(J, 4)))
       END SUB

       FUNCTION THYBRID (JD, AL)
' A FUNCTION TO DETERMINE A STUDENT'S t-QUANTILE TO TWELVE-FIGURE ACCURACY
' USING KEDGING.
' THE KEDGING IS PRIMED BY THE ASYMPTOTIC EXPANSION OBJECT 26.7.5 OF
' ABRAMOWITZ AND STEGUN IF THE DEGREES OF FREEDOM IS 6 THROUGH 370:
' OTHERWISE THE INITIAL APPROXIMATION IS COMPUTED BY HILL'S PROCESS
'       ARGUMENTS:
'          JD     THE DEGREES OF FREEDOM
'          AL     THE INTEGRAL PROBABILITY
' ( HP IS COMMON SHARED )
'
       AL1 = 2 * AL
       SELECT CASE JD
          CASE 1
             P = AL1 * HP: THYBRID = COS(P) / SIN(P)
          CASE 2
             THYBRID = SQR(2 / (AL1 * (2 - AL1)) - 2)
          CASE 6 TO 370
             T = TAPPROX(AL, JD)
             REFINERY4 JD, .00000001#, T, AL
             THYBRID = T
          CASE ELSE
             A = 1 / (JD - .5)
             B = 48 / A ^ 2
             C = ((20700 * A / B - 98) * A - 16) * A + 96.36
             D = ((94.5 / (B + C) - 3) / B + 1) * SQR(A * HP) * JD
             X = D * AL1: Y = X ^ (2 / JD)
             IF Y > .05 + A THEN
                   X = UND(.5 * AL1): Y = X ^ 2
                   IF JD < 5 THEN C = C + .3 * (JD - 4.5) * (X + .6)
                   C = (((.05 * D * X - 5) * X - 7) * X - 2) * X + B + C
                   Y = (((((.4 * Y + 6.3) * Y + 36) * Y + 94.5) / C - Y - 3) / B + 1) * X
                   Y = A * Y ^ 2
                      IF Y > .002 THEN
                            Y = EXP(Y) - 1
                         ELSE
                            Y = .5 * Y ^ 2 + Y
                      END IF
                ELSE
                   Y = ((1 / (((JD + 6) / (JD * Y) - .089 * D - .822) * (JD + 2) * 3) + .5 /
(JD + 4)) * Y - 1) * (JD + 1) / (JD + 2) + 1 / Y
             END IF
             T = SQR(JD * Y)
             REFINERY3 JD, .00000001#, T, AL
             THYBRID = T
       END SELECT
       END FUNCTION

       FUNCTION TPROB2 (T, JD)
' A SUBROUTINE TO DETERMINE THE PROBABILITY ASSOCIATED WITH STUDENT'S t
' BY FINITE SERIES SUMMATION.
' DEGREES OF FREEDOM MUST BE THREE OR MORE.
'       ARGUMENTS:
'          T      THE STUDENT'S t QUANTILE
'          JD     THE DEGREES OF FREEDOM
' ( PI IS COMMON SHARED )
'
       N = INT(JD / 2 - 1): KS = 1 - 2 * (JD / 2 - INT(JD / 2))
       G = 0#: TH = ATN(T / SQR(JD))
       RD = 1#: RP = 1#: RC = COS(TH): R2 = RC * RC
       I = 1 - KS: J = -KS
       IF KS = 0 THEN RT = RC ELSE RT = 1#
       K = 0
       DO
          K = K + 1: I = I + 2: J = J + 2: RD = J / I: RP = RP * RD: RT = RT * R2: G = G + RP
* RT
       LOOP UNTIL K = N
       SELECT CASE KS
          CASE 0
             P = (2 / PI) * (TH + SIN(TH) * (COS(TH) + G))
          CASE 1
             P = SIN(TH) * (1 + G)
       END SELECT
       TPROB2 = (P + 1) / 2
       END FUNCTION
```

