SOME QUADRATURE METHODS FOR GENERAL AND SINGULAR INTEGRALS IN ONE AND TWO DIMENSIONS

BY

VINCENT U. AIHIE

A Doctoral Thesis

submitted in partial fulfilment of

the requirements for the award of Doctor of Philosophy

of the Loughborough University of Technology

October, 1987.

DECLARATION

I declare that this thesis is a record of research work carried out by me, and that the thesis is of my own composition. I also certify that neither this thesis nor the original work contained therein has been submitted to this or any other institution for a higher degree.

Vincent U. Aihie.

ACKNOWLEDGEMENTS

The author is deeply indebted to his supervisor, Dr. G.A. Evans for his support and guidance throughout the duration of the work. I am also very grateful to the Head of Department, Professor C. Storey, for his support during my stay here.

It is a pleasure to acknowledge the support of the Federal Government of Nigeria for providing the scholarship.

ABSTRACT

In this thesis numerical integration in 1- and 2-dimensions is considered. In Chapter 2, transformation methods are considered primarily for singular integrals and methods of computing the transformations themselves are derived. The well known transformation based on the IMT rule and error function are extended to non-standard functions. The implementation of these rules and their performances are demonstrated.

These transformations are then extended to two-dimensions and are used to develop accurate rules for integrating singular integrals. In addition to this, a polynomial transformation with the aim of the reduction in the number of function evaluations is also considered and the resultant product rule is applied to two-dimensional non-singular integrals.

Finally, the use of monomials in the construction of integration rules for non-singular two-dimensional integrals is considered and some rules developed. In all these situations the rules developed are tested and compared with existing methods. The results show that the new rules compare favourably with existing ones.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION AND BACKGROUND

1.1 BACKGROUND

Quadrature has been a very valuable branch of applicable mathematics and the earliest evidence of this dates back to the days of Gauss (1866). Over the years, the areas of application have grown in bounds. In this regard the question of integral transforms readily comes to mind. While direct application of quadrature to the evaluation of Laplace, Fourier, Mellin and Hankel transforms has been demonstrated by A. Erdelyi (1954) and A. Talbot (1979), of greater interest is the application to the problems arising from quantum and fluid mechanics. Recently, it was realised that the integrals occuring in the calculations of energy levels of atomic and molecular systems can be simplified by Fourier representation of the Green's functions employed. Hall (1967) first employed this idea and evaluated the transformed integrals by using Gaussian product quadrature. This approach was confirmed and improved upon by the works of M. Blakemore, G.A. Evans and J. Hyslop (1974). These later contributors noticed that the transformed integrands were highly oscillatory and achieved better results by replacing the Gaussian quadrature with more efficient special ones. A similar approach was applied to the variation iteration method by P.M. Morse and H. Feshbach (1953) on problems involving the Hartree-Fock self-consistent field method by Eyring et al (1944). Still in the same area J. Hyslop (1972) carried out the calculation of the ground state energies of hydrogen and helium and the techniques involved required the recursive use of efficient quadrature.

In fluid mechanics the story is almost the same. In 1968 G.A. Evans observed that the asymptotic form of the general Navier-Stokes equation for slow viscous flows can be simplified by the application of the complex Fourier transform. Inherent in his proposal was the inversion problem which in general is a question of multiple integrals. Evans and Ockendon (1972) dealt with this problem by using quadratures that are well adapted to highly oscillatory integrands.

Another area where numerical quadrature is indispensible is in the solution of integral equations. L.M. Delves and Mohammed (1985) and C.T. Baker (1977) have extensive chapters in their respective books showing that one of the most successful ways of solving integral equations is by a process of successive approximation. This process is accomplished by a judicious use of quadrature. Closely associated with this is the evaluation of special functions by means of quadratures. M. Abramowitz and I.A. Stegun (1965) offer a whole range of such applications. In statistics quadratures also find useful applications as statistical distributions which usually turn out as special functions which can only be evaluated numerically. The above are a few applications in which quadrature is important.

Quadratures arise as a natural consequence from ordinary differential and partial differential equations. For once the problem of finding the solution of an ordinary differential equation has been posed, the most obvious way of finding a solution is to integrate the differential equation. Thus the solution of a differential equation,

$$\underline{y}' = f(x,\underline{y}) \,, \tag{1.1.1}$$

may be formally represented as

$$\underline{y} = \int f(x,\underline{y})\,dx \,, \tag{1.1.2}$$

from which many methods based on integration arise, such as the linear multistep methods.

In partial differential equations, for example, the heat equation,

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} , \tag{1.1.3}$$

may be solved easily (in traditional manner) for linear boundary conditions of the form,

$$\frac{\partial T}{\partial x} + bT = f(t) . \tag{1.1.4}$$

But this problem becomes intractable when the boundary condition is non-linear and is given for example as,

$$\frac{\partial T}{\partial x} = T^4 \quad , \text{at } x=0 , \tag{1.1.5}$$

$$\frac{\partial T}{\partial x} = 0 \quad , \text{at } x=1 . \tag{1.1.6}$$

W. Squire (1970) showed that this problem can be simplified by converting it into a non-linear integral equation of the form,

$$T(t) = 1 - \int_0^t \frac{T^4(t)\,d\tau}{[\pi(t-\tau)]^{\frac{1}{2}}} , \tag{1.1.7}$$

which he solved by the successive approximation method.

From the above it is clear that the basic problem is the evaluation of integrals defined as,

$$\int_R f(\underline{x})\,d\underline{x} . \tag{1.1.8}$$

Here f is called the integrand which is either piecewise continuous or may contain finite discontinuities. The process of evaluation of a definite integral of the form (1.1.8) by an approximation which involves the use of a linear combination of the values of the integrand in the region of integration R, is called quadrature. This approximation is usually represented as,

$$\int_R f(\underline{x})\,d\underline{x} \cong \sum_{i=1}^{n} w_i f(\underline{x}_i) \quad , \qquad (1.1.9)$$

where the w_i's are the weight functions and the x_i's are the abscissas which usually lie in the region of integration R.

We resort to numerical quadrature for a number of reasons. First, most integrals cannot be evaluated analytically in closed forms and secondly, where analytic methods are available, they are tedious and liable to error. A further reason for using numerical quadrature is that theoretical methods are wholly inapplicable to the problems of integrating experimental data.

Quadrature rules can be classified into many groups. The broad groupings of one dimensional and multidimensional are further divided according to the type of integrand. In one dimension a number of quadrature rules have been developed for well-behaved (smooth) integrands. Among these are Simpson, Newton-Cotes, Clenshaw-Curtis (1960), Gauss and Patterson quadrature formulas. P. Davis and P. Rabinowitz (1975) showed that the composite form of Simpson is an effective integrator while the works of W.W. Johnson (1915) and V.I. Krylov (1962) show Newton-Cotes as an effective alternative. The problem which arises with high order Newton-Cotes formulae is the large alternating coefficients which induce numerical cancellation and hence instability. Not only this, there are 2n degrees of freedom in an n point rule and only n of these (the weights) are used to improve the truncation error. The Gaussian formulae however use all 2n degrees to yield a highly accurate but non-progressive method. It is the introduction of a progressive property that makes both Clenshaw-Curtis and Patterson competitive despite some loss in accuracy over Gauss for the same number of points. The Clenshaw-Curtis routine depends on the good approximation properties of Chebyshev polynomials to form a basis for a quadrature rule whose abscissae are cosine weighted ($x_n = \cos \frac{h\pi}{n}$) and hence progressive. Patterson enhances a given Gaussian formula with extra (and hence progressive) points to yield a new formula. A.H. Stroud and D. Secrest (1966) have extensive demonstrations on the choice and use of the Gaussian quadrature while the works of H. O'Hara and F.J. Smith (1969) not only present Clenshaw-Curtis as one of the best rules for smooth functions but go further to show that it is one of the quadrature rules in which the error can be computed with some success.

For highly oscillatory integrands the above rules do not generally yield good results. Here the works of R. Piessens and F. Poleunis (1971), Bakhvalov and Vasil'eva (1968), A. Alaylioglu, G.A. Evans and J. Hyslop (1973 & 1974), H. Hurwitz and P.E. Zweifel (1959), H. Hurwitz, R.A.Pfeifer and P.F. Zweifel (1959), Y.L. Luke (1954), R. Piessens (1970), R. Piessens and A. Haegeman (1973) have provided the solution. They shed light on the problem of slow convergence for increasing number of function evaluations and offer efficient rules which are based on two strategies. In the first strategy the zeros x_i of the oscillatory part of the integrand are located and each subintegral $\int_{x_i}^{x_{i+1}}$ is evaluated by a rule. Here it is advantageous to use a rule that employs the values of the integrand at the end points of the integration interval since the integrand is zero at these points and more accuracy is obtained without additional computation. The second strategy splits up the integrand into factors such that one is oscillatory and the other a smooth function. The oscillatory one which is regarded as a weight function $w(x)$ is then treated like the n-th Legendre polynomial $P_n(x)$ which changes sign n times in $[0,1]$. Piessenset al (1970) using this device developed a way to circumvent the property of sign-changing of $w(x)$ by defining $w_0=\sup|w(x)|$ for $0\leq x\leq 1$, so that $w(x)+w_0$ does not change sign on $[0,1]$. With this set up they developed Gaussian quadratures with respect to weight function $w(x)+w_0$ using in general 2n points instead of n for the evaluation of integrals. Thus these two strategies help to circumvent the problem of oscillatory integrands.

Success with singular quadrature is a very recent feature. As a general principle singular integrands are subjected to techniques which eliminate or at least change the singularity before subjecting the integrals to numerical integration. Where singularities exist at more than one point, it is always possible to partition the range into sub-intervals each containing not more than one singularity which allows different techniques to be applied to turn. So far there has been two broad trends. One approach in which success has been recorded is the one based on the convergence of sequences of quadratures. The works of A. Cohen (1980), Genz and Rowland (1973), G.A. Evans, J. Hyslop and A.P.G. Morgan (1983) are in this direction. The other strategy which has been equally successful is that pioneered by C. Schwartz (1969). In this approach the singular integrand is transformed into a function which is infinitely differentiable. With such transformations the trapezoidal rule yields very accurate results because of the Euler-Maclaurin summation formula. Contributions along this line include the IMT rule (1970) (named after Iri M., Moriguti, S., Takasawa Y. who developed the rule), the works of M. Mori (1978 & 1985), K. Murota and M. Iri (1982), G.A. Evans, R.C. Forbes and J. Hyslop (1985). These aside, the Gauss-Chebyshev, Gauss-Log, Gauss-Jacobi, quadrature rules afford additional ways of dealing with integrands where the weight functions contain the singularities. Clearly these methods lack the generality to cope with a range of singularities and it is quite laborious to generate new methods unless the resulting formulae will have extended use.

In more than one-dimension, one line of development has been the Monte Carlo approach which includes all those based on probabilistic and number theoretic considerations. The other main approach has been the systematic methods which include the bulk of the methods based principally but not exclusively on polynomial approximation. The Monte Carlo approach is usually applicable to problems with large dimensions, irregular and erratic regions with integrands which may be highly discontinuous. The accuracy that can be achieved is usually low. In the main an integral $\int_0^1 f(x)dx$ is approximated by using the idea of uniformly distributed n pairs of random numbers (x_i, y_i), such that the following relationships hold,

$$\text{(i)} \quad 0 \leq y_i \leq f(x_i) \ , \qquad (1.1.10)$$

$$\text{(ii)} \quad f(x_i) < y_i \leq 1 \ , \qquad (1.1.11)$$

For n_0 pairs of random numbers satisfying (i) the integral $\int_0^1 f(x)dx$ can be evaluated approximately as n_0/n. Hence,

$$\int_0^1 f(x)dx \simeq n_0/n \ . \qquad (1.1.12)$$

This basic idea can be extended to m dimensions. S.K. Zaremba (1970) used the approach to calculate double and multiple integrals, N.S. Bakhvalov N.M. Korobov and N.N. Cencou (1961) applied it to various classes of functions in many dimensions. Other works in this area include those of R. Cerulus and R. Hagedorn (1958), D.R. Cowdrey and C.M. Reeves (1963-1964), and P.J. Davis and P. Rabinowitz (1956).

Under the systematic methods, there are two main subdivisions. The extension of the one-dimensional quadrature by product rules, comes under the first division. The Cartesian product technique (as it is called) which involves the recursive use of one dimensional rules was extended to balls and spheres by W.H. Pierce (1957), to spheres and cones by P.C. Hammer, O.J. Marlow and A. Stroud (1958) and to spheres by A.H. Stroud and D. Secrest (1966). Further extensions of the method to cubes are due to the works of T.W. Sag and G. Szekeres (1964), while the error inherent in the product rule was considered by N.S. Bakhvalov in one of his theorems in 1959. Closely related to the works on cubes is the extension of Romberg to higher dimensions by E.B. Anders (1966). The novelty in the extended Romberg is that it could cope with singularities and give accurate results.

Another approach with a fairly long history is the method of finding formulae which achieve a given degree of precision using the fewest possible points. Quadratures derived from this approach are usually called minimal point formulae. The basis of this type of formula is usually set in a relationship of the form,

$$\int_R f\, d\underline{x} = \sum a_i f(x_r) \qquad (1.1.13)$$

where (1.1.13) is of degree say d if the following conditions are satisfied.

$$\sum a_r = \int_R d\underline{x} \,, \qquad (1.1.14a)$$

$$\sum a_r x_r^i = \int_R x_r^i d\underline{x} \,,\ i=1,2,\ldots,d \qquad (1.1.14b)$$

$$\sum a_{rs} x_r^i x_s^j = \int_R x_r^i x_s^j d\underline{x} \,,\ i,j=1,2,\ldots,d \qquad (1.1.14c)$$

$$\vdots$$

with $\begin{pmatrix} s+d \\ d \end{pmatrix}$ numbers of equations in n(s+1) unknowns.

A.H. Stroud showed that equation (1.1.14) has no solution with n less than $\begin{pmatrix} s+[\frac{d}{2}] \\ [d/2] \end{pmatrix}$.

I.P. Mysovskih (1966) showed that minimal point formulae have 2s function evaluations less when compared with the Gauss product rule of the same precision s. Thatcher (1957) and Stroud (1960) were the first to produce methods for constructing (s+1) point formulae of degree 2 for any s-dimensional region but most of their points were outside the region of integration. G.W. Tyler (1953) invented the (2s+1) points formula and this was improved by Stroud by his equal weight formula which has a low error property. Other contributions include the one by Mysovskih and the most recent ones are those of A.M. Cohen and D.A. Gismalla (1985) and L.D. Jenkins (1985). Closely related to this is the fairly general approach for finding formulae of high degree of precision based on the symmetry properties of the region. In this respect the cube, ball and sphere, centred at the origin share the property that when $(x_1 \ldots x_n)$ is a point of the domain of integration every point of the form $(\pm x_1, \pm x_2, \ldots, \pm x_n)$ is also in the domain. Using this as a basis J.N. Lyness (1965,i,ii,iii,iv,v) produced a family of formulae of odd degree of 2d+1 for cubes while J. McNamee and F. Stenger (1967) developed similar formulae for spheres and cubes. Generally the formulae from partly symmetrical region have less points than other types of formulae. From the above it can be said that there is no general theory that embodies all the rules in higher dimensions. Furthermore, our discussion has been directed at established methods with emphasis on the efficiency of the various rules and leaving out many methods, most of which are still in their infancy.

1.2 INTRODUCTION

In this work our main focus is primarily on the use of transformations and monomials in developing rules for integration in one and two dimensions. In Chapter 2 two transformations are generalised and effective methods of generating these transformations are examined. Although these transformations are aimed specifically at singular integrals, they are applied to both general and singular integrals. This was aimed at establishing (if there are) any advantages of this method over the conventional approach.

In Chapter 3, the generalised error function and the tanh transformation are extended to generate accurate rules for integrating singular integrals in two dimensions and their performance is compared with those of Romberg. In addition a different philosophy of transformation is developed for specific use with non-singular integrals in two dimensions. The sole purpose here is to reduce the number of function evaluations which is one of the major defects of the product rule.

In Chapter 4, the idea of monomials in line with the minimal formula rule is used in constructing rules for non-singular integrals in two dimensions. As in other cases above the rules are tested and compared with existing ones.

Finally, Chapter 5 contains discussions, conclusions and recommendations for future work.

CHAPTER 2

TRANSFORMATION METHODS IN 1-DIMENSIONAL SINGULAR QUADRATURE

2.1 INTRODUCTION

In this chapter the use of transformations in constructing rules for singular integrals in 1-dimension is examined. Two types of transformations - the IMT type and the error function type are generated. These examinations have been motivated by various workers on singular quadrature. While authors like P.J. Davis and P. Rabinowitz (1975) have given a general overview of the problems involved, specific work in this area includes the cautious Romberg extrapolation approach by Cohen (1980), the accelerated quadrature sequence by Chisholm, Genz and Rowland (1973) and an extrapolation procedure for the evaluation of singular integrals by G.A. Evans, J. Hyslop and A.P.G. Morgan (1983). These references employ methods based on the sequence of quadratures whose convergence is accelerated by techniques such as Romberg, Shanks (1955) or Levin (1973).

Other studies in this area include the numerical integration by C. Schwartz (1969), the integration formula based on the trapezoidal formula by F. Stenger (1973), and the works of W. Squire (1976 & 1979). More recent works in this field include polynomial transformations for singular integrals by G.A. Evans, R.C. Forbes and J. Hyslop (1984), the IMT-type transformation in numerical quadrature by K. Murota and M. Iri (1982), an IMT-type double exponential formula for numerical integration by M. Mori (1978), and quadrature formulas obtained by variable transformation and the D-E rule by M. Mori (1985). These works involve finding a transformation $x=G(t)$ which suppresses the singularities by suitable choice of $G'(t)$ which multiplies the integrand in the transformed variables. In addition the transformations go further in not only suppressing the singularities but also smoothing the integrand sufficiently to make the trapezoidal rule an efficient integrator.

The methods that are being proposed in this investigation are in the same spirit as the transformations in the preceding paragraph. In addition to this attention is paid to efficient methods for computing the transformations as the functions are no longer standard.

2.2 FORMULATION OF THE IMT TYPE OF RULE

Here we shall consider the integral of the form,

$$A = \int_{-1}^{1} f(x)dx \ , \qquad (2.2.1)$$

with the transformation,

$$x = G(t) \ , \qquad (2.2.2)$$

where G(t) is defined as,

$$G(t) = \{g(t) - g(0)\}/g(0) \ , \qquad (2.2.3)$$

and

$$g(t) = d\int_{-1}^{t} e^{-(\frac{c}{1-x^2})} dx \qquad (2.2.4)$$

Differentiating (2.2.2) we have,

$$dx = G'(t)dt \ . \qquad (2.2.5)$$

Substituting (2.2.2) and (2.2.5) in (2.2.1) we have,

$$A = \int_{-1}^{1} f\{G(t)\}G'(t)dt \ . \qquad (2.2.6)$$

Since G(t) maps the interval (-1,1) on to itself, if we put,

$$k(t) = f\{G(t)\}G'(t) \ , \qquad (2.2.7)$$

then (2.2.6) becomes,

$$A = \int_{-1}^{1} k(t)dt \ . \qquad (2.2.8)$$

From (2.2.8) it is clear that k(t) yields a factor $e^{-(\frac{c}{1-t^2})}$ which tends strongly to zero at the end points ±1 as do all the higher derivatives of the new integrand. Hence applying the

Euler-Maclaurin formula,

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{2}\{f_0+f_n\} + h\sum_{k=1}^{n-1} f(x_k) + \sum_{i=1}^{m} \frac{B_{2i}}{(2i)!} h^{2i-1}[f_n^{(2i-1)}-f_0^{(2i-1)}]$$

$$+ R_m(n) \quad , \qquad (2.2.9)$$

where R_m is expressible as,

$$R_m = nB_{2m+2}\, h^{2m+2} f^{(2m+2)}(\xi)$$

and $x_0 < \xi < x_n$

to (2.2.8) we have,

$$\int_{-1}^{1} k(t)dt = h\sum_{i=1}^{n-1} k(t_i) + R_m \quad , \qquad (2.2.10)$$

with $k^{(n)}(-1) = k^n(1) = 0, \quad (k^n = \frac{d^n}{dt^n} k(t))$.

2.3 DERIVATION AND IMPLEMENTATION OF THE FUNCTION g(t)

The function g(t) is defined as,

$$g(t) = d\int_{-1}^{t} e^{-\left(\frac{c}{1-x^2}\right)} dx \quad , \tag{2.3.1}$$

Ignoring the normalisation constant d defined as,

$$d^{-1} = \int_{-1}^{\infty} e^{-\left(\frac{c}{1-x^2}\right)} dx \quad , \tag{2.3.2}$$

the resultant function g_1 which is defined as,

$$g_1 = \int_{-1}^{t} e^{-\left(\frac{c}{1-x^2}\right)} dx \quad , \tag{2.3.3}$$

can be generated in two ways:-

(a) By Direct Integration

By using the substitution,

$$y = \frac{1}{1-x^2} \quad , \tag{2.3.4}$$

and differentiating (2.3.4) we have,

$$dy = \frac{2x dx}{(1-x^2)^2} \quad . \tag{2.3.5}$$

(2.3.5) implies that,

$$dx = \pm \left(\frac{1}{y}\right)^2 \frac{dy}{2\left(1-\frac{1}{y}\right)^{\frac{1}{2}}} \quad . \tag{2.3.6}$$

Substituting (2.3.5) and (2.3.6) in (2.3.3) we have,

$$g_1(t) = -\tfrac{1}{2}\int_{\infty}^{\left(\frac{1}{1-t^2}\right)} \frac{e^{-cy}}{y^2\left(1-\frac{1}{y}\right)^{\frac{1}{2}}} dy \quad . \tag{2.3.7}$$

Using the binomial expansion of $1/(1-\frac{1}{y})^{\frac{1}{2}}$ we have,

$$g_1(t) = -\tfrac{1}{2}\int_{\infty}^{\left(\frac{1}{1-t^2}\right)} \frac{e^{-cy}}{y^2}\left\{1+(-\tfrac{1}{2})(-\tfrac{1}{4}) + \frac{(-\frac{1}{2})(-\frac{3}{2})}{2!}\frac{1}{y^2} + \frac{(\frac{1}{2})(\frac{3}{2})(\frac{5}{2})}{3!}\frac{1}{y^3} + \frac{1.3.5.7}{(2.2.2.2)4!}\times\frac{1}{y^4} + \ldots\right\} dy \; . \qquad (2.3.8)$$

By expressing the coefficients of y^n in factorial form each term of (2.3.8) can be generalised. Hence (2.3.8) can be written as,

$$g_1(t) = \tfrac{1}{2}\sum \frac{(2n)!}{2^{2n}(n!)^2}\int_{\frac{1}{1-t^2}}^{\infty} \frac{e^{-cy}}{y^{n+2}}\,dy \; . \qquad (2.3.9)$$

The exponential function $E_n(z)$ is defined as,

$$E_n(z) = \int_1^{\infty} \frac{e^{-zt}}{t^n}\,dt \; . \qquad (2.3.10)$$

Adopting the substitution u=zt, (2.3.11)

differentiating (2.3.11) and substituting in (2.3.10) we have,

$$E_n(z) = z^{n-1}\int_z^{\infty} \frac{e^{-u}}{u^n}\,du \; . \qquad (2.3.12)$$

Hence (2.3.9) becomes,

$$g_1(t) = \tfrac{1}{2}\sum_{n=0}^{\infty} \frac{(2n)!}{2^{2n}(n!)^2}(1-t^2)^{n+1}E_{n+2}\left(\frac{c}{1-t^2}\right) \; . \qquad (2.3.13)$$

(2.3.13) can be further written as,

$$g_1(t) = \tfrac{1}{2}(1-t^2)^{n+1} \times E_2\left(\frac{c}{1-t^2}\right) + \tfrac{1}{2}\sum_{n=1} \frac{(2n)!}{2^{2n}(n!)^2}(1-t^2)^{n+1}E_{n+2}\left(\frac{c}{1-t^2}\right) \qquad (2.3.14)$$

By using the recurrence relation $nE_{n+1}(t) = e^{-t}-tE_n(t)$ (2.3.15)

(2.3.14) can be implemented in a straightforward manner with only $E_2(t)$ requiring separate calculation. The function g(t) generated in this manner is effective for the range $-1.0 \leq t \leq -0.5$. For the range of t $-0.5 \leq t \leq 0.0$, g(t) can be generated by series solution.

(b) <u>The Generation of g(t) by a Series Solution</u>

Consider $g_1(t)$ expressed as,

$$\int_{-1}^{t} e^{-\left(\frac{c}{1-x^2}\right)} dx = (1-t^2)^2 \, e^{-\left(\frac{c}{1-t^2}\right)} \hat{g}_1(t) \; . \qquad (2.3.16)$$

Differentiating (2.3.16) we have,

$$e^{-\left(\frac{c}{1-t^2}\right)} = -4t(1-t^2)e^{-\left(\frac{c}{1-t^2}\right)} \hat{g}_1(t) + (1-t^2)^2 e^{-\left(\frac{c}{1-t^2}\right)} \left\{\hat{g}_1'(t) - \frac{2ct}{(1-t^2)^2}\hat{g}_1(t)\right\} . \qquad (2.3.17)$$

(2.3.17) implies that,

$$(1-t^2)^2\hat{g}_1'(t) - \hat{g}_1(t)\,\{2t(2+c-2t^2)\} - 1 = 0 \; . \qquad (2.3.18)$$

Adopting the substitution,

$$\hat{g}_1(t) = \sum_{k=0}^{\infty} a_k t^k \; . \qquad (2.3.19)$$

Differentiating (2.3.19) implies that,

$$\hat{g}_1'(t) = \sum_{k=1}^{\infty} ka_k t^{k-1} . \qquad (2.3.20)$$

Substituting (2.3.19) and (2.3.20) in (2.3.18) we have,

$$(1-t^2)^2 \sum_{k=1} ka_k t^{k-1} - \{2t(2+c-2t^2)\} \sum_{k=0}^{\infty} a_k t^k - 1 = 0 \qquad (2.3.21)$$

By changing the index of summation and substituting for k=0(1)4 in (2.3.21) we have the following result,

$$\sum_{k=5} ka_k t^{k-1} - \sum_{k=5} 2(k-2)a_{k-2}t^{k-1} + \sum_{k=5} (k-4)a_{k-4}t^{k-1} +$$

$$+ \sum_{k=5} 4a_{k-4}t^{k-1} - \sum 2(c+2)a_{k-2}t^{k-1} + a-1+2a_2 t$$

$$-2(c+2)a_0 t+3a_3 t^2-2a_1 t^2-2(c+2)a_1 t^2+4a_4 t^3$$

$$-4a_2 t^3+4a_0 t^3-2(c+2)a_2 t^3 = 0 . \qquad (2.3.22)$$

Equating coefficients in (2.3.22) yields the following relationships,

$$\left.\begin{aligned} a_1 - 1 &= 0 \\ 2a_2 - 2(c+2)a_0 &= 0 \\ 3a_3 - 2a_1 - 2(c+2)a_1 &= 0 \\ a_0 &= e^c g_1(0) \end{aligned}\right\} \qquad (2.3.23)$$

From (2.3.23),

$$\begin{aligned} a_1 &= 1 \\ a_2 &= a_0(c+2) \end{aligned}$$

$$\left.\begin{aligned} a_3 &= a_1(6+2c) \\ a_4 &= \tfrac{1}{2}\{(4+c)a_2 - 2a_0\} \\ &\text{and for } n>4, \\ a_{n+1} &= \{(2n+2+2c)a_{n-1} - (n+1)a_{n-3}\}/(n+1)\ . \end{aligned}\right\} \qquad (2.3.24)$$

In equation (2.3.24) the two term recurrence relation is used to generate a_n for $n>4$ and so time is saved. The sum is then used to generate $g(t)$ for $-0.5 \leqslant t \leqslant 0.0$. Symmetry is then applied to complete the entire range $(-1,1)$.

2.4 IMPLEMENTATION OF THE IMT TYPE OF RULE

Consider the integral,

$$I = \int_a^b g(x)\,dx \quad . \tag{2.4.1}$$

Using the transformation,

$$x = \frac{b-a}{2} t + \frac{b+a}{2} \quad , \tag{2.4.2}$$

for $t \in (-1,1)$ and (2.2.2), (2.4.1) becomes,

$$I = \frac{b-a}{2g(0)} \int_{-1}^{1} f\left[\frac{b-a}{2} G(t) + \frac{b+a}{2}\right] e^{-(\frac{c}{1-t^2})} dt \quad . \tag{2.4.3}$$

For $t<0$ (2.4.3) gives,

$$I = \frac{b-a}{2g(0)} \int_{-1}^{1} f\left[\frac{b-a}{2}\left\{\frac{g(t)}{g(0)} - 1\right\} + \frac{b+a}{2}\right] e^{-(\frac{c}{1-t^2})} dt \tag{2.4.4}$$

and for $t>0$

$$I = \frac{b-a}{2g(0)} \int_{-1}^{1} f\left[\frac{b-a}{2}\left\{1 - \frac{g(-t)}{g(0)}\right\} + \frac{b+a}{2}\right] e^{-(\frac{c}{1-t^2})} dt \tag{2.4.5}$$

Hence the formulae in (2.4.4) and (2.4.5) reduce to,

$$\left.\begin{array}{l} I = \frac{b-a}{2g(0)} h \sum_{i=1}^{n-1} f\left[a + \frac{b-a}{2g(0)} g(t)\right] e^{-(\frac{c}{1-t^2})} \\ \text{for } t<0.0 \text{ and,} \\ I = \frac{b-a}{2g(0)} h \sum_{i=1}^{n-1} f\left[b - \frac{b-a}{2g(0)} g(-t)\right] e^{-(\frac{c}{1-t^2})} \\ \text{for } t>0.0. \end{array}\right\} \tag{2.4.6}$$

Using the double parameter coding to avoid cancellation (2.4.6) is coded as,

$$\left.\begin{aligned} I &= \frac{b-a}{2g(0)}\, h \sum_{i=1}^{n-1} \hat{f}\left[\frac{b-a}{2g(0)}\, g(t)\right] e^{-\left(\frac{c}{1-t^2}\right)} \quad \text{for } t<0.0 \\ &\text{and} \\ I &= \frac{b-a}{2g(0)}\, h \sum_{i=1}^{n-1} \hat{f}\left[\frac{b-a}{2g(0)}\, g(t)\right] e^{-\left(\frac{c}{1-t^2}\right)} \quad \text{for } t>0.0 \end{aligned}\right\} \qquad (2.4.7)$$

and the function $\hat{f}(x)$ is coded as,

$$\hat{f}(x) = \begin{cases} f(a+x) \ , & x < 0.0 \\ f(b-x) \ , & x > 0.0 \end{cases} \qquad (2.4.8)$$

This was given an extensive test and the results are documented in Tables A and B. The computer program for the work is listed in Appendix (i).

2.5 ERROR ESTIMATE IN THE IMT TYPE OF RULE

The error estimate in the proposed IMT type of rule is analysed in two ways. The approach where the error is assumed to be expressable in closed form and the recent approach by Masaloke Mori (1978) and generalised by Morota and M. Iri (1982).

(a) Closed Form Approach

F. Hildebrand (1956) showed that the error in Euler-Maclaurin's rule,

$$\int_{t_0}^{t_r} k(t)dt = h\sum_{i=1}^{r-1} k(t_i) + \tfrac{1}{2}[k(t_0)+k(t_r)] + \sum_{i=1}^{m} \frac{B_{2i}}{(2i)!}\left(k^{2i-1}(t_r)-k^{2i-1}(t_0)\right)+E_m(r) , \qquad (2.5.1)$$

is expressable in closed form as,

$$E_m(r) = \frac{rB_{2m+2}}{(2m+2)!} h^{2m+2}k^{(2m+2)}(\xi) , \quad t_0<\xi<t_r . \qquad (2.5.2)$$

From the definition of k(t) (2.4.7) becomes,

$$I = \int_{-1}^{1} k(t)dt = h\sum_{i=1}^{n-1} k(t_i) + E_m(r) . \qquad (2.5.3)$$

Hence by successive approximation similar to that of the Romberg approach $E_m(r)$ can be estimated by calculating I for h, 2h, and 4h for any particular integral. For example this principle was used to estimate the error when the integral $\int_{-1}^{1} (1-x)^{\frac{1}{2}}\cos\pi x dx$ was evaluated using (2.4.7) with c=5.62, n=32,16, and 8. On substituting in (2.5.3) and assuming $k^{2m+2}(\xi)$ is constant and represented by U we have,

$$I = -6.90686122 + 32 \frac{B_{2m+2}}{(2m+2)!} h^{2m+2} U , \qquad (2.5.4)$$

$$I = -6.90494590 + 16 \frac{B_{2m+2}}{(2m+2)!} (2h)^{2m+2} U , \qquad (2.5.5)$$

$$I = -6.78026862 + 8 \frac{B_{2m+2}}{(2m+2)!} (4h)^{2m+2)} U . \qquad (2.5.6)$$

Subtracting (2.5.4) from (2.5.5) and (2.5.6) from (2.5.5) we have,

$$-.00191532 \doteq \frac{B_{2m+2}}{(2m+2)!} U[16(2h)^{2m+2} - 32(h)^{2m+2}] , \qquad (2.5.7)$$

$$0.1265926 = \frac{B_{2m+2}}{(2m+2)!} U[16(2h)^{2m+2} - 8(4h)^{2m+2}] . \qquad (2.5.8)$$

Dividing (2.5.7) by (2.5.8) gives,

$$-.0151297943 = \frac{16(2h)^{2m+2} - 32(h)^{2m+2}}{16(2h)^{2m+2} - 8(4h)^{2m+2}} . \qquad (2.5.9)$$

(2.5.9) implies that,

$$2m+1 = 6.0464972$$

and that,

$$m = 2.523234860 \approx 3 .$$

Hence
$$E_m(r) = \frac{rB_8 h^8 U}{8!} . \qquad (2.5.10)$$

This shows that the IMT type of rule developed is of high order and that the error in using the rule is of order 8 in this particular case.

(b) K. Morota and M. Iri's (1982) approach to error estimation in using the IMT type of rule essentially uses an asymptotic expansion and saddle point analysis. Applying this to the IMT type of rule, the error can be estimated in the following way.

By the transformation g(t) the integrand, k(t) = f[G(t)]G'(t) in (2.2.7) is infinitely differentiable on (-1,1) and vanishes with all its derivatives at ±1 (end points). Hence by Fourier series k(t) can be expressed as,

$$k(t) = \sum_{n=-\infty}^{\infty} C_n e^{in\pi(t+1)} , \qquad (2.5.11)$$

and,

$$C_n = \tfrac{1}{2} \int_{-1}^{1} k(t) e^{-in\pi(t+1)} dt , \qquad (2.5.12)$$

and (2.5.11) converges absolutely on (-1,1). Using the Poisson's summation formula, the approximation,

$$S = \frac{2}{N} \sum_{\ell=1}^{N-1} k\left(\frac{2\ell}{N}\right) , \qquad (2.5.13)$$

can be equated to $\int_{-1}^{1} k(t)dt$.

Hence the N-1 point formula can be expressed in terms of the Fourier coefficients as,

$$S = \int_{-1}^{1} \sum_{n=-\infty}^{\infty} C_n \, e^{in\pi(t+1)} dt \qquad (2.5.14)$$

$$= C_0 + \sum_{p=1}^{\infty} (C_{pN} + C_{-pN}) . \qquad (2.5.15)$$

From the above, since the integral equals C_0, the error ε is given by,

$$\varepsilon = \sum_{p=1}^{\infty} (C_{pN} + C_{-pN}) , \qquad (2.5.16)$$

and hence by K. Murota and M. Iri (1982),

$$\varepsilon = 2\sum_{p=1}^{\infty} \mathrm{Re}\, C_{pN} \qquad (2.5.17)$$
$$\simeq 2\, \mathrm{Re}\, C_N$$

where $\sum_{p=1}^{\infty} \mathrm{Re}C_{pN}/\mathrm{Re}C_N \to 0$ as $N \to \infty$.
Hence the error estimation is reduced to the evaluation of the Fourier coefficients C_N of k(t).