```
      FUNCTION UND (AL)
' A FUNCTION TO APPROXIMATE A NEGATIVE NORMAL DEVIATE.
' THIS ROUTINE EMPLOYS A POLYNOMIAL APPROXIMATOR DUE TO HASTINGS
' TAKEN FROM 26.2.23 ON PAGE 933 OF "HANDBOOK OF MATHEMATICAL FUNCTIONS"
' BY ABRAMOWITZ AND STEGUN ( ISBN 0-486-61272-4 ).
' SEGMENT REFINERY2 DUE TO JAMES R WARREN IS AN ITERATED COUPLE OF
' SERIES SUMMATION AND FOUR-POINT LAGRANGIAN INTERPOLATION
' WHICH TAKES THE APPROXIMATION TO TWELVE-FIGURE ACCURACY
'      ARGUMENT:
'         AL     THE INTEGRAL PROBABILITY
' ( PI IS COMMON SHARED )
'
      UV = QND(AL)
      REFINERY2 .0000000000001#, UV, AL
      UND = -UV
      END FUNCTION
```

Appendix Three

Program THYBRID.BAS

```
'      PROGRAM THYBRID.BAS
'      A PROGRAM TO COMPUTE STUDENT'S t-QUANTILES TO TWELVE-FIGURE ACCURACY
'      USING KEDGING.
'      FOR DEGREES OF FREEDOM 6 THROUGH 370 THE ASYMPTOTIC EXPANSION
'      ABRAMOWITZ AND STEGUN 26.7.5 IS EMPLOYED TO YIELD AN INITIAL
'      APPROXIMATION: ELSEWHERE HILL'S PROCESS IS USED
'
'      WRITTEN BY:-
'
'             JAMES R WARREN BSc MSc PhD PGCE
'             "SOUTHGATE"
'             31 VICTORIA AVENUE
'             BLOXWICH
'             WS3 3HS
'             UNITED KINGDOM
'
'             20 OCTOBER 1996
'
'      THIS PROGRAM IS WRITTEN IN MICROSOFT QBASIC
'
' VARIABLE TYPE DEFAULTS
       DEFDBL A-H, O-R, T-Z
       DEFSTR S
       DEFINT I-K, M-N
       DEFLNG L
' SEGMENT DECLARATIONS ( THEMATIC GROUPAGE )
       DECLARE SUB ALASSIGN (NA, AL())
       DECLARE SUB GETTER ()
       DECLARE SUB NOTE (S)
       DECLARE SUB WARBLE ()
       DECLARE SUB GND1 (AL, DV, PV)
       DECLARE FUNCTION QND (AL)
       DECLARE FUNCTION UND (AL)
       DECLARE FUNCTION THYBRID (JD, AL)
       DECLARE SUB REFINERY2 (TL, UV, PV)
       DECLARE SUB REFINERY3 (JD, TL, UV, PV)
       DECLARE SUB REFINERY4 (JD, TL, UV, PV)
       DECLARE SUB SAMEX (IW, M, X(), TL)
       DECLARE SUB LAGRANGIAN (N, XL(), FL(), XT, FT)
       DECLARE FUNCTION TAPPROX (AL, JD)
       DECLARE FUNCTION TPROB2 (T, JD)
' COMMON VARIABLES
       COMMON SHARED PI, HP, IA, SA, IV, SOU
' STATIC ARRAY DEFINITIONS
       DIM AL(25), XL(10), FL(10), IL(5), SB(30)
' DYNAMIC ARRAY DEFINITIONS
'      ( none )
' DEVICE ATTRIBUTIONS
       SCREEN 12: WINDOW (1, 1)-(640, 480)
' LOGICAL UNIT, TRANSPUT MODE AND DEVICE SETTINGS
       IV = 2: SXV = ".TTI"
       SP = "C:\QBASIC\QBFILES\"
       SOU = SP + "TTIMER" + SXV
       SOU = "CON"
' FORMAT DEFINITIONS
'      ( none )
' NUMERICAL CONSTANT DEFINITIONS
       PI = 3.141592653589793#: HP = PI / 2
' STRING CONSTANT DEFINITIONS
'      ( none )
' TEXT VARIABLE DEFINITIONS
'      ( none )
'
' ** THE ALGORITHM **
'
       ALASSIGN NA, AL()
       JL = 500: JU = 500: JS = 1
       NP = NA * ((JU - JL) / JS + 1)
       CLS : COLOR 15
       PRINT "FOR DEGREES OF FREEDOM "; JL; " TO "; JU; " IN STEPS OF "; JS
       PRINT NP; " ELABORATIONS OF THYBRID TAKE ";
       COLOR 14
       TI = TIMER
       FOR J = JL TO JU STEP JS: FOR I = 1 TO NA: TT = THYBRID(J, AL(I)): NEXT I: NEXT J
       TJ = TIMER
       TR = TJ - TI
       PRINT TR;
       COLOR 15
       PRINT " SECONDS"
       WARBLE
       END
```