From (2.2.7) k(t) is regular in the domain Γ which includes (-1,1) and hence by way of complex integral,

$$C_N = \int_{\Gamma} k(t)e^{-in\pi(t+1)}dt \quad . \qquad (2.5.18)$$

Differentiating (2.3.9) we have,

$$g_1'(t) = \tfrac{1}{2}\sum \frac{(2n)!}{2^{2n}(n!)^2}\, 2t(1-t^2)^n\, e^{-(\frac{c}{1-t^2})} \quad . \qquad (2.5.19)$$

By asymptotic expansion of (2.3.9) $g_1(t)$ is approximately represented as,

$$g(t) \simeq \sum \tfrac{1}{2}\frac{(2n)!}{2^{2n}(n!)^2}(1-t^2)^{n+2}\, e^{-(\frac{c}{1-t^2})}\Big\{1 - \frac{(1-t^2)}{c}(n+2) + \frac{(1-t^2)^2}{c^2}(n+3) - \dots\Big\} \qquad (2.5.20)$$

Hence,

$$g(t) \simeq g'(t)\,\frac{(1-t^2)^2}{2tc} \quad . \qquad (2.5.21)$$

Differentiating (2.3.1) we have that,

$$g'(t) = d.e^{-(\frac{c}{1-t^2})} \quad . \qquad (2.5.22)$$

Substituting (2.5.22) in (2.5.21) we have,

$$g(t) = de^{-\left(\frac{c}{1-t^2}\right)} \times \frac{(1-t^2)^2}{2tc} . \qquad (2.5.23)$$

For t>0 G(t) is defined as,

$$G(t) = 1 - \frac{g(-t)}{g(0)} . \qquad (2.5.24)$$

Differentiating (2.5.24) yields,

$$G'(t) = \frac{g'(-t)}{g(0)} . \qquad (2.5.25)$$

Substituting (2.5.21) in (2.5.24) we have,

$$G'(t) = 1 - \frac{g'(t)(1-t^2)^2}{2tcg(0)} . \qquad (2.5.26)$$

Hence $$G(t) = 1 + \frac{G'(t)(1-t^2)^2}{2tc} = 1 - \frac{d(1-t^2)^2 e^{-\left(\frac{c}{1-t^2}\right)}}{2tcg(0)} \qquad (2.5.27)$$

In order to simplify the analysis the multiplying constant d is assumed to be 1. Using symmetry and confining attention to the first quadrant we define f for real t>0 as,

$$f(z) = A(1-z)^\alpha + 0|z|^\alpha , \qquad \alpha > -1$$

then k(t) = f[G(t)]G'(t)

$$= A\left[\frac{(1-t^2)^2}{2tcg(0)} e^{-\left(\frac{c}{1-t^2}\right)}\right]^\alpha e^{-\left(\frac{c}{1-t^2}\right)} . \qquad (2.5.28)$$

(2.5.28) implies that,

$$k(t) = A \frac{[(1-t^2)^{2\alpha}]}{(2tc)^\alpha} \frac{e^{-\frac{c(\alpha+1)}{1-t^2}}}{\{g(0)\}^{\alpha+1}} \qquad (2.5.29)$$

$$\lesssim At^\alpha e^{-\frac{c(\alpha+1)}{1-t^2}} \quad \text{since } (1-t^2)^{2\alpha} < t^\alpha \quad \alpha > 0$$

as $t \to 1$,

$$\therefore \qquad C_N \leq \int_{\Gamma} \frac{At^{\alpha}}{(2c)^{\alpha}}\, g(0)^{\alpha+1}\, e^{-\frac{(\alpha+1)}{1-t^{2}}}\, e^{-iN\pi(t+1)}\,dt \qquad (2.5.30)$$

Applying the saddle point method, we can define h(t) as,

$$h(t) = -\frac{c(\alpha+1)}{t^{2}} - iN\pi(t+1)\ . \qquad (2.5.31)$$

Differentiating (2.5.31) with respect to t we have,

$$h'(t) = \frac{2c(\alpha+1)}{t^{3}} - iN\pi\ , \qquad (2.5.32)$$

and,

$$h''(t) = -\frac{6c(\alpha+1)}{t^{4}}\ . \qquad (2.5.33)$$

When h'(t) = 0, this implies that,

$$t = \left\{\frac{2c(\alpha+1)}{N\pi}\right\}^{1/3} e^{-(\frac{i\pi}{6})}\ . \qquad (2.5.34)$$

In the vicinity of the saddle point,

$$t = \left\{\frac{2c(\alpha+1)}{N\pi}\right\}^{1/3} e^{-(\frac{i\pi}{6})}$$

therefore,

$$t - \left\{\frac{2c(\alpha+1)}{N\pi}\right\}^{1/3} e^{-(\frac{i\pi}{6})} = \delta e^{i\beta}$$

hence,

$$h\left\{(\frac{2c(\alpha+1)}{N\pi})^{1/3} e^{-(\frac{i\pi}{6})} + \delta e^{i\beta})\right\}$$

$$= \frac{-c(\alpha+1)}{\left[\left\{\frac{2c(\alpha+1)}{2\pi N}\right\}^{1/3} e^{-(\frac{i\pi}{6})} + \delta e^{i\beta}\right]^{2}} -$$

$$iN\pi\left[\left\{\frac{2c(\alpha+1)}{2\pi N}\right\}^{1/3} e^{-(\frac{i\pi}{6})} + \delta e^{i\beta}+1\right] \qquad (2.5.35)$$

By binomial expansion and neglecting terms greater than order 2 (2.5.35) becomes,

$$\frac{c(\alpha+1)}{\delta^2 e^{2i}} + \frac{2c}{\delta^3}\frac{(\alpha+1)^{4/3}}{(2\pi N)^{1/3}} e^{-i[\frac{\pi}{6}+3\beta]} -$$

$$i N\pi \left[\left\{\frac{2c(\alpha+1)}{2\pi N}\right\}^{1/3} e^{-(\frac{i\pi}{6})} + \delta e^{i\beta} + 1\right] \tag{2.5.36}$$

(2.5.36) implies,

$$\frac{c(\alpha+1)}{\delta^2}\left\{\cos\beta - i\sin\beta\right\} + \frac{2c(\alpha+1)^{4/3}}{\delta^3(2\pi N)^{1/3}}\left\{\cos(\frac{\pi}{6}+3\beta) - i\sin(\frac{\pi}{6}+3\beta)\right\}$$

$$-iN\pi(\frac{2c(\alpha+1)}{2\pi N})^{1/3}\left\{\cos(\frac{\pi}{6} - i\sin(\frac{\pi}{6})\right\} + \delta\left\{\cos\beta + i\sin\beta\right\} \tag{2.5.37}$$

The real part of (2.5.37) is,

$$\frac{c(\alpha+1)}{\delta^2}\cos\beta + \frac{2c(\alpha+1)^{4/3}}{\delta^2(2\pi N)^{1/3}}\cos(\frac{\pi}{6}+3\beta) +$$

$$N\pi\left\{\frac{2c(\alpha+1)}{2\pi N}\right\}^{1/3}\left[\sin\frac{\pi}{6} - \delta\sin\beta\right] \tag{2.5.38}$$

The dominant term in (2.5.38) for very small δ is $\frac{2c(\alpha+1)^{4/3}}{\delta^3(2\pi N)^{1/3}}\cos(\frac{\pi}{6}+3\beta)$.

Hence the maximum of (2.5.38) is when,

$$\frac{\pi}{6} + 3\beta = 0 . \tag{2.5.39}$$

(2.5.39) implies that,

$$\beta = -\frac{\pi}{18} . \tag{2.5.40}$$

By the Laplace method,

$$I(s) = \int_C g(z)e^{sf(z)}dz \approx \frac{\sqrt{2\pi}g(z_0)e^{sf(z_0)}e^{i\beta}}{|sf''(z_0)|^{\frac{1}{2}}} \tag{2.5.41}$$

Hence,

$$C_N = \frac{A}{(2c)^\alpha}\frac{1}{(g(0))^{\alpha+1}}\left\{\frac{2c(\alpha+1)}{N}\right\}^{\alpha/3} e^{-(\frac{i\alpha\pi}{6})} .\sqrt{2\pi} \times$$

$$e^{-[c(\alpha+1)\left\{\frac{N\pi}{2c(\alpha+1)}\right\}^{2/3} e^{-(\frac{i\pi}{3})} + iN\pi(\left\{\frac{2c(\alpha+1)}{N\pi}\right\}^{1/3} e^{-\frac{i\pi}{6}} + 1)]} \times$$

$$\frac{1}{\{6c(\alpha+1)\}^{\frac{1}{2}}} \left\{\frac{2c(\alpha+1)}{N\pi}\right\}^{2/3} e^{-(\frac{i\pi}{3})} \tag{2.5.42}$$

$\therefore$ Taking the real part,

$$C_N = \frac{A\sqrt{2\pi}}{(2c)^{\alpha}(g(0))^{\alpha+1}} \times \left\{\frac{2c(\alpha+1)}{N\pi}\right\}^{\alpha/3} \cos\frac{\alpha\pi}{6} \times$$

$$e^{-[c(\alpha+1)\left\{\frac{N\pi}{2c(\alpha+1)}\right\}^{2/3} \cos\frac{\pi}{3} + N\pi\left\{\frac{2c(\alpha+1)}{N\pi}\right\}^{1/3} \sin\frac{\pi}{6}}$$

$$\times \left\{\frac{2c(\alpha+1)}{N\pi}\right\}^{2/3} \cos\frac{\pi}{3} \times \frac{1}{6c(\alpha+1)^{\frac{1}{2}}} . \tag{2.5.43}$$

When $\alpha=0$ the value of C_N gives the intrinsic error of the IMT type of formula.

2.6 THE ERROR FUNCTION TYPE OF TRANSFORMATION

H. Takahasi and H. Mori first used the transformation of the form,

$$x = \phi(t) \quad , \tag{2.6.1}$$

where $\phi(t)$ is defined as the error function. In this investigation this transformation is generalised. The generalised transformation is defined as,

$$g_2(t) = C \int_0^t e^{-x^{2n}} dx \quad , \tag{2.6.2}$$

where C is the normalisation constant defined as,

$$1/C = \int_0^\infty e^{-x^{2n}} dx \quad . \tag{2.6.3}$$

The transformation $g_2(t)$ maps the interval (-1,1) into $(-\infty,\infty)$ and it is generated in two ways, valid in complimentary regions.

(a) Generation of $g_2(t)$ by Series Expansion

For small t and ignoring the normalisation constant C, $g_2(t)$ can be generated by series expansion of $\int_0^t e^{-x^n} dx$ which is expressed as,

$$\int_0^t e^{-x^{2n}} dx = \sum_{m=0}^{\infty} \frac{(-1)^m t^{2mn+1}}{m!(2mn+1)} \quad . \tag{2.6.4}$$

This series suffers from instability for increasing values of t because of large alternating terms causing cancellation before the series falls off eventually for large m. Hence $g_2(t)$ is generated in an alternative way as in (b).

(b) <u>Differential Form of $g_2(t)$</u>

Ignoring the constant C, $g_2(t)$ can be expressed as,

$$\int_0^t e^{-x^{2n}}dx = e^{-t^{2n}}\hat{g}_2(t) . \tag{2.6.5}$$

Differentiating (2.6.5) we have,

$$\hat{g}_2'(t) - 2nt^{2n-1}\hat{g}_2(t) = 1 . \tag{2.6.6}$$

Adopt the substitution,

$$\hat{g}_2(t) = \sum_{i=0}^{\infty} a_i t^i . \tag{2.6.7}$$

Differentiating (2.6.7) gives,

$$\hat{g}_2'(t) = \sum_{i=1}^{\infty} ta_i t^{i-1} . \tag{2.6.8}$$

Substituting (2.6.7) and (2.6.8) in (2.6.6) we have,

$$\begin{aligned} &a_1 + 2a_2t + 3a_3t^2 + 4a_4t^3 + \dots \\ &\quad -2n(a_0t^{2n-1} + a_1t^{2n} + a_2t^{2n+1} \dots a_it^{2n+i-1}) = 1 \end{aligned} \tag{2.6.9}$$

Equating coefficients in (2.6.7) and (2.6.9) we have,

$$\left.\begin{aligned} &\hat{g}_2(0.0) = 0.0 = a_0 \\ &a_1 = 1 \\ &a_2, a_3, \dots, a_{2n-1} = 0 \\ &2na_{2n} - 2na_0 = 0 \\ &\vdots \\ &(2n+1)a_{2n+1} - 2na_1 = 0 \\ &\vdots \\ &(2n+i)a_{2n+i} - 2na_i = 0 \end{aligned}\right\} \tag{2.6.10}$$

for terms containing t^{2n+i-1}.

Hence,

$$\hat{g}_2 \quad = \quad t + \frac{2n}{2n+1}\, t^{2n+1} + \frac{2n}{2n+1} \cdot \frac{2n\, t^{4n+1}}{4n+1} + \ldots \qquad (2.6.11)$$

The terms in (2.6.11) can be generalised and expressed as,

$$\hat{g}_2 \quad = \quad t + \sum_{m=1}^{\infty} \frac{(2n)^m t^{2nm+1}}{(2n+1)(4n+1)\ldots(2mn+1)} \; . \qquad (2.6.12)$$

The series is effective for $0 \leqslant t \leqslant 5$ and $n=1$, for $0.0 \leqslant t \leqslant 2.0$ for $n=2$ and for $0 \leqslant t \leqslant 1.25$ and $n=3$. For large $t, g_2(t)$ is generated by asymptotic expansion.

(c) <u>Generation of $g_2(t)$ by Asymptotic Expansion</u>

For large $t, g_2(t)$ is generated by an asymptotic expansion formula based on the successive integration by parts of I_{2n} defined as,

$$I_{2n} \quad = \quad \int_t^{\infty} e^{-x^{2n}}\, dx \; . \qquad (2.6.13)$$

By putting, $\quad y \quad = \quad x^{2n} \; ,$ $\qquad (2.6.14)$

and differentiating (2.6.14) we have,

$$2nx^{2n-1} dx \quad = \quad dy \quad . \qquad (2.6.15)$$

Substituting (2.6.14) and (2.6.15) in (2.6.13) we have,

$$I_{2n-1} \quad = \quad \int_t^{\infty} \frac{e^{-y}}{2nx^{2n-1}}\, dy \; , \qquad (2.6.16)$$

(2.6.16) implies,

$$I_{2n-1} \quad = - \int_t^{\infty} \frac{d(e^{-y})}{2nx^{2n-1}} \quad . \qquad (2.6.17)$$

Integrating (2.6.17) by parts we have,

$$I_{2n-1} = \left. \frac{-e^{-y}}{2nx^{2n-1}} \right|_t^\infty + \int_t^\infty \frac{e^{-y}}{2n} \frac{(-2n+1)}{x^{2n}} \frac{dy}{2nx^{2n-1}} \quad . \tag{2.6.18}$$

Simplifying (2.6.18) we have,

$$I_{2n-1} = \frac{e^{-t^{2n}}}{2nt^{2n-1}} - \int_t^\infty \frac{(2n-1)}{(2n)^2} \frac{e^{-y}dy}{x^{4n-1}} \quad . \tag{2.6.19}$$

$$= \frac{e^{-t^{2n}}}{2nt^{2n-1}} - \frac{(2n-1)}{(2n)^2} I_{4n-1} \quad . \tag{2.6.20}$$

By repeating the integration we have from (2.6.20),

$$I_{2n-1} = \frac{e^{-t^{2n}}}{2nt^{2n-1}} - \frac{(2n-1)}{(2n)^2} \frac{e^{-t^{2n}}}{t^{4n-1}} + \frac{(4n-1)(2n-1)}{(2n)^2(2n)} \int_t^\infty \frac{e^{-y}dy}{x^{6n-1}} \quad . \tag{2.6.21}$$

Hence (2.6.21) can be generalised to give,

$$I_{2n-1} = e^{-t^{2n}} \sum (-1)^{m+1} \frac{(2n-1)\ldots(2mn-1)(2n(m+1)-1)}{(2n)^m t^{2mn-1}} \quad . \tag{2.6.22}$$

But $g_2(t) = 1 - I_{2n}C$, and hence $g_2(t)$ can be generated for large t by (2.6.22).

2.7 IMPLEMENTATION OF THE ERROR TYPE OF TRANSFORMATION

For accurate integration the distance from the curve $x=g_2(t)$ to the asymptote $x=1$ needs to be computed accurately and this is given directly by (2.6.22). On a low accuracy machine calculating the distance $1-g_2(t)$ from (2.6.12) can be a problem as $g_2(t)$ tends rapidly to 1.0. This problem is handled by using a short range Gaussian quadrature to give accurate values of the area between $x=g_2(t)$ and its asymptote. Hence for a seven digit machine a 16 point Gauss Legendre formula was used to evaluate the integral,

$$I = \int_t^N e^{-x^{2n}} dx \; . \tag{2.7.1}$$

The choice of N is such that,

$$e^{-N^{2n}} = 10^{-m} \; , \tag{2.7.2}$$

where 10^{-m} is the smallest real representation of the machine used for the exponent. Hence,

$$N = \exp(\ell n(m(\ell n(10)*2n) \; . \tag{2.7.3}$$

Integrals of the general form,

$$S = \int_a^b f(x)dx \; , \tag{2.7.4}$$

where $f(x)$ may have singularities at end points were considered. Using the standard substitution,

$$x = \tfrac{1}{2}(b+a)+\tfrac{1}{2}(b-a)g(t) \; , \tag{2.7.5}$$

where $g(t)$ is defined as,

$$g(t) = \frac{1}{c}\int_0^t e^{-x^{2n}}\,dx\ , \tag{2.7.6}$$

(2.7.4) becomes,

$$S = \frac{1}{2c}(b-a)\int_{-\infty}^{\infty} f[\tfrac{1}{2}(b+a)+\tfrac{1}{2}(b-a)g(t)]\,e^{-t^{2n}}\,dt \tag{2.7.7}$$

Applying the trapezoidal rule to (2.7.7) we have,

$$\frac{h}{2c}(b-a)\sum_{i=-N}^{N} f[\tfrac{1}{2}(b+a)+\tfrac{1}{2}(b-a)g(t)]\,e^{-t^{2n}}\,dt \tag{2.7.8}$$

where $t=ih$, $i=0,\pm1,\pm2,\pm3,\ldots$, with the summation truncated at $i=\pm N$ when further contribution becomes insignificant. (2.7.8) is implemented by taking an initial step size h (such as 0.5) and stepping from 0.0 on both sides until further contributions to the quadrature are no longer significant. This fixes the upper limit of summation and yields the initial estimate of the quadrature. Further intermediate points are then injected into the underlying rule in a progressive manner increasing the accuracy until convergence is achieved. In addition a double parameter form of f is used to handle end point cancellation. For example the integral $\int_0^1 \frac{dx}{\sqrt{1-x}}$ has a singularity at the point 1.0. Hence for f(x),

$$f(x) = \frac{1}{\sqrt{1-x}} \text{ is coded as,}$$

$$f = \begin{cases} \dfrac{1}{\sqrt{1-x}} & \text{for } x < 0.5 \\[2ex] \dfrac{1}{\sqrt{d}} & \text{otherwise.} \quad \text{where } d = 1-x \end{cases}$$

The rules developed as a result of the error type of transformation were given an extensive test and the result is documented in Tables A and B. Appendix (ii) contains the program for this work.

2.8 ERROR ESTIMATE OF THE ERROR TYPE OF RULE

For the transformation

$$x = \operatorname{erf} u \tag{2.8.1}$$

the integral,

$$I = \int_{-1}^{1} f(x)\,dx$$

becomes,

$$I = \int_{-\infty}^{\infty} f(\operatorname{erf} u)\frac{2}{\sqrt{\pi}}\exp(-u^2)\,du \ . \tag{2.8.2}$$

Using the trapezoidal rule on (2.8.2), we have,

$$I_T = \frac{2h}{\sqrt{\pi}} \sum_{n=-\infty}^{\infty} f(\operatorname{erf} u)\exp(-n^2h^2) \ . \tag{2.8.3}$$

By defining the function f as,

$$f(x) = (1-x^2)^{-\alpha}, \quad \alpha<1 \ , \tag{2.8.4}$$

and using the saddle point analysis on the transformed integrand,

$$F(w) = \hat{\phi}(w)\frac{2}{\sqrt{\pi}}\exp(-w^2)(1-\operatorname{erf}^2 w)^{-\alpha} \ , \tag{2.8.5}$$

where $$\hat{\phi}(w) = \int_{-\infty}^{\infty}\frac{du}{w-u} - \sum\frac{A_i}{w-a_i} \tag{2.8.6}$$

with A_i as the coefficients and a_i as the abscissae of the trapezium rule. H. Takahashi and M. Mori (1970) establish the error in using (2.8.3), as,

$$\varepsilon(h) \simeq 2\pi|\phi(w)| \simeq 2\pi\exp(\frac{-12.5}{h}) \tag{2.8.7}$$

and that the truncated error $\varepsilon(N)$ is,

$$\varepsilon(N) \approx 2\pi\exp(-3.4\times\sqrt[3]{1-\alpha}N^{2/3}) \tag{2.8.8}$$

When $\alpha=0$ the inherent error of the method was order $\exp\left(\frac{-\pi^2}{h^2}\right)$.

Using the generalised transformation,

$$x = \frac{2}{\sqrt{\pi}} \int_0^u \exp(-t^{2n})\,dt \ , \qquad (2.8.9)$$

the generalised error of the trapezoidal rule can be written as,

$$\varepsilon(h) \simeq 2\pi\exp\left(\frac{-n*12.5}{h}\right) \qquad (2.8.10)$$

the generalised form of the truncated error as,

$$\varepsilon(N) \approx 2\pi\exp(-n\times 3.4\times\sqrt[3]{(1-\alpha)}\ N^{2/3}) \qquad (2.8.11)$$

and the inherent error of the method is expressible as of order $\exp(-(\pi/h)^{2n})$.

2.9 RESULTS

The rules developed were applied to the following test integrals.

$$I_1 = \int_0^1 x^{-\alpha} dx = 10^6 \quad (\alpha=1-10^{-6})$$

$$I_2 = \int_0^1 x^{0.95} e^x \, dx = 1.020457359$$

$$I_3 = \int_0^1 (1 + x^2)^{-1} (\ln x)^2 \, dx = 1.937892293$$

$$I_4 = \int_0^1 x^{-1/2} e^{-x} (1 + x)^{-1} \, dx = 1.237643927$$

Test integrals of Chrisholm et al.

$$J_1 = \int_0^{2\pi} \ln x \sin x \, dx = 2.437653393$$

$$J_2 = \int_0^1 x^{3/2} \, dx = 0.4$$

$$J_3 = \int_0^1 x^{1/2} \ln x \, dx = 4/9$$

$$J_4 = \int_0^1 x^{3/4} \cos x \, dx = 0.4451649239$$

$$J_5 = \int_0^1 x^{-1/2} \, dx = 2$$

$$J_6 = \int_0^1 [x^{1/2} + x^{1/3}]^{-1} \, dx = 0.8411169166$$

$$J_7 = \int_0^1 \ln(1 - \cos x) \, dx = -2.721065445$$

$$J_8 = \int_0^1 x^{-1/2} \ln x \, dx = -4$$

$$J_9 = \int_0^\infty ue^{-u}(1+u^2)^{-1}du = \int_0^1 u(1+u^2)^{-1}dx = 0.3433779615$$

$$J_{10} = \int_0^\infty \ell^{-u}(1+u)^{-1/2}du = \int_0^1 (1+u)^{-1/2}dx = 0.7578721561$$

$$J_{11} = \int_0^\infty \ell^{-u} u^{7/2}du = \int_0^1 u^{7/2} dx = 11.63172840$$

$$J_{12} = \int_0^\infty \ell^{-u}u^{-1/2}(1+u)^{-1}du = \int_0^1 u^{-1/2}(1+u)^{-1}dx$$

$$= 1.343293422$$

$(u = -\ln x)$.

Test integrals of Harris and Evans,

$$K_1 = \int_0^1 x^{1/2} dx = \frac{2}{3}$$

$$K_2 = \int_0^1 x^{-1/3} dx = \frac{3}{2}$$

$$K_3 = \int_0^1 x^{-2/3} dx = 3$$

$$K_4 = \int_0^1 x^{7/2} dx = \frac{2}{9}$$

$$K_5 = \int_0^1 (\ln x)^2 dx = 2$$

$$K_6 = \int_0^1 (\ln x)^4 dx = 24$$

$$K_7 = \int_0^1 (1+x^2)^{-1} dx = \frac{\pi}{4}$$

$$K_8 = \int_0^1 x^{-1/2} \ln x \, dx = -4$$

$$K_9 = \int_0^1 (1-\ln x)^{-1} (-\ln x)^{-1/2} dx = 1.343293422$$

Test integrals with singularities at end points,

$$L_1 = \int_0^1 x^{-1/2} (1 - x)^{-1/2} \, dx = \pi$$

$$L_2 = \int_0^1 x^{-1/2} \ln \ln x^{-1} \, dx = 0.2318630313$$

$$L_3 = \int_0^1 (1 + x)^{-2} \ln \ln x^{-1} \, dx = -0.06281647981$$

$$L_4 = \int_0^1 (1 - \ln x)^{-1} (-\ln x)^{-1/2} \, dx = 1.343293422$$

$$L_5 = \int_0^1 \ln x \ln(1 - x) \, dx = 0.3550659332$$

$$L_6 = \int_0^1 (1 - x)^{-1} \ln x \, dx = -1.644934067$$

$$L_7 = \int_0^1 (x - 2)^{-1} (1 - x)^{-1/4} (1 + x)^{-3/4} \, dx =$$
$$= -1.949054259166746$$

H. O'Hara and Francis J. Smith non-singular test integrals,

$$OS_1 = \int_0^1 \frac{dx}{1 + x} = 0.69314718$$

$$OS_2 = \int_0^1 \frac{dx}{1 - 0.5x^4} = 1.14366727$$

$$OS_3 = \int_0^1 \frac{dx}{1 + 100x^2} = 0.147112768$$

$$OS_4 = \int_0^1 \phi(x)\,dx\ , \quad \phi(x) = \begin{cases} \ell^x, & x < \frac{1}{2} \\ \frac{1}{2}(1+\ell^{\frac{1}{2}}), & x = \frac{1}{2} \\ \ell^{x-\frac{1}{2}}, & x > \frac{1}{2} \end{cases}$$

$$OS_5 = \int_0^1 \frac{4\,dx}{1 + 256(x - 3/8)^2} = 0.17979846$$

$$OS_6 = \int_0^1 \frac{dx}{1 - 0.98x^4} = 1.89633557$$

$$OS_7 = \int_0^1 \frac{dx}{1 + x^2} = 0.785398148$$

and the results compared with known methods.

Tables A and B show a comparative performance of the new rules with singular and non-singular integrals respectively. In Table B the rules were compared with Clenshaw Curtis.

In general the new rules compare very well with existing methods. The IMT type of rule is very difficult to apply at its best efficiency because there is no way of knowing the appropriate choice of C beforehand and this experiment has demonstrated that there is no discernable trend in the way of choosing the value of C. Among the erf class, the erf function which is recovered for n=1 failed in J_2, K_4 and L_8. The erf class with n=2 appears to be the best in that the number of function evaluations is generally less than when n=3. However the new rules compare favourably with other existing methods.

In addition the new rules are also robust for non-singular integrals but they require too many function evaluations (as demonstrated in Table B). So the recommendation is that where possible the new rules should be used for singular integrals only.

INTEGRALS	I_1	I_2	I_3	I_4	J_1	J_2	J_3	J_4	J_5	J_6	J_7
Method 1 - Evans G.A., Hyslop J. and Morgan A.P.C.(1982)	21	27	111	127	57	29	59	51	45	133	87
Method 2 - Evans G.A., Forbes F.C. and Hyslop J.(1985)	-	15	31	15	31	15	15	15	3	15	15
Method 3 - Evans G.A., Forbes F.C. and Hyslop J.(1984)	-	25	31	29	33	25	21	29	21	21	25
IMT TYPE	-	16	16	32	16	16	16	16	16	16	16
WITH C VALUES		5.2	6.01	5.23	3.55	5.205	5.2	5.233	4.49	3.324	5.19
erf CLASS											
WITH n = 1	-	13	21	13	29	-	15	13	25	21	17
n = 2	-	17	21	21	33	13	17	17	21	21	17
n = 3	-	25	33	33	25	25	25	25	33	33	25

TABLE A

INTEGRALS	J_8	J_9	J_{10}	J_{11}	J_{12}	K_1	K_2	K_3	K_4	K_5	K_6
Method 1 - Evans G.A., Hyslop J. and Morgan A.P.C.(1982)	75	79	79	141	127	45	45	45	17	97	141
Method 2 - Evans G.A., Forbes F.C. and Hyslop J.(1985)	31	31	15	31	31	7	3	3	15	15	31
Method 3 - Evans G.A., Forbes F.C. and Hyslop J.(1984)	25	33	25	25	25	25	23	25	29	25	25
IMT TYPE	16	16	16	16	32	16	16	16	16	32	32
WITH C VALUES	5.19	4.671	5.201	5.04	1.691	3.389	3.185	3.537	3.37	3.37	3.635
erf CLASS											
WITH n = 1	27	33	17	21	15	15	21	29	-	21	21
n = 2	25	17	17	21	17	17	21	25	13	21	21
n = 3	33	33	33	33	33	25	33	13	25	33	33

TABLE A (continued)

INTEGRALS	K_7	K_8	K_9	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8
Method 1 - Evans G.A., Hyslop J. and Morgan A.P.C.(1982)	29	75	129	127	135	110	127	54	88	150	105
Method 2 - Evans G.A., Forbes F.C. and Hyslop J.(1985)	31	31	31	31	31	31	31	31	31	31	63
Method 3 - Evans G.A., Forbes F.C. an- Hyslop J.(1984)	29	25	25	21	25	29	25	17	17	33	41
IMT TYPE	32	32	32	16	16	16	32	32	32	32	32
WITH C VALUES	3.6	3.63	3.63	5.53	1.64	1.53	1.0922	1.092	1.077	4.8	5.62
erf CLASS											
WITH n = 1	33	27	15	25	27	19	15	13	19	33	-
n = 2	17	25	17	21	25	21	17	17	21	25	21
n = 3	33	33	33	33	33	33	33	25	33	33	33

TABLE A(continued)

INTEGRALS	OS_1	OS_2	OS_3	OS_4	OS_5	OS_6	OS_7
Method 4 - Clenshaw C.W. and Curtis A.R. (1960)	8	16	16	16	32	16	16
IMT TYPE	32	32	32	32	-	32	32
WITH C VALUES	5.1	5.1	5.2	1.745		4.8	5.1
erf CLASS							
WITH n = 1	17	33	37	129	257	33	33
n = 2	17	17	41	129	129	65	17
n = 3	33	33	65	129	129	65	33

<u>TABLE B</u>

CHAPTER 3

TRANSFORMATION METHODS IN 2-DIMENSIONAL QUADRATURE

3.1 INTRODUCTION

For accurate work, separate rules are usually developed for singular and non-singular integrals. The first part of this chapter is devoted to singular 2-dimensional quadratures while the second half deals with non-singular quadrature in 2-dimensions.

Unlike the situation in one-dimension there has not been much progress in developing systematic rules for multi-dimensional singular integrals. Some of the attempts so far include the numerical evaluation of higher dimensional integrals by T.W. Sag and G. Szekeres (1964), the Gauss-Legendre and Gauss-Hermite product rules, and the variational principle for integrals by C. Schwartz (1969). T.W. Sag and G. Szekeres developed rules similar to the IMT rule for the interval (-1,1) by choosing a transformation which ensures that the integrand and its derivatives vanish at the end points of the interval. Using the trapezoidal product rule, they produced a formula which could be easily extended to higher dimensions. In order to operate their new rule in higher dimensions and transfer the advantages of the vanishing derivatives and the transformed integrand at the boundary, they established that the unit sphere was the most suitable region for which such transformations were valid. In this way they produced a quadrature formula which was more reliable than the Monte Carlo method and also produces better results. The Gauss-Legendre and Gauss-Hermite product rules have also been used with a limited amount of success. Here the idea is to exploit the higher accuracy which these rules enjoy with one-dimensional singular integrals despite the singularity reducing the effective order of the quadrature. For this approach to be of any use, the integral in question has to be translated into iterated integrals to which the relevant Gauss-Legendre and Gauss-Hermite method can be applied.

In the variational method by C. Schwartz, every integral is treated as a special and separate case. Here a given integral such as,

$$I = \int \frac{1}{W} \quad , \tag{3.1.1}$$

is transformed into a functional of the form,

$$J = 2\int \phi - \int \phi W \phi \quad . \tag{3.1.2}$$

By a careful choice of basis function $U_n(x)$, ϕ is expressed as,

$$\phi = \sum_n C_n U_n \quad . \tag{3.1.3}$$

Hence by variational principle,

$$\delta J = 2 \int \delta\phi [1 - W\phi] \ , \tag{3.1.4}$$

(3.1.4) vanishes for $\phi = \frac{1}{W}$,

and the stationary value of J is then given as,

$$J(\phi = \frac{1}{W}) = \int \frac{1}{W} = I \quad . \tag{3.1.5}$$

Hence,

$$J(\phi = \frac{1}{W} + \Delta) = I - \int \Delta W \Delta \tag{3.1.6}$$

By expressing $M_{n'}$ as,

$$M_{n'} = \int U_n W U_{n'} \tag{3.1.7}$$

and

$$r_n = \int U_n \tag{3.1.8}$$

C. Schwartz established that $\frac{\delta J}{\delta C_n} = 0$ (3.1.9)

(3.1.9) implies that,

$$I = \sum_n c_n r_n \quad . \tag{3.1.10}$$

This method is cumbersome and restrictive. It may be available to only the specialist and can never be used as a black box. So far the most successful approach is the extension of the Romberg method of numerical integration to multi-dimension by E.B. Anders (1966). In his Ph.D. thesis Anders showed that an n-dimensional integral could be approximated in a manner similar to the approach by Romberg in one dimension.

These methods above have their shortcomings. While the work of T.S. Sag and G. Szekeres can be easily extended to higher dimensions the accuracy is poor. The Gauss-Legendre and Gauss-Hermite product rules require special care to translate a given integral into suitable form especially where variable limits and singular points and lines need to be handled. Romberg in high dimension enjoys some measure of success but it requires a high number of function evaluations for any accuracy to be achieved. It is the desire to overcome some of these short-comings above that has led to the work presented here.

In Chapter 2 the tanh and error functions were shown as very good transformations for dealing with singular integrals. They do this by spreading the integrands over the infinite interval where their values fall off rapidly away from the origin. In addition they also smooth the integrand sufficiently to make the trapezoidal rule an efficient integrator because the vanishing end point derivatives make the rule high order from the Euler Maclaurin summation formula. It is the success associated with these properties that has motivated us to investigate how these capacities translate into domains of more than one dimension.

In this investigation the aim is to demonstrate that the tanh and error transformations are capable of dealing with singularities in 2-dimensional integrals and producing accurate results.