```
      SUB ALASSIGN (NA, AL())
' A SUBROUTINE TO FILL THE ARRAY AL() WITH THE TWENTY VALUES OF INTEGRAL PROBABILITY:
' 0.25, 0.1, 0.05, 0.025, 0.01, 0.005, 0.0025, 0.001, 0.0005, 0.00025, 0.0001,
' 0.00005, 0.000025, 0.00001, 0.000005, 0.0000025, 0.000001, 0.0000005, 0.00000025,
' 0.0000001
'      ARGUMENTS:
'         NA     THE NUMBER OF INTEGRAL PROBABILITIES
'         AL()   THE ARRAY  OF INTEGRAL PROBABILITIES
'
      NA = 20
      AL(1) = .25: II = 1: IC = 1
      FOR I = 1 TO 19
         IF I = IC THEN
            AL(I + 1) = AL(I) / 2.5: II = II + 1: IC = 3 * II - 2
         ELSE
            AL(I + 1) = AL(I) / 2
         END IF
      NEXT I
      END SUB

      SUB GETTER
' A SUBROUTINE TO ACCEPT A KEYSTROKE AS SA AND TO YIELD ITS ASCII CODE AS IA
' ( SA AND IA ARE COMMON SHARED )
      DO
         SA = INKEY$
      LOOP UNTIL SA <> ""
      IA = ASC(SA)
      END SUB

      SUB GND1 (AL, DV, PV)
' A SUBROUTINE TO APPROXIMATE A GAUSSIAN PROBABILITY AL FROM THE DEVIATE DV.
' THIS ROUTINE SUMMATES THE FIRST M NECESSARY AND SUFFICIENT TERMS OF A
' RAPIDLY CONVERGENT SERIES.
'      ARGUMENTS:
'         AL     THE GAUSSIAN PROBABILITY
'         DV     THE ( POSITIVE ) NORMAL DEVIATE
'         PV     THE REQUIRED MANTISSA PRECISION
' ( PI IS COMMON SHARED )
'
      M = INT(.8 + DV ^ 2.09414342149# + .921892385# * PV * DV ^ .5481692781800001#)
      AL = DV
      I = 1: J = 1: FA = 1#: RN = 1#: D = DV: DW = DV * DV
      FOR N = 1 TO M: I = -I: J = J + 2: D = D * DW: FA = FA * N: RN = 2 * RN: AL = AL + I *
D / (J * FA * RN): NEXT N
      AL = .5# - (1 / SQR(2 * PI)) * AL
      END SUB

      SUB LAGRANGIAN (N, XL(), FL(), XT, FT)
' A SUBROUTINE TO COMPUTE A LAGRANGIAN INTERPOLATE FT BEING A FUNCTIONAL VALUE
' AT ABSCISSOR XT FROM KNOWLEDGE OF THE N POLYNOMIAL CO-ORDINATES (XL(),FL())
'      ARGUMENTS:
'         N      THE NUMBER OF KNOWN CO-ORDINATES
'         XL()   THE ARRAY  OF KNOWN X CO-ORDINATES
'         FL()   THE ARRAY  OF KNOWN Y CO-ORDINATES
'         XT     THE ABSCISSAL VALUE AT WHICH FT IS TO BE ESTIMATED
'         FT     THE INTERPOLATED FUNCTION OF XT
'
      FT = 0#
         FOR I = 1 TO N
            A = 1#
               FOR J = 1 TO I - 1
                  A = A * (XT - XL(J)) / (XL(I) - XL(J))
               NEXT J
               FOR J = I + 1 TO N
                  A = A * (XT - XL(J)) / (XL(I) - XL(J))
               NEXT J
            FT = FT + FL(I) * A
         NEXT I
      END SUB

      SUB NOTE (S)
' A SUBROUTINE TO SOUND A NOTE UPON THE COMPUTER SPEAKER
'      ARGUMENT:
'         S     THE NOTE SPECIFIER STRING "IN$NL" e.g. "2B0506"
'                  I   THE OCTAVE NUMBER        ( 0-6 )
'                  N$  THE NOTE LETTER          ( ABCDEFG )
'                  N   THE NOTE NUMBER          ( 0-84 )
'                  L   THE LENGTH OF THE NOTE   ( 1-64 )
'
      I = VAL(MID$(S, 1, 1)): N$ = MID$(S, 2, 1)
```