3.2 THE FORMULATION OF SINGULAR QUADRATURE IN 2-DIMENSIONS - tanh RULE

Consider the integral,

$$I_2 = \int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dxdy \;, \tag{3.2.1}$$

For other finite ranges of integration, transform into the interval (-1,1) by using the relation:

$$S = \frac{(b-a)t}{2} + \frac{b+a}{2} \tag{3.2.2}$$

for each variable, where $S \in (a,b)$ and $t \in (-1,1)$. Using the transformations,

$$x = \tanh \alpha^n \;, \tag{3.2.3}$$

and,

$$y = \tanh \beta^n \;, \tag{3.2.4}$$

and differentiating (3.2.3) and (3.2.4) we have,

$$dx = \operatorname{sech}^2\alpha^n . n . \alpha^{n-1} d\alpha \tag{3.2.5}$$

$$dy = \operatorname{sech}^2\beta^n . n . \beta^{n-1} d\beta \quad . \tag{3.2.6}$$

Substituting (3.2.3), (3.2.4), (3.2.5) and (3.2.6) in (3.2.1) we have,

$$I_2 = n^2 \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f[\tanh\alpha^n, \tanh\beta^n]\,(\alpha\beta)^{n-1} \operatorname{sech}^2\alpha^n . \operatorname{sech}^2\beta^n \; d\alpha d\beta \tag{3.2.7}$$

This equation is now transformed to polar coordinates by putting

$$\alpha = r\cos\theta \quad \text{and} \quad \beta = r\sin\theta \;, \tag{3.2.8}$$

and substituting (3.2.8) and the Jacobian of the transformation into (3.2.7). Hence,

$$I_2 = n^2 \int_0^\infty\int_0^{2\pi} f[\tanh(r\cos\theta)^n, \tanh(r\sin\theta)^n](r^2\sin\theta\cos\theta)^{n-1}$$
$$\operatorname{sech}^2(r\cos\theta)^n \times \operatorname{sech}^2(r\sin\theta)^n \times r\,dr\,d\theta \qquad (3.2.9)$$

If we put,

$$f[\tanh\alpha^n, \tanh\beta^n] = g(\alpha,\beta)\ , \qquad (3.2.10)$$

and substitute (3.2.10) into (3.2.9) we have,

$$I_2 = n^2 \int_0^\infty\int_0^{2\pi} g(\alpha,\beta)(r^2\sin\theta\cos\theta)^{n-1}\operatorname{sech}^2(r\cos\theta)^n\operatorname{sech}^2(r\sin\theta)^n r\,dr\,d\theta$$
$$(3.2.11)$$

(3.2.11) can be simplified by setting,

$$G(\alpha,\beta) = g(\alpha,\beta)\alpha^{n-1}\beta^{n-1}\operatorname{sech}^2\alpha^n\operatorname{sech}^2\beta^n\ . \qquad (3.2.12)$$

Hence (3.2.11) becomes,

$$I_2 = n^2 \int_0^\infty\int_0^{2\pi} G(r\cos\theta, r\sin\theta)r\,dr\,d\theta \qquad (3.2.13)$$

From (3.2.12), $\operatorname{sech}^2\alpha^n\operatorname{sech}^2\beta^n$ will be very small away from the origin. This implies that the transformed integrand will fall off rapidly and all the higher derivatives will also be small away from the origin. Hence just as in one dimension the trapezoidal rule will work well in the infinite range. So applying the trapezoidal rule to (3.2.13) from 0.0 to some limit M in m steps where the step size $h = \frac{M}{m}$ gives

$$I_2 = n^2 \left\{ h \sum_{i=1}^{m-1} \int_0^{2\pi} G(ih\cos\theta, ih\sin\theta)ih\,d\theta + \frac{h}{2}\int_0^{2\pi} G(mh\cos\theta,\right.$$
$$\left. mh\sin\theta)mh\,d\theta \right\} \qquad (3.2.14)$$

Finally, applying Clenshaw Curtis to (3.2.14) on the range $\int_0^{2\pi}$ gives the new rule. In this way the efficiency of the trapezoidal and Clenshaw Curtis rules is exploited to give an efficient integrator for 2-dimensional singular integrals.

3.3 IMPLEMENTATION OF THE tanh RULE

The resultant rule from (3.2.14) is applied with odd values of $n=1,3$ and the integration is carried out along concentric circles as we step out from the origin until further contribution to the quadrature is insignificant. Each Clenshaw application yields an accurate quadrature round a circle in a progressive manner. An initial step size h (say 0.5) is taken and steps from 0.0 are made until further contribution is no longer significant. This initial step then fixes the upper limit of summation and yields an initial estimate of the quadrature. Intermediate points are then injected into the underlying rule in a progressive manner increasing the accuracy until convergence is obtained. This involves new Clenshaw Curtis applications to intermediate circles. (See Appendix (iii) for the computer program used).

3.4 ERROR ESTIMATE OF THE TANH RULE

The error in using (3.2.14) as an integrator of two-dimensional singular integrals is due to the following:

(a) The error introduced as a result of using Clenshaw

(b) The error inherent in the use of the transformation method and the error due to the truncated use of the trapezoidal rule.

Applying Clenshaw Curtis to (3.2.14) the result can be represented formally as,

$$I = n^2 \int_0^\infty \{Q_N(r) + E_N(r)\}dr \quad . \tag{3.4.1}$$

Using an m point trapezoidal rule on (3.4.1) gives,

$$I = q_m + e_m + n^2 mE_n \quad , \tag{3.4.2}$$

So the error E_2 of applying the new rule is,

$$E_2 = e_m + n^2 mE_N \quad . \tag{3.4.3}$$

Clenshaw and Curtis showed that if an integrand can be expanded in an infinite Chebyshev series as,

$$F(t) = \sum{}' A_r T_r(t) \quad , \tag{3.4.4}$$

where $\sum{}'$ denotes a summation in which the final term is halved, then the error E_N is defined as,

$$E_N = I - I_N = \int f - Q_N \quad , \tag{3.4.5}$$

(Q_N is the approximation by Clenshaw Curtis rule) is given as,

$$E_N = \sum_{r=0}^{\frac{1}{2}N-1} \frac{2A_{2N-r}}{4r^2-1} + \sum_{r=0}^{\frac{1}{2}N-1} \frac{2A_{2N+2r}}{4r^2-1}$$

$$- \sum_{r=1}^{N-1} \frac{2A_{N+2r}}{(N+2r+1)(N+2r-1)} \quad . \tag{3.4.6}$$

By assuming that N is even they showed that (3.4.6) can be simplified to take the form,

$$E_N = \frac{16.1.N}{(N^2-1^2)(N^2-3^2)} A_{N+2} + \frac{16.2.N}{(N^2-1)(N^2-5^2)} A_{N+4}$$

$$+ \frac{16(N/2-1)N}{3(2N-1)(2N-3)} A_{2N-2} - (2 + \frac{2}{4N^2-1})A_{N+2}$$

$$+ (\frac{2}{3} - \frac{2}{(2N+1)(2N+3)})A_{2N+2} + \cdots \tag{3.4.7}$$

From the above Clenshaw and Curtis suggested two methods of error estimate. For integrands whose Chebyshev expansion converges quickly they stated that the error estimate should be expressed as,

$$E_N = \max\left[\frac{|a_N|}{4(N+1)}, \frac{1}{32(N-1)} |2a_{N-2}-a_N|, \frac{1}{128(N-3)}|a_{N-4}-a_{N-2}|\right] \tag{3.4.8}$$

where the quantity a_r is defined as,

$$a_{N-2r} = \frac{2}{N} \sum_{s=0}^{N} (-1)^s F(\cos \frac{s\pi}{N}) \cos \frac{2rs\pi}{N} \tag{3.4.9}$$

and for an integrand whose Chebyshev expansion converges slowly E_N is bounded as shown below,

$$E_N < 2k_N/N \quad , \tag{3.4.10}$$

where the value of k_N is shown in Table C.

N	4	6	8	12	16	24	32	48	64	96	128	192
k_N	0.28	0.12	0.14	0.21	0.24	0.28	0.28	0.28	0.29	0.29	0.29	0.29

TABLE C

H. O'Hara and F.J. Smith gave other error estimates arising from (3.4.7). For an integrand where the singularity is far from ±1 they expressed the error bound as,

$$|E_N| < \frac{32N}{(N^2-1)(N^2-9)} |A_{N+2}| \quad . \tag{3.4.11}$$

By also expanding the integrand in Chebyshev series of the second kind, they arrived at an estimate equivalent to (3.4.8). But of relevance to this work is their third error estimate. This estimate is based on the assumption that the integrand is continuous and differentiable. Hence they stated that at worst the coefficients A_r in (3.4.7) falls as $1/r^2$. On equating $A_r=k_N/r^2$ for r>N, summing up the right hand side of (3.4.7) to infinity and putting,

$$\tfrac{1}{2}|a_N| = (k_N/N^2)[1+3^2+5^2 \ldots] \tag{3.4.12}$$

they established a new error estimate written as,

$$|E_N| < C_N|a_N| \quad , \tag{3.4.13}$$

where the values of C_N are given in Table D.

N	4	8	16	32	64	128	256
C_N	0.586	0.628	0.646	0.654	0.658	0.660	0.662

TABLE D

When $|a_N|$ is very small, then $|E_N|$ takes the form,

$$|E_N| = C_N \max[|a_N|, 2|a_{N-2}|, |I_N - I_{N/2}|] \qquad (3.4.14)$$

From (3.4.2) $n^2 m E_N$ can be estimated roughly since E_N the error due to the use of Clenshaw Curtis can be estimated using (3.4.13) or (3.4.14) for small a_N.

e_m is difficult to estimate analytically and can be done if $Q_N(r)$ in (3.4.1) which is the result of applying Clenshaw Curtis can be represented as a function of r. If this is possible the error e_m may be estimated by the error analysis approach due to H. Takahasi and M. Mori (1970).

3.5 THE FORMULATION OF A SINGULAR QUADRATURE RULE IN 2-DIMENSIONS - ERROR FUNCTION TYPE OF RULE

In this paragraph integrals of the form,

$$I_2 = \int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dxdy \quad , \tag{3.5.1}$$

are considered.

As in (3.2.1) any other finite interval of integration can be transformed into the interval (-1,1) by,

$$S = \frac{(b-a)t}{2} + \frac{b+a}{2} \quad , \tag{3.5.2}$$

for each variable where $S \in (a,b)$ and $t \in (-1,1)$. Again put,

$$x = g(u) \tag{3.5.3}$$

and $$y = g(z) \quad , \tag{3.5.4}$$

where the function $g(u)$ is defined now as,

$$g(u) = C\int_0^u e^{-w^{2n}}\,dw \quad , \tag{3.5.5}$$

Differentiating (3.5.3) and (3.5.4) we have,

$$dx = Ce^{-u^{2n}}\,du \; , \tag{3.5.6}$$

$$dy = Ce^{-z^{2n}}\,dz \; , \tag{3.5.7}$$

Substituting (3.5.3),(3.5.4),(3.5.6) and (3.5.7) in (3.5.1) gives,

$$I_2 = C^2\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f[g(u),g(z)]e^{-u^{2n}}\,e^{-z^{2n}}\,dudz \; . \tag{3.5.8}$$

Adopt the substitution,

$$u = r\cos\theta \tag{3.5.9}$$

and $$z = r\sin\theta \quad . \tag{3.5.10}$$

Differentiating (3.5.9) and (3.5.10) and substituting in (3.5.8), we have,

$$I_2 = C^2 \int_0^{2\pi}\int_0^{\infty} f[g(r\cos\theta), g(r\sin\theta)] e^{-(r\cos\theta)^{2n}} e^{-(r\sin\theta)^{2n}} r\,dr\,d\theta \tag{3.5.11}$$

put, $\quad f[g(u), g(z)] = b(u,z)$. (3.5.12)

Then (3.5.11) becomes,

$$I_2 = C^2 \int_0^{2\pi}\int_0^{\infty} b(u,z) e^{-(r\cos\theta)^{2n}} e^{-(r\sin\theta)^{2n}} r\,dr\,d\theta \ . \tag{3.5.13}$$

If we put,

$$B(u,z) = b(u,z) e^{-u^{2n}} e^{-z^{2n}} \ , \tag{3.5.14}$$

Hence (3.5.1) becomes,

$$I_2 = C^2 \int_0^{2\pi}\int_0^{\infty} B(r\cos\theta, r\sin\theta) r\,dr\,d\theta \ . \tag{3.5.15}$$

From (3.5.13) $e^{-(r\cos\theta)^{2n}} e^{-(r\sin\theta)^{2n}}$ becomes very small away from the origin and hence the value of the transformed integrand falls off even more rapidly than the situation in (3.2.11). Again the efficiency of the trapezoidal rule under this situation is exploited. So applying the trapezoidal rule on r from 0.0 to some limit M in m steps where the step-size $h = \frac{M}{m}$, to (3.5.15) we have,

$$I_2 = C^2 \left\{ h \sum_{i=1}^{m-1} \int_0^{2\pi} B(ih\cos\theta, ih\sin\theta) ih\,d\theta \ + \right.$$

$$\left. \frac{h}{2} \int_0^{2\pi} B(mh\cos\theta, mh\sin\theta) mh\,d\theta \right\} \tag{3.5.16}$$

The Clenshaw-Curtis rule is finally applied to (3.5.16) to give the new rule. So as in (3.2.14) a judicious combination of two standard rules is used to produce an efficient integrator for 2-dimensional singular integrals.

3.6 IMPLEMENTATION OF THE ERROR FUNCTION TYPE OF RULE

The integrator derived from (3.5.16) is implemented in exactly the same way as in (3.2.14). The integration is carried out along concentric circles continuing until further contributions to the quadrature are insignificant. The rule can only be used for the function $g(u) = C\int_0^u e^{-w^{2n}}\,dw$ for values of n=1 and 2. This is because the values of the integrand falls off so rapidly that it is impossible to use the rule for higher values of n. Clenshaw-Curtis is applied progressively and it is found that a maximum of 64 points is sufficient even in difficult problems.

3.7 ERROR ESTIMATES

The error associated with the integrator in (3.5.16) is similar to that in (3.2.14). It is made up of a combination of the inherent error of the transformation methods, the error due to the approximation by Clenshaw-Curtis and the error due to the application of the truncated trapezoidal rule. So the application of Clenshaw-Curtis to (3.5.15) can be represented formally as,

$$I_2 = C^2 \int_0^\infty [B_N(r) + E_N(r)]dr \quad . \tag{3.7.1}$$

Applying an m point trapezoidal rule to (3.7.1) we have,

$$I_2 = b_m + e_m + C^2 mE_N \quad . \tag{3.7.2}$$

Hence $(e_m + C^2 mE_N)$ the error of this quadrature rule can be analyzed as in (3.4).

3.8 THE GRID METHOD OF EVALUATING 2-DIMENSIONAL SINGULAR INTEGRALS

The integral I_2 in (3.2.7) was defined as,

$$I_2 = n^2 \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f[\tanh\alpha^n, \tanh\beta^n](\alpha\beta)^{n-1}\text{sech}^2\alpha^n . \text{sech}^2\beta^n d\alpha d\beta \quad (3.8.1)$$

As indicated in §3.2 the values of the integrand in (3.8.1) will fall off in the infinite plane as one moves away from the origin. So instead of applying another transformation as in §3.2 and §3.5 the trapezoidal product rule is applied to (3.8.1) to give accurate results.

3.9 IMPLEMENTATION OF THE GRID APPROACH

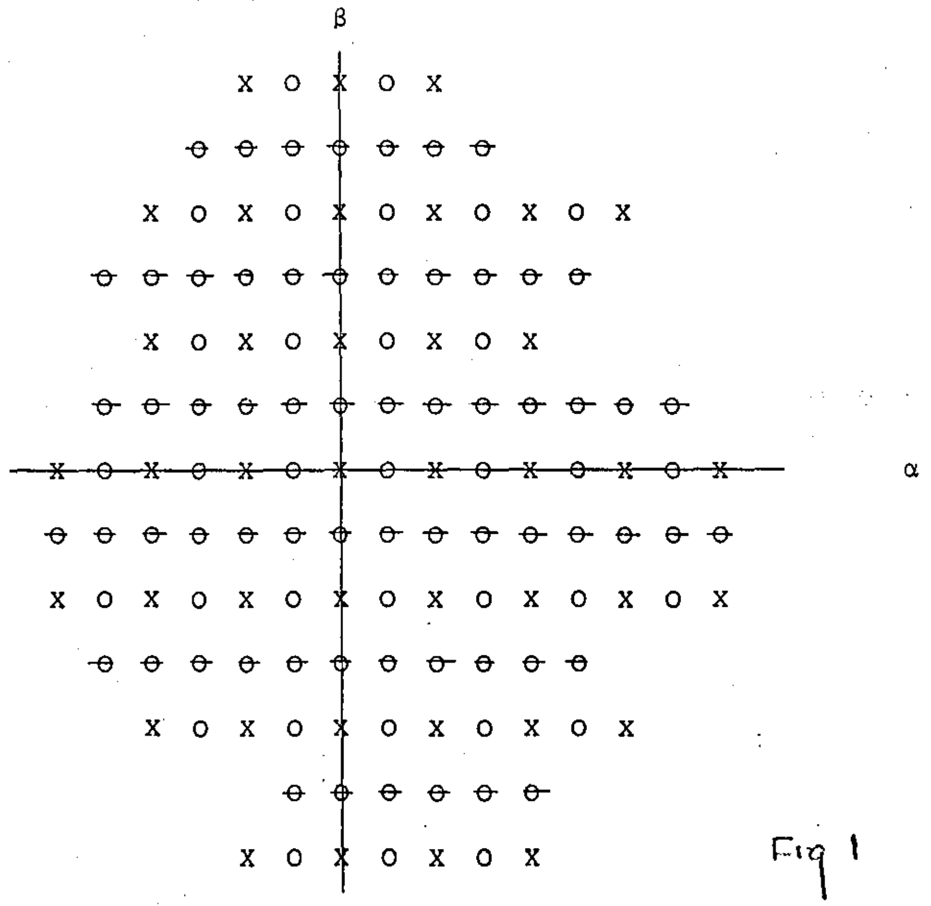
To evaluate (3.8.1) using a trapezoidal product rule, it was necessary to establish the extent of a grid in the infinite plane, the boundary of which is determined by careful monitoring of the values of the integrand. The grid was set up by analogy with the implementation of the tanh rule transformation in one-dimension.

An initial step size h (say 0.4) was used to step along the α axis for a given β. Steps were taken until the values of the integrand fall below a significant value from two successive steps (to account for zeros in the integrand). The process was carried out both towards $+\infty$ and $-\infty$ and β increased outwards from the origin in both directions. The whole process is terminated when the initial point gives no contribution so determining the end point in the β direction.

The trapezium rule in 2-dimensionsgives weights as shown in the diagram below for a uniform grid (Fig. 1) and as there is no edge contribution, we can evaluate the integrals as,

$$I_2 = \frac{h_1 \times h_2}{4} [4\sum f_i] \quad , \qquad (3.9.1)$$

and hence the integral can be accumulated simultaneously with the search pattern as is the case in 1-dimension. The final grid has an irregular outline as shown overleaf.



X indicates points with the initial step size

O indicates intermediate points between initial steps

-θ- indicates the points along α-line search for mid-point β values.

A subdivision scheme is then implemented with new points progressively added to the lower order rule. There are two types of added point. Firstly along existing β values the mid-points are simply added in and secondly the mid-point β values require a whole new α-line search at the fine grid level. Hence convergence to a given accuracy can be obtained. Appendix (iv) is the computer program for this scheme.

3.10 RESULTS AND CONCLUSIONS

In general the new rules perform very well. However it is less easy to achieve a required accuracy by successive doubling of points than in one-dimension as the number of points used will grow impractically after one doubling. In the tanh product rule the underlying grid is fixed in the initial choice of h and the doubling in each dimension yields the results quoted in Table G. The tanh product rule is most efficient with the values of n=3 and 5; the doubling problem is less marked with the Clenshaw-Curtis and trapezium rule combination as each integral along a circle is accurately evaluated and no more extra points are then needed if sub-division in the trapezium rule takes place. As in one-dimension, increasing n to 7 or above makes the integrand very steep sided and accuracy falls off.

I_6 and I_7 are different from the earlier examples having singular derivativesalong a curve in the range of integration. As one might expect a trapezium product rule with no transformation performs quite well in that a 64×64 rule yields 1.864304 for I_6 and 0.532604 for I_7. The error depends on f" rather than some higher singular derivative. The Clenshaw-Curtis, trapezium rule combination gives very good accuracy for these examples as the tanh transformation enhances the effectiveness of the trapezium rule because it is operating on a set of Clenshaw-Curtis quadratures which are well-known for coping with derivative singularities.

INTEGRALS	POINTS OF SINGULARITIES	ANALYTIC ANSWER	ROMBERG		TANH FUNCTION (CLENSHAW-CURTIS AND TRAPEZOIDAL RULE	
			ANSWERS	NO.OF FUNCTION EVALUATION	ANSWERS	NO.OF FUNCTION EVALUATION
$I_1 \int_0^1\int_0^1 \frac{dxdy}{1-xy}$	x=y=1	1.644934	1.6446286	2^{18}	1.64491078	3840
$I_2 \int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{1-x^2y^2}}$	x=y=±1	4.3551723	4.3547947	32060	4.35516707	3840
$I_3 \int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{2-x-y}}$	x=y=1	3.12419433	3.124216	32060	3.1241953	3840
$I_4 \int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{3-x-2y}}$	x=y=1	2.5790076	2.5792585	32060	2.5790076	8153
$I_5 \int_0^1\int_0^1 (xy)^{-\frac{1}{2}}dxdy$	Along lines x=y=0	4	3.9424276	32060	4.0004645	6786
$I_6 \int_{-1}^1\int_{-1}^1 \lvert x^2+y^2-0.25\rvert dxdy$	Along the circle of radius .5	1.8630162	1.8630151	$2^{12}.3^2$	1.86301629	2944
$I_7 \int_0^1\int_0^1 \lvert x-y\rvert^{\frac{1}{2}}dxdy$	Along x=y	.5333333	.533300	2^{14}	.533333369	7119

TABLE E

INTEGRALS	POINTS OF SINGULARITIES	ANALYTIC ANSWER	ROMBERG		ERROR TYPE OF FUNCTION (CLENSHAW-CURTIS AND TRAPEZOIDAL RULE	
			ANSWERS	NO.OF FUNCTION EVALUATION	ANSWERS	NO.OF FUNCTION EVALUATION
$I_1 \int_0^1\int_0^1 \frac{dxdy}{1-xy}$	x=y=1	1.644934	1.6446286	2^{18}	1.64490010	26980
$I_2 \int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{1-x^2y^2}}$	x=y=±1	4.3551723	4.3547947	32060	4.3551086	27474
$I_3 \int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{2-x-y}}$	x=y=1	3.12419433	3.124216	32060	3.12418592	70332
$I_4 \int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{3-x-2y}}$	x=y=1	2.5790076	2.5792585	32060	2.5790047	69804
$I_5 \int_0^1\int_0^1 (xy)^{-\frac{1}{2}}dxdy$	Along lines x=y=0	4	3.9424276	32060	3.97345200	4032
$I_6 \int_{-1}^1\int_{-1}^1 \|x^2+y^2-0.25\|dxdy$	Along the circle of radius .5	1.8630162	1.8630151	$2^{12}.3^2$	1.86301634	3904
$I_7 \int_0^1\int_0^1 \|x-y\|^{\frac{1}{2}}dxdy$	Along x=y	.53333333	.533300	2^{14}	.533333942	3870

TABLE F

INTEGRALS	ANALYTIC ANSWERS	ROMBERG	GAUSS-LEGENDRE PRODUCT RULE	TANH PRODUCT RULE		
				n=1	n=3	n=5
$I_1 \int_0^1\int_0^1 \frac{dxdy}{1-xy}$	1.644934	1.644628 (2^{18})	1.644741 (4096)	1.6449328 (8213)	1.64493325 (3597)	1.64493744 (1975)
$I_2 \int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{1-x^2y^2}}$	4.3551723	4.3547949 (32060)	4.355164 (4096)	4.35517223 (88009	4.35517228 (3653)	4.35517271 (1973)
$I_3 \int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{2-x-y}}$	3.12419433	3.124216 (32060)	3.124191 (4096)	3.12419072 (7660)	3.12419385 (3535)	3.12419418 (1958)
$I_4 \int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{3-x-2y}}$	2.5790076	2.5792585 (32060)	2.579004 (4096)	2.57900488 (7558)	2.57900685 (3514)	2.57900709 (1956)
$I_5 \int_0^1\int_0^1 (xy)^{-\frac{1}{2}}dxdy$	4	3.9424276 (32060)	3.946186 (4096)	3.99999562 (8864)	3.99999899 (4532)	3.99999928 (3542)
$I_6 \int_{-1}^1\int_{-1}^1 \lvert x^2+y^2-0.25\rvert dxdy$	1.8630162	1.8630151 $(2^{12}.3^2)$	1.862980 (4096)	1.8630477 (10057)	1.86303155 (4453)	1.86393924 (1961)
$I_7 \int_0^1\int_0^1 \lvert x-y\rvert^{\frac{1}{2}}dxdy$	.53333333	.533300 (2^{14})	.532197 (4096)	.532373518 (10057)	.530764960 (3441)	.52827057 (2751)

*The enclosed figures represent the number of function evaluations

TABLE G

3.11 THE TRANSFORMATION METHOD FOR 2 DIMENSIONAL NON-SINGULAR INTEGRALS - INTRODUCTION

Quadrature for a multidimensional integral is not a straightforward affair. In one dimension it is usually sufficient to develop rules for standard intervals $[0,1]$, $[0,\infty]$ and $[-\infty,\infty]$ since the transformation which takes intervals into one of these leaves almost all the properties of a quadrature unchanged. In higher dimension the equivalent of such transformations are difficult to come by and there is the need to develop rules for each affine class of region. The earliest success in higher dimensions was recorded with simple regions like the n-dimensional cube, ball, spheres and simplex. In general it will be necessary to transform a given region into one of these special regions and this may be non-trivial. As a general procedure any integration region may be enclosed in one for which formulae have been developed. For example any bounded region may be enclosed in a cube and the integration treated by extending the definition of the integrand to the whole enclosing region by defining it as zero on the compliments of the original region. Good as this seems, the modification introduces discontinuity which is a problem in its own right.

One of the most successful approaches over the n-dimensional unit cube and in general n-dimensional integral is to regard the integral as an n-fold iterated integral and apply a 1-dimensional quadrature formula for each variable separately. Thus if $Q_1, Q_2, \ldots, Q_n$ are quadrature formulae for the interval $[0,1]$, we might write the integral to be evaluated as,

$$\int_{R_n} f = \int_0^1 \cdots \int_0^1 f(x_1,x_2,\ldots,x_n)dx_1,\ldots,dx_n \ . \qquad (3.11.1)$$

Applying Q_1 to (3.11.1) we have,

$$\int_{R_n} f \approx \int_0^1 \cdots \int_0^1 Q(f,x_1)dx_2,\ldots,dx_n \ . \qquad (3.11.2)$$

By successive application of $Q_2,\ldots,Q_n$ we have,

$$\int_{R_n} f \approx Q_n(Q_{n-1}(\ldots Q_1(f,x_1)x_n) \ . \qquad (3.11.3)$$

The R.H.S. of (3.11.3) can be represented by $Q=Q_1 \cdot Q_2 \cdots Q_n$ a cartesian product since the set of points at which f is evaluated in (3.11.3) is the Cartesian product of the set of evaluation points of the Q_i formula. The precision of Q is the minimum of $\{d_1,d_2,\ldots,d_n\}$ where for each i, d_i is the degree of precision of Q_i.

The Cartesian product technique can be modified to produce quadrature formulae for regions other than the cube. Pierce (1957) showed that it can be applied to the ball and spherical shells, while Hammer, Marlow and Stroud (1956) applied it to cones and simplexes. In these extensions the contributors demonstrated that it was necessary to use the 1-dimensional quadrature formulae with respect to certain weight functions. They further established that the maximum error estimate of the product rule is given by the expression,

$$\left|\int_{R_n} f - (\prod_i^n Q_i)f\right| \leq E_1 + A_1E_2 \ldots + A_{n-1}E_n \ , \ * \qquad (3.11.4)$$

and that if for a given M,

$$\frac{\partial^{\ell} f}{(\partial x_i)^{\ell}} \leq M \ , \ i=1,2,\ldots,n \ , \qquad (3.11.5)$$

* where A_i are the weights and E_i are error bounds for each dimension

throughout R_n such that f is uniformly continuous in x_i's for $Q_1=Q_2,\ldots,=Q_n$ (3.10.4) becomes,

$$\left|\int_{R_n} - Q^n f\right| \leqslant (1 + A + A^2 \ldots + A^{n-1})kn^{-\ell} . \qquad (3.11.6)$$

N.S.Bakhvalov (1965) established the lower bound for the quadrature error of Q over a region $r \in E_n$ for a family of functions and showed that there exists a C such that,

$$\left|\int_{R_n} - Qf\right| > C . \qquad (3.11.7)$$

The product rule is efficient but the number of function evaluations increases rapidly with increase in dimension. So other competitive methods have to be devised. Among the rules devised to reduce the number of function evaluations are the minimum point rule and the associated symmetric rules which are referred to in more detail in Chapter 4.

Another approach for constructing rules for multiple quadrature that is aimed at reducing function evaluationsis the one pioneered by Bakhvalov. By Bakhvalov's theorem the error in approximating an integral by a quadrature can be minimised and the resultant rule is called the minimum-norm formula. He further established that the minimum error in such a formula is bounded by $kN^{-\ell/s}$ when an N point formula is used. In the main he considered a function which is square integrable in $R \in E_n$ and where the ℓ derivative exists. By defining an inner product as,

$$(f.g) = \int_R \sum_{|\underline{i}|=\ell} f^{(\underline{i})} g^{(\underline{i})} \qquad (3.11.8)$$

where $|\underline{i}|=\ell$ indicates a sum over all the ℓ^{th} derivatives of f, with the norm,

$$||f||_{\ell_2} = ||f.f||^{\frac{1}{2}} = \left\{ \int_{R} \sum_{|\underline{i}|=\ell} f^{(\underline{i})^2} \right\}^{\frac{1}{2}} \qquad (3.11.9)$$

he defined a Hilbert space $C^s_{\ell,2}(R)$. The quantity,

$$E_Q(f) = \int_{R_i} f - Q(f) \quad , \qquad (3.11.10)$$

defines a linear functional on $C^s_{\ell,2}(R)$ and,

$$|E_Q| = \sup_{f \in C^s_{\ell,2}(R)} \frac{|E_Q(f)|}{||f||} \quad , \qquad (3.11.11)$$

defines the norm of the functional E_Q. If E_Q is not infinite then E_Q is a bounded functional and its value is known. Then the simple error bound is given as,

$$|\int_R f - Qf| \leq ||E_Q||.||f||_{\ell,2} \quad . \qquad (3.11.12)$$

If Q is an N point formula Bakhvalov proved that,

$$||E_Q|| > kN^{-\ell/s} \quad , \qquad (3.11.13)$$

for some k.

So the object here is to make $||E_Q||$ as small as possible. He then posed this problem in two ways viz:

(a) Given N points $x_1,x_2,\ldots,x_N$ in E_N, determine $a_1,a_2,\ldots,a_N$ such that the quadrature formula Q(f) defined by,

$$Q(f) = \sum a_r f(x_r) = \int_{R_n} f \quad , \qquad (3.11.14)$$

has as small as possible values of $||E_Q||$.

(b) Determine $x_1,x_2,\ldots,x_N$ and $a_1,a_2,\ldots,a_N$ such that (3.11.14) has $||E_Q||$ as small as possible.

The solution to (a) gives the minimum-norm formula while the solution to (b) is called a relative minimum norm formula. Sard and Sobolov outlined the main elements of the theory and set out the procedure for solving (3.11.14) as follows. They stated that the error in (3.11.14) could be written as,

$$E_Q(f) = \int_{R_n} f(x)L(x)\,dx \ , \qquad (3.11.15)$$

where,

$$L(x) = \mathcal{E}_R(x) - \sum_{r=1}^{N} a_r \delta(x-x_r) \ . \qquad (3.11.16)$$

Here $\mathcal{E}_R$ is the characteristic function of the region R_n and δ is the Dirac delta - the generalised function with the property that

$$\int_R f(x)\delta(x-x_0)\,dx = f(x_0) \ , \qquad (3.11.17)$$

for any f in the space under consideration. Given a Q, quadrature formula of degree $\geq \ell-1$, from Sobolov's theorem they established that E_Q is bounded if $\ell > s/2$. So to determine $||E_Q||$ is to look for a function f_0 in $C^s_{\ell,2}$ such that,

$$||E_Q(f_0)||/||f_0||_{\ell,2} = \sup||E_Q(f)||/||f||_{\ell,2} \qquad (3.11.18)$$

From the above they established that finding f_0 is equivalent to finding $g_0 \in C^s_{\ell,2}$ which minimizes,

$$H(g) = [g,g] + 2\int_R gL \qquad (3.11.19)$$

(3.11.19) is a variational problem for if g_0 minimises H, then for any $f \in C^s_{\ell,2}$,

$$\frac{d}{dt}H(g_0 + tf)|_{t=0} = 0 , \qquad (3.11.20)$$

(3.11.20) is exactly,

$$[g_0,f] = \int_R fL-E_Q(f) , \qquad (3.11.21)$$

Sobolov by expressing g_0 as a convolution established that,

$$g_0(x) = \int_{E^s} G(x-y)L(y)dy , \qquad (3.11.22)$$

when,

$$G(x) = k||x||^{2\ell-s} \times \begin{cases} 1 & s \text{ odd} \\ \log||x|| & s \text{ even} \end{cases} ; \qquad (3.11.23)$$

as the solution of the equation $\Delta^\ell q$ with $k=k(\ell,s)$ being a constant.

Then,

$$||E_Q|| = ||g_0||_{\ell,2} = ||E_Q(g_0)||^{\frac{1}{2}} . \qquad (3.11.24)$$

Babushka (1963) showed that in the situation where $(x_1,x_2,\ldots,x_n)$ were given for $n \geq \binom{s+1}{\ell}$ the so-called relative minimum norm formula has the property that its extreme function will take on at the point $(x_1,x_2,\ldots,x_N)$ the same values as some polynomial,

$$\sum_{|i|\leq\ell-1} b_i x^\ell \qquad (3.11.25)$$

This ensures that the L.H.S. of (3.11.22) is equal to (3.11.25). Hence the problem in (a) is reduced to solving a system of N linear algebraic equations. For large N the procedure is different. This procedure and that for dealing with the problem in (b) is ignored in this write-up because they are strictly of very little practical importance as well as being tedious to operate.

From the foregoing it is clear that the product rule is still one of the most efficient methods as far as smooth functions are concerned. However this efficiency is hampered by the rapid growth in the number of function evaluations as the dimensions become higher. It is the desire to retain some of the advantages of the product rule and minimise the astronomical growth in the number of function evaluations that has given rise to the search for a good transformation method. In the numerical evaluation of high-dimensional integrals, T.W. Sag and G. Szekeres proposed the following series of transformations,

$$x = \tfrac{1}{2}(1+t)$$
$$t = \tanh u/(1-u^2)$$
$$u = 2y-1$$

which maps the interval (0,1) into itself. They established that with smooth functions the trapezoidal rule was more efficient without such a transformation. This inefficiency is due to the violent influence of the transformation near the end points of the interval before the integrand and the derivatives vanish at the end points.

In this investigation we are proposing a polynomial transformation with some of the properties above with in-built tuning capacity.

3.12 THE FORMULATION OF THE TRANSFORM TRAPEZOIDAL PRODUCT RULE

Consider the integral,

$$I_2 = \int_0^1\int_0^1 f(x,y)\,dxdy \quad . \tag{3.12.1}$$

For all other intervals of integration the interval (a,b) is transformed by using the relation,

$$t = (b-a)s + a \quad , \tag{3.12.2}$$

for $t \in (a,b)$ and $s \in (0,1)$ for each variable. The polynomial transformation,

$$x = \{1-(1-u^{\ell})^{n}\}^{m} \quad , \tag{3.12.3}$$

and,

$$y = \{1-(1-v^{\ell})^{n}\}^{m} \text{ is now applied.} \tag{3.12.4}$$

This transformation maps the interval (0,1) into itself. By differentiating (3.12.3) and (3.12.4) we have,

$$dx = \ell nm[(1-(1-u^{\ell})^{n}]^{m-1}(1-u^{\ell})^{n-1}u^{\ell-1}du \ , \tag{3.12.5}$$

and,

$$dy = \ell nm[(1-(1-v^{\ell})^{n}]^{m-1}(1-v^{\ell})^{n-1}v^{\ell-1}dv \ . \tag{3.12.6}$$

Substituting (3.12.3), (3.12.4), (3.12.5) and (3.12.6) in (3.12.1) gives,

$$I_2 = \int_0^1\int_0^1 f[(1-(1-u^{\ell})^{n})^{m},(1-(1-v^{\ell})^{n})^{m}]\times(nm\ell)^2$$
$$\left\{(1-(1-u^{\ell})^{n}\times(1-(1-v^{\ell})^{n}\right\}^{m-1}\left\{(1-u^{\ell})\,(1-v^{\ell})^{n-1}(uv)^{\ell-1}\right\}dudv \tag{3.12.7}$$

The integrand and its derivatives vanish for u=0 or 1 and v=0 or 1 for any polynomial of degree $mn\ell$ or less. By putting,

$$g(u,v) = f[(1-(1-u^{\ell})^{n})^{m},(1-(1-v^{\ell})^{n})^{m}] \quad , \tag{3.12.8}$$

and substituting (3.12.8) in (3.12.7) we have,

$$I_2 = \int_0^1\int_0^1 (mn\ell)^2 g(u,v)\left\{(1-(1-u^{\ell})^{n}\times(1-(1-v^{\ell})^{n}\right\}^{m-1}[(1-u^{\ell})\times(1-v^{\ell})]^{m-1} \times (uv)^{\ell-1}dudv \ . \tag{3.12.9}$$

Hence applying an s point trapezoidal product rule to (3.12.9) gives,

$$I_2 = (mn\ell)^2h^2 \sum_{j=1}^{s}\sum_{i=1}^{s} g(u_iv_j)[(1-(1-u_i^{\ell})^{n}\times(1-(1-v_j^{\ell})^{n}]^{m-1} \times[1-u_i^{\ell})\times(1-v_j^{\ell})]^{n-1}\times(u_iv_j)^{\ell-1} \quad , \tag{3.12.10}$$

With contributions from the outer edge of the region being zero.