```
      N = VAL(MID$(S, 3, 2)): L = VAL(MID$(S, 5, 2))
      PLAY "O" + STR$(I) + "N" + STR$(N) + "L" + STR$(L) + "X" + VARPTR$(N$)
      END SUB

      FUNCTION QND (AL)
' A FUNCTION TO APPROXIMATE A NORMAL DEVIATE.
' THIS ROUTINE EMPLOYS A POLYNOMIAL APPROXIMATOR DUE TO HASTINGS
' TAKEN FROM 26.2.23 ON PAGE 933 OF "HANDBOOK OF MATHEMATICAL FUNCTIONS"
' BY ABRAMOWITZ AND STEGUN ( ISBN 0-486-61272-4 ).
'      ARGUMENT:
'         AL     THE INTEGRAL PROBABILITY
' ( PI IS COMMON SHARED )
'
      DIM C(6)
      C(1) = 2.515517: C(2) = .802853: C(3) = .010328
      C(4) = 1.432788: C(5) = .189269: C(6) = .001308
      U = SQR(LOG(1 / AL ^ 2))
      U1 = C(1) + C(2) * U + C(3) * U ^ 2
      U2 = 1 + C(4) * U + C(5) * U ^ 2 + C(6) * U ^ 3
      QND = U - U1 / U2
      END FUNCTION

      SUB REFINERY2 (TL, UV, PV)
' A SUBROUTINE TO IMPROVE AN APPROXIMATE NORMAL DEVIATE.
' THIS SUBROUTINE UTILISES THE RAPIDLY-CONVERGENT SERIES
' OF THE GND1 ROUTINE
'      ARGUMENTS:
'         TL     THE REQUIRED ( PLUS OR MINUS ) TOLERANCE
'         UV     THE ESTIMATED NORMAL DEVIATE
'         PV     THE PROBABILITY ATTACHING TO THE TRUE DEVIATE
'
      DIM XL(6), FL(6)
      M = 3: TM = .00045#
      DO
         XL(1) = UV - TM: XL(2) = UV: XL(3) = UV + TM
         FOR I = 1 TO M
            GND1 FL(I), XL(I), 12#
         NEXT I
         UW = UV
         SAMEX IW, M, FL(), .000000000001#
         IF IW = 0 THEN LAGRANGIAN M, FL(), XL(), PV, UV
         TM = ABS(UV - UW)
      LOOP UNTIL TM < TL OR IW = 1
      PV = .5# - PV
      END SUB

      SUB REFINERY3 (JD, TL, UV, PV)
' A SUBROUTINE TO IMPROVE AN APPROXIMATE STUDENT'S t QUANTILE.
' THE PROBABILITIES ASSOCIATED WITH t QUANTILES ARE COMPUTED BY
' THE FINITE SERIES TECHNIQUE OF FUNCTION TPROB2
'      ARGUMENTS:
'         JD     THE DEGREES OF FREEDOM
'         TL     THE REQUIRED ( PLUS OR MINUS ) TOLERANCE
'         UV     THE ESTIMATED t QUANTILE
'         PV     THE PROBABILITY ATTACHING TO THE TRUE DEVIATE
' ( PI IS COMMON SHARED )
'
      DIM XL(6), FL(6)
      M = 3: TM = .00005#
      DO
         XL(1) = UV - TM: XL(2) = UV: XL(3) = UV + TM
         FOR I = 1 TO M
            FL(I) = TPROB2(XL(I), JD)
         NEXT I
         UW = UV
         SAMEX IW, M, FL(), 9.999999999999999D-12
         IF IW = 0 THEN LAGRANGIAN M, FL(), XL(), 1 - PV, UV
         TM = ABS(UV - UW)
      LOOP UNTIL TM < TL OR IW = 1
      END SUB

      SUB REFINERY4 (JD, TL, UV, PV)
' A SUBROUTINE TO IMPROVE AN APPROXIMATE STUDENT'S t QUANTILE.
' THE PROBABILITIES ASSOCIATED WITH t QUANTILES ARE COMPUTED BY
' THE FINITE SERIES TECHNIQUE OF FUNCTION TPROB2
'      ARGUMENTS:
'         JD     THE DEGREES OF FREEDOM
'         TL     THE REQUIRED ( PLUS OR MINUS ) TOLERANCE
'         UV     THE ESTIMATED t QUANTILE
'         PV     THE PROBABILITY ATTACHING TO THE TRUE DEVIATE
' ( PI IS COMMON SHARED )
```