Since the integrand and its derivatives vanish for u=0 or 1 and s=0 or 1 for the integrand of degree $mn\ell$ or less (3.12.10) gives an $s^2-4(s-1)$ point formula. So for s=9 we have a formula in which the number of function evaluations is 49 instead of the usual 81 in a product rule. Moreover the violent behaviour of the transformation by T.W. Sag and G. Szekeres near the end points is now mellowed down by the polynomial since exponential functions decrease or increase rapidly compared with other functions.

3.13 IMPLEMENTATION

The transformation involved in (3.12.10) is a polynomial and in order to get the desired effect the integrand dictates the value of m,n and ℓ to be used. The harder the integral the higher the values of m,n and ℓ to be chosen. In our experiment the values of $m \gtrsim 7$, $n \gtrsim 8$ and $\ell \gtrsim 2$ were good enough to give comparable accurate results. Hence (3.12.10) is implemented directly with the above caution in mind.

3.14 ERROR ESTIMATE OF THE PRODUCT RULE

As shown in (3.11.6) the error E_n in approximating the integral,

$$I_n = \int_0^1 \int_0^1 f(x_1,x_2,\ldots,x_n)dx_1,dx_2,\ldots,dx_n \quad , \qquad (3.14.1)$$

by a product rule defined as,

$$I \approx (\prod_i^n Q_i)f \quad , \qquad (3.14.2)$$

is given as,

$$E_n = \left|\int_R f - (\prod_i^n Q_i)f\right| \leq (1 + A + A^2 \ldots +A^{n-1})kn^{-\ell} \qquad (3.14.3)$$

where $\frac{\partial^\ell f}{\partial x_i^\ell} \leq k$ for f continuous in each variable and A the sum of the absolute values of the coefficients of $Q=(Q_1,Q_2,\ldots,Q_n)$. Putting $Q_1=Q_2=\ldots =Q_n$ and adopting (3.14.3) we have that the error E_n in using (3.12.10) as an integrator is bounded and defined as,

$$E_2 \leq (1 + A^*)k^*s^{*-\ell} \qquad (3.14.4)$$

where A* is the sum of the absolute values of the coefficients of an n point trapezoidal product rule, s* the number of variables in the function, $Q=Q_1Q_2=Q_1^2$ and k* is defined as,

$$k^* \leq (mn\ell)^2 \frac{\partial^\ell}{\partial u^\ell}(G(u,v)) \quad , \qquad (3.14.5)$$

and,

$$G(u,v) = g(u,v)[(1-(1-u^\ell)^n \times (1-(1-v^\ell)^n]^{m-1} \times$$

$$[(1-u^\ell)(1-v^\ell)]^{n-1} \times (uv)^{\ell-1} \quad . \qquad (3.14.6)$$

3.15 RESULTS AND CONCLUSION

The rule in (3.12.10) was tested on a number of integrals and the results compared with known methods as shown in Table (H). The results show that the new rule is quite competitive and achieves some economy. However for the test integrals considered the other rules are more efficient in that they achieve the accuracy quoted in the table with a maximum number of function evaluations of not more than 10. This notwithstanding the new rule has the advantage of wider applications and greater flexibility because of its in-built tuning capacity.

METHODS	TEST INTEGRALS			
	$\int_0^1\int_0^1 e^{-xy}dxdy$	$\int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{3-x^2-y^2}}$	$\int_{-1}^1\int_{-1}^1 \frac{dxdy}{4+x+y}$	$\int_{-1}^1\int_{-1}^1 \frac{dxdy}{\sqrt{4-x-2y}}$
ANALYTIC ANSWERS	0.7965996	2.6555866	1.0464963	2.0958446
NEW METHOD 49 PTS $s^2-4(s-1)$	0.79659920	2.65599740	1.04649323	2.0958249
GAUSS 9 POINT PRODUCT RULE	0.7965995	2.6514268	1.0464731	2.0953358
COHEN AND GISMALLA	0.7965994	2.6541559	1.0464845	2.0913961
L. JENKINS	0.7965995	2.6556216	1.0464947	2.097067
RADON	0.7965800	2.6472813	1.0453342	2.094937
ALBRECHT AND COLLATZ	0.7965700	2.6472812	1.0463703	2.094441

TABLE (H)

CHAPTER 4

THE CONSTRUCTION OF MINIMUM POINT FORMULAE FOR 2-DIMENSIONAL NON-SINGULAR INTEGRALS

4.1 INTRODUCTION

This chapter is devoted to the construction of minimum point quadrature rules for two-dimensional non-singular integrals. As referred to in the introductory chapter of this thesis, the minimum point procedure is the oldest method of finding formulae which achieves a given precision using the fewest possible points. In the main a minimum point quadrature of the form,

$$\int_{R_n} f \simeq \sum_{r=1}^{n} a_r f(x_r) \quad , \tag{4.1.1}$$

is of degree d if it satisfies,

$$\sum_{r=1}^{n} a_r = \int_{R_n} dx \tag{4.1.2}$$

$$\sum_{r=1} a_r x_r^i = \int_{R_n} x^i dx \quad , \ i=1,2,\ldots,s \tag{4.1.2(1)}$$

$$\sum a_{rs} x_r^i x_s^j = \int_{R_n} x^i x^j \, dx \ , \ i,j=1,2,\ldots,s \tag{4.1.2(2)}$$

$$\vdots$$

Stroud showed that the system of equations in (4.1.2) has no solution with n less than,

$$\begin{pmatrix} s + [\frac{d}{2}] \\ [\frac{d}{2}] \end{pmatrix}$$

He further showed that (4.1.2) can only become a linear system in n unknowns $a_1, a_2, \ldots, a_n$ if we choose x_r in advance in R_n and thus make the coefficient matrix of the linear system non-singular. So for a one point formula of degree of precision 1 the point is the centroid of the domain and the coefficient a measure of the domain of integration. If the domain of integration is not convex its centroid may lie outside it. In general there is no assurance that when (4.1.2) has a solution, the point defined by the solution will lie in the region R_n. This raises serious objections to such formulae because in some cases the function being integrated may not be defined outside the domain of integration. So formulae of this nature may be impractical. In this connection Tha cher (1957) and Stroud (1960) have developed a number of (n+1) point formulae of degree 2 for any s dimensional symmetric region. Although some of the points in their formulae lie outside the region of integration they have one point going for them in that most of the formulae have equal weights which maintain good stability characteristics.

Another aspect in the construction of minimal point formulae is the fairly general method of finding efficient formulae of high degree based on the symmetric properties of certain regions. Here the cube, ball and sphere when centred at the origin, show the property that whenever $(x_1,x_2,\dots,x_n)$ is a part of the domain every point of the form $(\pm x_1 \pm x_2 \dots,\pm x_n)$ is also in the domain. Such a region is fully symmetrical if for any permutation of $1,2,\dots,n$, $g(x_1,\dots,x_n)= g(\pm x_{p(1)},\dots,\pm x_{p(n)})$. By a similar reasoning a quadrature formula is fully symmetrical if it contains the points $(x_1,x_2,\dots,x_n)$ with the associated coefficients and it contains all the points $(\pm x_1,\pm x_2,\dots,\pm x_n)$. For example the formula,

$$\int_{R_2} f = a_1 f(0,0) + a_2 \sum_{r=1}^{4} f(\underline{x}_r) , \qquad (4.1.3)$$

where $x_1, x_2, \ldots, x_4$ are the four vertices of R_2 is fully symmetrical.

Thus if R_n is a fully symmetrical region and Q is a fully symmetrical formula (4.1.2(1)), (4.1.2(3)),(4.1.2(5)),... are automatically satisfied since each side of the equations is identically zero. In addition s(s+1)/2 equations (4.1.2(1)) become equivalent to,

$$\sum a_r (x^1)^2 = \int_{R_n} (x^1)^2 dx \quad , \tag{4.1.4}$$

and each of $\begin{pmatrix} s+1 \\ 4 \end{pmatrix}$ equations of (4.1.2(4)) is equivalent to one of these two sets of equations,

$$\sum a_r (x^1)^2 (x^2)^2 = \int_{R_n} (x^1)^2 (x^2)^2 dx \tag{4.1.5}$$

$$\sum a_r (x^1)^4 = \int_{R_n} (x^1)^4 dx \quad . \tag{4.1.6}$$

Hence in a fully symmetrical situation the condition in (4.1.2) becomes greatly simplified and the number of equations to be satisfied by a formula of degree d becomes independent of the number of variables. Using this approach J.N. Lyness (1965, i,ii,iii,iv,v) produced some families of formulae with odd degrees 2d+1 for G_n, n-dim, octahedron while J. McNamee and F. Stenger (1967) also obtained formulae for G_n and other regions. What is common to the works above is that the final rule generated may incorporate a basic rule or its composite form. The most recent contributions in this direction include the construction of quadrature rules by parameter optimization by A.M. Cohen and D.A. Gismalla (1985) and a note on the papers by Cohen and Gismalla on quadrature formulae for symmetric integrands by L.D. Jenkins (1985). The added novelty of these contributions is that the square of the error of approximating an integral by a given rule is minimised.

For a fully symmetric region Tha cher (1957) in his optimum quadrature formulae in s dimensions generated (4.1.2). By defining a set of m*m diagonal matrices,

$$G = [\sqrt{c_i}\,\delta_{hi}] \quad , \tag{4.1.7}$$

and $$X^{(j)} = [X_i^{(j)}\,\delta_{hi}] \, ,$$

he expressed (4.1.2) as;

$$\mathrm{tr}\left\{G \prod_{j=1}^{s} X^{(j)^{n_j}} G\right\} = I_{n_1,n_2,\ldots,n_s} \tag{4.1.8}$$

with,

$$I_{n_1,\ldots,n_s} = \begin{cases} 0 \text{ if at least one } n_j \text{ is odd} \\ \dfrac{2^s}{\prod_{j=1}^{s}(n_j+1)} \text{ if no } n_j \text{ is odd} \end{cases} \tag{4.1.9}$$

Hence he established that for a second degree formula,

$$\mathrm{tr}\{GG\} = 2^s \tag{4.1.10a}$$

$$\mathrm{tr}\{GX^{(j)}G\} = 0 \tag{4.1.10b}$$

$$\mathrm{tr}\{GX^{(j)}X^{(k)}G\} = \frac{2^s}{3}\,\delta_{jk} \quad . \tag{4.1.10c}$$

He converted (4.1.10) into vector equations by defining $\xi = G\,\epsilon$ where ϵ has all unit elements. So that (4.1.10) becomes

$$\xi^T\xi = 2^s \tag{4.1.11a}$$

$$\xi T \xi_j = 0 \tag{4.1.11b}$$

$$\xi_{jk}^{T}\xi_{k} = \frac{2^{s}}{3}\delta_{jk} \quad . \qquad (4.1.11c)$$

By the orthogonal relationship of the vectors, he established a second degree formula for (4.1.1). For formulae of higher degrees he noted that the condition of orthogonality was not sufficient and that additional conditions of s(s+1)/2 new vectors ξ_{jk} such that $\xi_j^{'}$s are orthogonal to ξ_k but not to ξ while s(s-1)/2 $\xi_{jk}(j\neq k)$ are orthogonal to both sets or else the null vector.

The difference in all these methods is the strategy of solving the resultant equation from (4.1.2). Stroud's method does not guarantee that the solution points will lie in the region of integration. Thatcher's approach is similar to Stroud's and requires considerable amounts of work. The work of Cohen et al is geared mainly at symmetric integrands.

The method that is being proposed in this investigation accepts the basic approach in (4.1.2) but adopts a different strategy in solving the resultant equations. Moreover it introduces an element of consistency into the set of equations (4.1.2) by using an extra set of equations which can in turn ensure that the weight functions are positive and that all the solution points lie in the region of integration.

4.2 FORMULATION OF THE 2-DIMENSIONAL MINIMUM POINT RULE

The approach here is to consider the integration of the function $f(x,y)$ (in two variables) defined in a standard region $-1\leqslant x\leqslant 1$ and $-1\leqslant y\leqslant 1$, i.e.,

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dxdy \quad , \quad -1\leqslant x\leqslant 1 \text{ and } -1\leqslant y\leqslant 1 \tag{4.2.1}$$

The strategy is to force $\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dxdy$ to be equal to $\sum_{i=1}^{n} w_i x_i^{\alpha} y_i^{\beta}$.

Hence,

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dxdy = \sum_{i=1}^{n} w_i x_i^{\alpha} y_i^{\beta} \quad , \tag{4.2.2}$$

for $\qquad f(x,y) = x^{\alpha} y^{\beta}$,

where w_i's are weights $0\leqslant\alpha$, $0\leqslant\beta$, $1<n<\infty$ and α,β and n are integers.

This implies that (4.2.2) becomes,

$$\int_{-1}^{1}\int_{-1}^{1} x^{\alpha} y^{\beta} = C_e \tag{4.2.3}$$

where,

$$C_e = \begin{cases} 0 \text{ if } \alpha \text{ or } \beta \text{ is odd} \\[2ex] \dfrac{4}{(\alpha+1)(\beta+1)} \text{ otherwise.} \end{cases}$$

Substituting (4.2.3) in (4.2.2), we have,

$$\sum w_i x_i^{\alpha} y_i^{\beta} = C_e \quad . \tag{4.2.4}$$

In the model above the total number of unknowns is 3n, i.e. (x_i, y_i, w_i), $i=1,n$. From theory (A.H. Stroud) it is known that a true multidimensional Gaussian formula in which all 3m degrees are used to satisfy equations of the form (4.2.2) exactly does not exist. Hence extra equations are introduced to invoke symmetry conditions (based often on the spatial geometry) and some equations of the form

(4.2.2) are dropped. Moreover not all equations of the form, (4.2.2) are independent. For example with the cubic region above choosing $\alpha=0$, $\beta=2$ and $\alpha=2$, $\beta=0$ gives the same equation. This is also a symmetry effect.

Hence three ways of increasing the number of equations without introducing dependence are considered.

(a) We could use further independent equations of the form (4.2.2). (Note even though some of the sequence of equations in this set are omitted due to symmetry they will be satisfied identically). In fact (α,β) takes on the series of values such as (0,0),(0,1), (2,0),(1,1),(0,2) with symmetrically dependent terms missing.

(b) Symmetry conditions on the abscissas can be invoked directly. For example,

$$\left.\begin{aligned} x_i &= x_j \\ y_i &= y_j \\ x_i &= y_j \end{aligned}\right\} \qquad (4.2.5)$$

where $1\leq i\leq n$, $1\leq j\leq n$, i,j and n are integers.

(c) Finally weight symmetry can be used. For example $w_i=w_j$, $1\leq i\leq n$, $1\leq j\leq n$, i,j and n are integers and n the number of points in the rule.

The third option could result in the extreme case of Gauss-Chebyshev type of formulae with equal weights although the effectiveness of such formulaein multidimensional quadrature is not well known.

In solving for w_i's, x_i's and y_i's, the first set of n equations is treated as linear in w_i's so that the w_i's can be expressed in terms of x_i's and y_i's (by Gaussian elimination). On substitution the remaining set of 2n equations is then treated as a non-linear set of equations in x_i's and y_i's. This is then solved by using Newton's method. Both processes are treated fully numerically, the matrix element representing the coefficients of w_i's being declared as functions of x_i and y_i so that the non-linear equations being fed to the Newton algorithm contains a dynamic Gauss eliminator in their codes.

In order to monitor any ill-conditioning the condition numbers of the various sets of linear equations involved were found after off-loading the relevant coefficients. In all cases quoted the level of ill-conditioning was found to be very small, though clearly increasing for higher point number formulae. The program for this routine is displayed in Appendix (v).

4.3 <u>THE NEW MINIMUM POINT RULES</u>

(i) The model in (4.2.2) was used to produce the 2-point Gauss product rule. The exact set of equations for the 2-point product rule are as shown below.

From (4.2.2) we have that,

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dxdy = \sum w_i x_i^{\alpha} y_i^{\beta}$$

The first set of 4 equations and second set of 8 equations are as follows:

$$\left.\begin{array}{ll}
(1) & \sum\limits_{i}^{4} w_i = 4 \\[2ex]
(2) & \sum\limits_{i}^{4} w_i x_i = 0 \\[2ex]
(3) & \sum\limits_{i}^{4} w_i y_i = 0 \\[2ex]
(4) & \sum\limits_{i=1}^{4} w_i x_i y_i = 0
\end{array}\right\}\ \text{1st set of 4 equations treated as linear in } w_i\text{'s.}$$

$$\left.\begin{array}{ll}
(5) & \sum w_i x_i^2 y_i^2 = \frac{4}{9} \\[2ex]
 & \left.\begin{array}{ll}
(6) & x_2 = x_4 \\
(7) & x_1 \doteq x_3 \\
(8) & x_1 = -x_2 \\
(9) & x_2 = y_3 \\
(10) & x_4 = x_5 \\
(11) & x_1 = y_1 \\
(12) & x_4 = y_4
\end{array}\right\}\ \text{equations from symmetry conditions}
\end{array}\right\}\ \text{2nd set of 8 equations treated as non-linear after sub. for the values of } w_i\text{'s.}$$

Using Gaussian elimination in the first set of 4 equations and substituting for w_i's, the resulting equations in (5)-(12) are treated as non-linear in x_i's and y_i's and are solved using Newtons method.