```
'
       DIM XL(6), FL(6)
       M = 3: TM = .001#
       DO
          XL(1) = UV - TM: XL(2) = UV: XL(3) = UV + TM
          FOR I = 1 TO M
             FL(I) = TPROB2(XL(I), JD)
          NEXT I
          UW = UV
          SAMEX IW, M, FL(), 9.999999999999999D-12
          IF IW = 0 THEN LAGRANGIAN M, FL(), XL(), 1 - PV, UV
          TM = ABS(UV - UW)
       LOOP UNTIL TM < TL OR IW = 1
       END SUB

       SUB SAMEX (IW, M, X(), TL)
' A SUBROUTINE TO WARN OF EQUIVALENCY BETWEEN ANY ARRAYED X()
'      ARGUMENTS:
'         IW     THE EQUIVALENCY INDICATOR SWITCH
'                   0   NO EQUIVALENCY DETECTED
'                   1   EQUIVALENCY DETECTED
'         M      THE NUMBER OF VALUES X()
'         X()    THE ARRAY OF VALUES FOR INTERCOMPARISON
'         TL     THE TOLERANCE
'
       IW = 0
       FOR I = 1 TO M
          FOR J = 1 TO I - 1
             IF X(I) < X(J) + TL AND X(I) > X(J) - TL THEN IW = 1
          NEXT J
          FOR J = I + 1 TO M
             IF X(I) < X(J) + TL AND X(I) > X(J) - TL THEN IW = 1
          NEXT J
       NEXT I
       END SUB

       FUNCTION TAPPROX (AL, JD)
' A FUNCTION TO APPROXIMATE THE STUDENT'S t-QUANTILE.
' THIS ROUTINE EMPLOYS A SERIES APPROXIMATOR TAKEN FROM 26.7.5 ON PAGE 949
' OF "HANDBOOK OF MATHEMATICAL FUNCTIONS" BY ABRAMOWITZ AND STEGUN
' ( ISBN 0-486-61272-4 )
'      ARGUMENTS:
'         AL     THE INTEGRAL PROBABILITY
'         JD     THE DEGREES OF FREEDOM
'
       X = QND(AL)
       G1 = .25 * (X ^ 3 + X)
       G2 = .01041666666# * (5 * X ^ 5 + 16 * X ^ 3 + 3 * X)
       G3 = .00260416666# * (3 * X ^ 7 + 19 * X ^ 5 + 17 * X ^ 3 - 15 * X)
       G4 = .00001085069# * (79 * X ^ 9 + 776 * X ^ 7 + 1482 * X ^ 5 - 1920 * X ^ 3 - 945 * X)
       TAPPROX = X + G1 / JD + G2 / JD ^ 2 + G3 / JD ^ 3 + G4 / JD ^ 4
       END FUNCTION

       FUNCTION THYBRID (JD, AL)
' A FUNCTION TO DETERMINE A STUDENT'S t-QUANTILE TO TWELVE-FIGURE ACCURACY
' USING KEDGING.
' THE KEDGING IS PRIMED BY THE ASYMPTOTIC EXPANSION OBJECT 26.7.5 OF
' ABRAMOWITZ AND STEGUN IF THE DEGREES OF FREEDOM IS 6 THROUGH 370:
' OTHERWISE THE INITIAL APPROXIMATION IS COMPUTED BY HILL'S PROCESS
'      ARGUMENTS:
'         JD     THE DEGREES OF FREEDOM
'         AL     THE INTEGRAL PROBABILITY
' ( HP IS COMMON SHARED )
'
       AL1 = 2 * AL
       SELECT CASE JD
          CASE 1
             P = AL1 * HP: THYBRID = COS(P) / SIN(P)
          CASE 2
             THYBRID = SQR(2 / (AL1 * (2 - AL1)) - 2)
          CASE 6 TO 370
             T = TAPPROX(AL, JD)
             REFINERY4 JD, .00000001#, T, AL
             THYBRID = T
          CASE ELSE
             A = 1 / (JD - .5)
             B = 48 / A ^ 2
             C = ((20700 * A / B - 98) * A - 16) * A + 96.36
             D = ((94.5 / (B + C) - 3) / B + 1) * SQR(A * HP) * JD
             X = D * AL1: Y = X ^ (2 / JD)
             IF Y > .05 + A THEN
```

```
                  X = UND(.5 * AL1): Y = X ^ 2
                  IF JD < 5 THEN C = C + .3 * (JD - 4.5) * (X + .6)
                  C = (((.05 * D * X - 5) * X - 7) * X - 2) * X + B + C
                  Y = (((((.4 * Y + 6.3) * Y + 36) * Y + 94.5) / C - Y - 3) / B + 1) * X
                  Y = A * Y ^ 2
                     IF Y > .002 THEN
                           Y = EXP(Y) - 1
                        ELSE
                           Y = .5 * Y ^ 2 + Y
                     END IF
               ELSE
                  Y = ((1 / (((JD + 6) / (JD * Y) - .089 * D - .822) * (JD + 2) * 3) + .5 /
(JD + 4)) * Y - 1) * (JD + 1) / (JD + 2) + 1 / Y
            END IF
            T = SQR(JD * Y)
            REFINERY3 JD, .00000001#, T, AL
            THYBRID = T
      END SELECT
      END FUNCTION

      FUNCTION TPROB2 (T, JD)
' A SUBROUTINE TO DETERMINE THE PROBABILITY ASSOCIATED WITH STUDENT'S t
' BY FINITE SERIES SUMMATION.
' DEGREES OF FREEDOM MUST BE THREE OR MORE.
'      ARGUMENTS:
'         T      THE STUDENT'S t QUANTILE
'         JD     THE DEGREES OF FREEDOM
' ( PI IS COMMON SHARED )
'
      N = INT(JD / 2 - 1): KS = 1 - 2 * (JD / 2 - INT(JD / 2))
      G = 0#: TH = ATN(T / SQR(JD))
      RD = 1#: RP = 1#: RC = COS(TH): R2 = RC * RC
      I = 1 - KS: J = -KS
      IF KS = 0 THEN RT = RC ELSE RT = 1#
      K = 0
      DO
         K = K + 1: I = I + 2: J = J + 2: RD = J / I: RP = RP * RD: RT = RT * R2: G = G + RP
* RT
      LOOP UNTIL K = N
      SELECT CASE KS
         CASE 0
            P = (2 / PI) * (TH + SIN(TH) * (COS(TH) + G))
         CASE 1
            P = SIN(TH) * (1 + G)
      END SELECT
      TPROB2 = (P + 1) / 2
      END FUNCTION

      FUNCTION UND (AL)
' A FUNCTION TO APPROXIMATE A NEGATIVE NORMAL DEVIATE.
' THIS ROUTINE EMPLOYS A POLYNOMIAL APPROXIMATOR DUE TO HASTINGS
' TAKEN FROM 26.2.23 ON PAGE 933 OF "HANDBOOK OF MATHEMATICAL FUNCTIONS"
' BY ABRAMOWITZ AND STEGUN ( ISBN 0-486-61272-4 ).
' SEGMENT REFINERY2 DUE TO JAMES R WARREN IS AN ITERATED COUPLE OF
' SERIES SUMMATION AND FOUR-POINT LAGRANGIAN INTERPOLATION
' WHICH TAKES THE APPROXIMATION TO TWELVE-FIGURE ACCURACY
'      ARGUMENT:
'         AL     THE INTEGRAL PROBABILITY
' ( PI IS COMMON SHARED )
'
      UV = QND(AL)
      REFINERY2 .0000000000001#, UV, AL
      UND = -UV
      END FUNCTION

      SUB WARBLE
' A SUBROUTINE TO SOUND A SUCCESSION OF FOUR NOTE CALLS
' ON THE COMPUTER SPEAKER
      NOTE "3C0506": NOTE "4D0509": NOTE "5E0512": NOTE "6F0515"
      END SUB

`
```

Appendix Four

The Gross Timings For The Four Methods Applied to 120 Quantile Determinations

Method 1

1 Five-Figure Accuracy
Hill's Method resources a Negative Normal Deviate determined by Hastings 26.2.23 refined by an Interpolative Iteration with Ordinates erected by Romberg Integration

This Time Test is for Interpreted QBASIC Code
Execution Time for 120 t-Quantiles = 386.1796875 seconds

Method 2

2 Five-Figure Accuracy
Hill's Method resources a Negative Normal Deviate determined by Hastings 26.2.23 refined by an Interpolative Iteration with Ordinates erected by the use the Convergent Series for Gaussian Probability

This Time Test is for Interpreted QBASIC Code
Execution Time for 120 t-Quantiles = 89.91015625 seconds

Method 3

3 Twelve-Figure Accuracy
Hill's Approximation is determined resourcing a Negative Normal Deviate determined by Hastings 26.2.23 refined by an Interpolative Iteration with Ordinates erected by use of the Convergent Series for Gaussian Probability.
Hill's Approximation is then itself refined against the Target Probability using an Interpolative Iteration whose Ordinates are erected using the Finite Trigonometric Series for the Probability associated with the Student's t-Quantile

This Time Test is for Interpreted QBASIC Code
Execution Time for 120 t-Quantiles = 108.703125 seconds

Method 4

4 Fifteen-Figure Accuracy
The Series Approximator for the t-Quantile 26.7.5 is applied to develop near three-figure accuracy.
This estimate is then refined using an Interpolative Iteration whose Ordinates are erected using the Finite Trigonometric Series for the Probability associated with the Student's t-Quantile

This Time Test is for Interpreted QBASIC Code

Execution Time for 120 t-Quantiles = 37.12890625 seconds

Appendix Five

The Tabulations of Execution Times Versus Degrees of Freedom

TABLE ONE (For Figure One)

TIMES IN SECONDS TO COMPUTE 20 AND 120 t-QUANTILES
UTILISING SERIES SUMMATIONS UNENHANCED FOR FACTORIALS AND PRODUCTS

DEGREES OF FREEDOM	EXECUTION TIME (SECONDS) METHOD ONE	TWO	THREE	FOUR
3	8.1796875	3.3984375	5.87890625	7.91015625
5	19.9921875	11.48046875	15.09765625	8.77734375
11	59.09765625	68.66015625	75.6875	11.41796875
12	67.12109375	86.1796875	93.37890625	11.80859375
17	78.8203125	117.2109375	127.5390625	15.87109375
24	78.8203125	117.2695313	132.0898438	21.3671875
27	78.76171875	118.0898438	134.7890625	24.66796875
30	78.8671875	117.1601563	135.28125	28.2890625
5;30;5	388.1015625	535.75	600.28125	103.0390625

TABLE TWO (For Figure Two)

TIMES IN SECONDS TO COMPUTE 20 AND 120 t-QUANTILES
UTILISING SERIES SUMMATIONS ENHANCED FOR PRODUCT TALLY

DEGREES OF FREEDOM	EXECUTION TIME (SECONDS) METHOD ONE	TWO	THREE	FOUR
3	8.12890625	1.26171875	3.45703125	7.140625
4	15.9296875	2.41015625	4.5	6.26953125
5	19.828125	3.12890625	5.55078125	6.4765625
8	39.37890625	7.19921875	9.609375	5.66015625
11	58.9921875	12.58203125	15.55078125	5.87890625
12	66.77734375	14.98828125	17.80078125	5.59765625
17	78.6015625	19.0625	22.30078125	6.2109375
24	78.6484375	19.05859375	22.36328125	6.37109375
27	78.7109375	19.05859375	22.7890625	6.93359375
30	78.7109375	19.05078125	22.6796875	6.9765625
5;30;5	385.3085938	89.41796875	108.2578125	38.05859375

TABLE THREE (For Figure Three)

TIMES IN SECONDS TO COMPUTE 20 AND 120 t-QUANTILES
UTILISING SERIES SUMMATIONS ENHANCED FOR PRODUCT TALLY

DEGREES OF FREEDOM	EXECUTION TIME (SECONDS) METHOD ONE	TWO	THREE	FOUR
3	8.12890625	1.26171875	3.45703125	7.140625
4	15.9296875	2.41015625	4.5	6.26953125
5	19.828125	3.12890625	5.55078125	6.4765625
6	27.62109375	4.6171875	6.859375	5.87890625
7	31.578125	5.44140625	8.08203125	6.203125
8	39.37890625	7.19921875	9.609375	5.66015625
9	47.33984375	9.33984375	12.08984375	6.0390625
10	51.12890625	10.37890625	13.0078125	5.8203125
11	58.9921875	12.58203125	15.55078125	5.87890625
12	66.77734375	14.98828125	17.80078125	5.59765625
17	78.6015625	19.0625	22.30078125	6.2109375
24	78.6484375	19.05859375	22.36328125	6.37109375
27	78.7109375	19.05859375	22.7890625	6.93359375
30	78.7109375	19.05078125	22.6796875	6.9765625
65	78.59765625	19.21875	24.8203125	11.26171875
100	78.65234375	19.23046875	26.96875	14.98828125
125	78.59765625	19.21875	28.77734375	18.1328125
133	78.6484375	19.22265625	29.16015625	18.77734375
134	78.6015625	19.22265625	29.21875	18.90234375
135	78.6484375	19.23046875	29.5	19.27734375
136	78.6484375	19.27734375	29.21875	18.83984375
137	78.6484375	19.21875	29.55078125	19.5
138	78.6484375	19.21875	29.48828125	19.33984375
150	78.65234375	19.23046875	30.16015625	20.4921875
175	78.65234375	19.23046875	31.91015625	23.5625
200	78.6484375	19.21875	33.62109375	26.41796875
250	78.6484375	19.2734375	37.734375	32.1796875
300	78.6484375	19.2265625	40.6484375	36.96875
325	78.6484375	19.21875	FOUNDER	
350	78.703125	19.28125	FOUNDER	
5;30;5	385.3085938	89.41796875	108.2578125	38.05859375

TABLE FOUR (For Figure Four)

TIMES IN SECONDS TO COMPUTE 20 AND 120 t-QUANTILES
UTILISING SERIES SUMMATIONS ENHANCED FOR PRODUCT TALLY
AND ALSO USING PRE-DIVISION IN PRODUCT SERIES COMPUTATION

DEGREES OF FREEDOM	EXECUTION TIME (SECONDS) METHOD TWO	THREE	FOUR
3	1.26171875	3.4609375	7.140625
4	2.421875	4.5	6.26171875
5	3.12890625	5.66015625	6.54296875
6	4.6171875	6.87109375	5.9296875
7	5.44140625	8.1328125	6.2109375
8	7.25	9.73046875	5.6484375
9	9.27734375	12.19921875	6.08984375
10	10.390625	13.01953125	5.87109375
20	19.28125	22.5703125	6.16015625
30	19.27734375	22.9609375	7.02734375
60	19.28125	24.390625	10.4296875
100	19.21875	27.078125	15.16015625
130	19.33984375	29.28125	18.73046875
131	19.16796875	29.16796875	18.83984375
132	19.171875	29	18.6796875
133	19.171875	29.44140625	19.37890625
134	19.16015625	29.16015625	18.890625
135	19.171875	29.55078125	19.5
138	19.171875	29.44140625	19.33984375
139	19.171875	29.66015625	19.71875
140	19.171875	29.44140625	19.33203125
150	19.28125	30.4296875	20.9296875
200	19.27734375	33.78125	26.75
300	19.27734375	41.02734375	37.5703125
360	19.28125	45.1484375	44.33203125
364	19.26953125	45.96875	45.703125
365	19.27734375	45.69140625	45.19921875
366	19.28125	46.08203125	45.91796875
367	19.21875	45.359375	46.359375
368	19.21875	45.97265625	45.80078125
369	19.28125	46.46875	46.578125
370	19.27734375	46.36328125	46.41015625
380	19.28125	47.0703125	47.5703125
385	19.28125	47.01953125	47.51171875
395	19.28125	47.7265625	48.66796875
400	19.21875	48.4375	49.87109375
500	19.22265625	54.8203125	60.1875
750	19.28125	73	90.30078125
1000	19.23046875	93.48046875	119.1796875
5;30;5	90.08203125	109.1914063	38.33984375

TABLE FIVE (For Figure Five)

TIMES IN SECONDS TO COMPUTE 20 AND 120 t-QUANTILES
UTILISING SERIES SUMMATIONS ENHANCED FOR PRODUCT TALLY
AND ALSO USING PRE-DIVISION IN PRODUCT SERIES COMPUTATION
AS INCORPORATED IN THE MIXED-METHOD PROGRAM THYBRID.BAS

DEGREES OF FREEDOM	EXECUTION TIME (SECONDS) BY ANALYTIC MEANS OR METHOD 3 OR METHOD 4
1	0.109375
2	0.05078125
3	3.3984375
4	4.44140625
5	5.55078125
6	5.8203125
7	6.09375
8	5.6015625
9	6.0390625
10	5.8203125
11	5.8203125
12	5.546875
20	6.04296875
30	6.96875
60	10.390625
100	15.046875
200	26.52734375
300	38.0625
370	46.078125
400	47.12890625
450	51.078125
500	53.94140625
5;30;5	37.01953125