With starting values of,

$x_1 = -0.6$

$y_1 = -0.6$

$x_2 = 0.6$

$y_2 = -0.6$

$x_3 = 0.6$

$x_4 = 0.6$

$y_4 = 0.6$

We have the following result:

i	w_i	x_i	y_i
1	1.0	-0.57735026918962576467	-0.5773502618962576467
2	1.0	0.57735026918962576467	-0.5773502618962576467
3	1.0	-0.57735026918962576467	0.5773502618962576467
4	1.0	0.57735026918962576467	0.5773502618962576467

This result is in total agreement with the two-point Gaussian product and so the above model formed a validated basis for this work. Hence the starting values for x and y are dictated by the symmetry of the region of integration.

(ii) <u>5 points rule</u>

In 5 points rule the integral of the form $\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dxdy$ was considered. As indicated in the model above, we have that,

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y) = \sum w_i x_i^{\alpha} y_i^{\beta}$$

The two sets of equations are made up of the following:

$$\left.\begin{array}{ll}
(1) & \sum_{i=1}^{5} w_i = 4 \\
(2) & \sum w_i x_i = 0 \\
(3) & \sum w_i y_i = 0 \\
(4) & \sum w_i x_i y_i = 0 \\
(5) & \sum w_i x_i^2 = \frac{4}{3}
\end{array}\right\} \text{1st set of 5 equations linear in } w_i\text{'s}$$

$$\left.\begin{array}{ll}
(6) & \sum w_i x_i^2 y_i = 0 \\
(7) & \sum w_i x_i y_i^2 = 0 \\
(8) & \sum w_i x_i^2 y_i^2 = \frac{4}{9} \\
(9) & \sum w_i y_i^3 = 0 \\
(10) & \sum w_i x_i^3 = 0 \\
\left.\begin{array}{ll}
(11) & x_2 = x_4 \\
(12) & x_1 = x_3 \\
(13) & x_1 = -x_2 \\
(14) & x_1 = y_1 \\
(15) & x_4 = y_4
\end{array}\right\} & \text{equations from symmetry conditions}
\end{array}\right\} \text{2nd set of 10 equations treated as non-linear in } x_i\text{'s and } y_i\text{'s}$$

Treating the first set of 5 equations as linear in w_i's, the w_i's are then expressed in terms of x_i's and y_i's. Substituting for w_i's in (6)-(15) and treating these as non-linear equations in x_i's and y_i's the equations are again solved using Newton. With starting values of,

$$x_1 = -0.5, \quad y_1 = -0.5$$

$$x_2 = 0.5, \quad y_2 = -0.5$$

$$x_3 = -0.5, \quad y_3 = 0.5$$

$$x_4 = 0.5, \quad y_4 = 0.5$$

$$x_5 = 0.0, \quad y_5 = 0.0$$

we have the following results:

5(a)

i	w_i	x_i	y_i
1	0.64265722402873640779	-0.66673264565535991723	-0.66673264565535991723
2	0.85704591459275138404	0.6667326456553599734	-0.49995052065537574118
3	0.85704591459275138426	-0.66673264565535991745	0.49995052065537574107
4	0.64265722402873640704	0.66673264565535991745	0.66673264565535991752
5	1.0005937227570244172	0.0	0.0

(iii) Another 5 point rule can be produced with weight conditions introduced. Here the first set of 5 equations is the same as in (ii) but with the 2nd set of ten equations taking the form below.

$$\left.\begin{array}{lll}
(6) & \sum w_i x_i^2 y_i = 0 & \\
(7) & \sum w_i x_i y_i^2 = 0 & \\
(8) & \sum w_i x_i^2 y_i^2 = \frac{4}{9} & \\
(9) & \sum w_i y_i^3 = 0 & \\
(10) & w_1 = w_4 \;\} & \text{wt equation} \\
(11) & x_2 = x_4 & \\
(12) & x_1 = x_3 & \\
(13) & x_1 = -x_2 & \\
(14) & x_1 = y_1 & \\
(15) & x_4 = y_4 &
\end{array}\right\} \text{2nd set of 10 equations}$$

Using starting values of:

$x_1 = -0.6$

$y_1 = -0.6$

$x_2 = 0.6$

$y_2 = -0.6$

$x_3 = -0.6$

$y_3 = 0.6$

$x_4 = 0.6$

$y_4 = 0.6$

$x_5 = 0.0$

$y_5 = 0.0$

we have the following results:

5(b)

i	w_i	x_i	y_i
1	0.68465843842649082437	-0.65339533882747565289	-0.65339533882747565213
2	0.87689437438233944937	0.65339533882747565289	-0.51015566461111166064
3	0.87689437438233945078	-0.6533953388274755258	0.51015566461111165977
4	0.68465843842649082556	0.6533953388274565267	0.65339533882747565228
5	0.87689437438233944981	0.0	0.0

(iv) <u>5 points rule with equal weights</u>

By imposing the condition that all the weights be equal the set of equations we have for a five point rule are:

$$\left.\begin{array}{ll} (1) & \sum_{i=1}^{5} w_i = 4 \\ (2) & \sum w_i x_i = 0 \\ (3) & \sum w_i y_i = 0 \\ (4) & \sum w_i x_i y_i = 0 \\ (5) & \sum w_i x_i^2 = \frac{4}{3} \end{array}\right\} \text{1st set of 5 equations linear in } w_i\text{'s}$$

$$\left\{\begin{array}{ll} (6) & \sum w_i y_i^2 = \frac{4}{3} \\ (7) & \sum w_i x_i^2 y_i = 0 \\ (8) & \sum w_i x_i y_i^2 = 0 \\ (9) & \sum w_i x_i^3 = 0 \end{array}\right.$$

$$\left.\begin{array}{ll}(10) & \sum w_i y_i^3 = 0 \\ (11) & \sum w_i x_i^3 y_i = 0 \\ (12) & \left.\begin{array}{l} w_1 = w_2 \\ w_3 = w_4 \\ w_5 = w_1 \\ w_2 = w_4 \end{array}\right\} \text{wt equations} \\ (13) & \\ (14) & \\ (15) & \end{array}\right\} 2^{nd} \text{ set of 10 equations}$$

With the introduction of weights the abscissas are freed and starting with the following values,

$x_1 = -0.5, \quad y_1 = -0.5$

$x_2 = 0.5, \quad y_2 = -0.5$

$x_3 = -0.5, \quad y_3 = 0.5$

$x_4 = 0.5, \quad y_4 = 0.5$

$x_5 = 0.0, \quad y_5 = 0.0$

We have the following results:

5(c)

i	w_i	x_i	y_i
1	0.8	-0.64549722436790281442	-0.64549722436790281431
2	0.8	0.64549722436790281399	-0.64549722436790281442
3	0.8	-0.6454972243679081442	0.64549722436790281491
4	0.8	0.64549722436790281431	0.64549722436790281420
5	0.8	0.0	0.0

(v) <u>6-point rule</u>

With the 6 points rule (with equal weights) we have the following sets of equations:

$$\left.\begin{array}{ll}
(1) & \sum_{i=1}^{6} w_i = 4 \\
(2) & \sum w_i x_i = 0 \\
(3) & \sum w_i y_i = 0 \\
(4) & \sum w_i x_i y_i = 0 \\
(5) & \sum w_i x_i^2 y_i = 0 \\
(6) & \sum w_i y_i^2 = \frac{4}{3}
\end{array}\right\} 1^{st} \text{ set of 6 equations}$$

$$\left.\begin{array}{ll}
(7) & \sum w_i x_i^4 = \frac{4}{5} \\
(8) & \sum w_i x_i y_i^2 = 0 \\
(9) & \sum w_i x_i^3 = 0 \\
(10) & \sum w_i x_i^5 = 0 \\
(11) & \sum w_i y_i^5 = 0 \\
(12) & \sum w_i x_i y_i^3 = 0 \\
(13) & \sum w_i y_i^4 = \frac{4}{5} \\
\multicolumn{2}{l}{\left.\begin{array}{ll}
(14) & w_1 = w_2 \\
(15) & w_3 = w_4 \\
(16) & w_5 = w_6 \\
(17) & w_4 = w_2 \\
(18) & w_1 = w_5
\end{array}\right\} \text{wt equations}}
\end{array}\right\} \begin{array}{l} 2^{nd} \text{ set of 12 equations} \\ \text{treated as non-linear in} \\ x_i\text{'s and } y_i\text{'s} \end{array}$$

Solving these equations with the following starting values of:

$x_1 = -0.5,\ y_1 = -0.5,\ x_2 = 0.0,\ y_2 = -0.8,\ x_3 = 0.5,\ y_3 = -0.5,$

$x_4 = -0.5,\ y_4 = 0.5,\ x_5 = 0.0,\ y_5 = 0.8,\ x_6 = 0.5,\ y_6 = 0.5,$

we have the following results:

6 points

i	w_i	x_i	y_i
1	0.66666666666666666620	-0.74008280449228525035	-0.35002117458154067810
2	0.66666666666666666685	0.0	-0.86889030072220120 4915
3	0.66666666666666666641	0.74008280449228525024	-0.35002117458154067815
4	0.66666666666666666631	-0.74008280449228525089	0.35002117458154067818
5	0.66666666666666666639	0.0	0.86889030072220120495
6	0.66666666666666666685	0.74008280449228525057	0.35002113458154067761

(vi)a <u>7-points rule</u>

With the 7 points rule (with equal weights) we have the following sets of equations:

$$\left.\begin{array}{lll}
(1) & \sum_{i=1}^{7} w_i & = 4 \\
(2) & \sum w_i x_i & = 0 \\
(3) & \sum w_i y_i & = 0 \\
(4) & \sum w_i x_i y_i & = 0 \\
(5) & \sum w_i x_i^2 & = \frac{4}{3} \\
(6) & \sum w_i y_i^2 & = \frac{4}{3} \\
(7) & \sum w_i x_i^2 y_i & = 0
\end{array}\right\} \quad 1^{st} \text{ set of 7 equations}$$

(8) $\sum w_i x_i^5 = 0$

(9) $\sum w_i y_i^5 = 0$

(10) $\sum w_i x^5 y_i = 0$

(11) $\sum w_i x_i y_i^5 = 0$

(12) $\sum w_i x_i y_i^4 = 0$

(13) $\sum w_i x_i^5 y_i^4 = 0$

(14) $\sum w_i x_i^5 y_i^5 = 0$

(15) $\sum w_i x_i^4 y_i^5 = 0$

(16) $w_1 = w_2$

(17) $w_3 = w_4$

(18) $w_5 = w_6$

(19) $w_7 = w_1$

(20) $w_3 = w_2$

(21) $w_5 = w_7$

weight equations (16)–(21)

2nd set of 14 equations treated as non-linear (8)–(21)

With starting values of:

$x_1 = -0.8, \quad y_1 = -0.3$

$x_2 = 0.0, \quad y_2 = 1.0$

$x_3 = 0.8, \quad y_3 = -0.3$

$x_4 = -0.8, \quad y_4 = 0.3$

$x_5 = 0.0, \quad y_5 = 1.0$

$x_6 = 0.8, \quad y_6 = 0.3$

$x_7 = 0.0, \quad y_7 = 0.0$

We have the following results:

i	w_i	x_i	y_i
1	0.57142549470901745452	-0.76377634150326558871	-0.29834285119026504075
2	0.57142906531711892620	0.0	-0.9943035730749995978
3	0.57142549470901745474	0.76377634150326558827	-0.29834285119026503994
4	0.57143164836930384893	-0.7637748899020107145	0.29836108534558329668
5	0.57142806588149074965	0.0	0.99430285531030323598
6	0.57143164336930384796	0.763374889902010	0.29836108534558329510
7	0.57142852644747771877	0.0	-0.00040439362823571 50

demonstrating ill-conditioning due to linear dependence

(vi)b When the equations (9),(10) and (11) in (vi) are replaced by

$$(9)\quad \sum w_i y_i^{25} = 0$$

$$(10)\quad \sum w_i x_i^{25} y^{11} = 0$$

$$(11)\quad \sum w_i x^{11} y^{25} = 0$$

and using the same starting values we have the following results:

i	w_i	x_i	y_i
1	0.57142857066576083983	-0.76376261922451045344	-0.29999950285202012778 5
2	0.57142857155633176749	0.0	-0.99331262182355891 07
3	0.57142857066576084016	0.76376261922451045810	-0.29999950285020127687
4	0.57142857219138202998	-0.76376261242743619996	0.29999950738891820168
5	0.57142857130419388910	0.0	0.99331125910324016284
6	0.57142857219138203074	0.76376261242743619562	0.29999950738891820374
7	0.57142857142518860313	0.0	-0.00000007161927907 92

(vii) <u>9-point rule with equal weights</u>

For a 9-point rule the set of equations involved are as follows:

$$\left.\begin{array}{ll}
(1) & \sum_{i=1}^{9} w_i = 4 \\
(2) & \sum w_i x_i = 0 \\
(3) & \sum w_i y_i = 0 \\
(4) & \sum w_i x_i y_i = 0 \\
(5) & \sum w_i x_i^2 = \frac{4}{3} \\
(6) & \sum w_i y_i^2 = \frac{4}{3} \\
(7) & \sum w_i x_i^2 y_i = 0 \\
(8) & \sum w_i x_i y_i^2 = 0 \\
(9) & \sum w_i x_i^2 y_i^2 = \frac{4}{9}
\end{array}\right\} \text{1st set of 9 equations linear in } w_i\text{'s.}$$

$$\left.\begin{array}{ll}
(10) & \sum w_i x_i^3 = 0 \\
(11) & \sum w_i y_i^3 = 0 \\
(12) & \sum w_i x_i^3 y_i = 0 \\
(13) & \sum w_i y_i^5 = 0 \\
(14) & \sum w_i y_i^6 = \frac{4}{7} \\
(15) & \sum w_i y^7 = 0 \\
(16) & \sum w_i x_i^3 y_i^2 = 0 \\
(17) & \sum w_i x_i^3 y_i^3 = 0
\end{array}\right|$$

$$\left.\begin{array}{ll}
(18) & \sum w_i x_i^2 y_i^3 = 0 \\
(19) & \sum w_i x_i^5 y_i^2 = 0 \\
 & \left.\begin{array}{ll}
(20) & w_1 = w_2 \\
(21) & w_3 = w_4 \\
(22) & w_5 = w_6 \\
(23) & w_7 = w_8 \\
(24) & w_9 = w_1 \\
(25) & w_3 = w_5 \\
(26) & w_6 = w_7 \\
(27) & w_8 = w_2
\end{array}\right\} \text{weight equations}
\end{array}\right\} \begin{array}{l}2^{nd}\text{ set of 18 equations} \\ \text{treated as non-linear} \\ \text{in } x_i^{\prime}\text{s and } y_i^{\prime}\text{s.}\end{array}$$

With starting values of:

$$\begin{array}{llll}
x_1 = -0.78, & & y_1 = -0.78 \\
x_2 = 0.0 & & y_2 = -0.78 \\
x_3 = 0.78 & & y_3 = -0.78 \\
x_4 = -0.78 & & y_4 = 0.78 \\
x_5 = 0.0 & & y_5 = 0.78 \\
x_6 = 0.78 & & y_6 = 0.78 \\
x_7 = -0.78 & & y_7 = 0.0 \\
x_8 = 0.0 & & y_8 = 0.0 \\
x_9 = 0.78 & & y_9 = 0.0
\end{array}$$

We have the following results:

i	w_i	x_i	y_i
1	0.4444241582707352855	-0.8137089875143209672	-0.6146905493843060025
2	0.4444518803265492236	0.00031081203690234615	-0.8627945358277435807
3	0.4444130287316860095	0.8140162587744951551	-0.61435756439783453006
4	0.4448050817388163561	-0.812338166693188049714	0.61534747325323288633
5	0.4444248869405269037	0.00031764783683278901	0.86212930789494299406
6	0.4444308630563611389	-0.81269280438417541379	0.61500983479212011016
7	0.44415956445468844269	-0.42237173212933225069	0.00485904890168844906
8	0.44463381551067402060	-0.00083786962247918029	-0.01042840689288041142
9	0.44454080958780152911	0.42007286129450621307	0.0048861036966911564

with linear dependence as in (vi)a,

(vii)b. When equation (10) in (vii) is changed to $\sum^{9} w_i y_i = 0$ and using the same starting values we get the following results:

i	w_i	x_i	y_i
1	0.44444444444446292	-0.71730585014582323527	-0.69729904195897098290
2	0.44444444444433895	-0.00000000000000014943	-0.72608726096049206855
3	0.444444444444550278	0.71730585014582331365	-0.69729904195897097130
4	0.444444444444336165	-0.71577695410879641109	0.698291634125419729
5	0.444444444444530779	0.00000000000000012946	0.7246311372027099358
6	0.444444444444443316	0.71577695410879648200	-0.69829163403125419501
7	0.4444444444444564536	-0.68784858022192045686	0.00215427648444910291
8	0.444444444444464019	-0.00000000000000030024	-0.00483761335568257135
9	0.4444444444444325848	0.687848580221920294131	0.00215427648444912185

(viii) <u>13-points rule</u>

With the 13 points rule the two sets of equations considered are as follows:

$$\left.\begin{array}{ll}
(1) & \sum_{i=1}^{13} w_i = 4 \\
(2) & \sum w_i x_i = 0 \\
(3) & \sum w_i y_i = 0 \\
(4) & \sum w_i x_i y_i = 0 \\
(5) & \sum w_i x_i^2 = \frac{4}{3} \\
(6) & \sum w_i y_i^2 = \frac{4}{3} \\
(7) & \sum w_i x_i^2 y_i = 0 \\
(8) & \sum w_i x_i y_i^2 = 0 \\
(9) & \sum w_i x^2 y_i^2 = \frac{4}{9} \\
(10) & \sum w_i x_i^3 y_i^2 = 0 \\
(11) & \sum w_i x_i^3 y_i^3 = 0 \\
(12) & \sum w_i x_i^2 y_i^3 = 0 \\
(13) & \sum w_i x_i^2 y_i^4 = \frac{4}{15}
\end{array}\right\} \text{1st set of 13 equations}$$

$$\begin{array}{ll}
(14) & \sum w_i x_i^3 = 0 \\
(15) & \sum w_i x_i y_i^3 = 0 \\
(16) & \sum w_i y_i^4 = \frac{4}{5}
\end{array}$$

(17) $\sum w_i x_i^5 y_i = 0$

(18) $\sum w_i x_i y^5 = 0$

(19) $\sum w_i x_i^2 y^7 = 0$

(20) $\sum w_i x_i y_i^4 = 0$

(21) $\sum w_i x_i y_i^6 = 0$

(22) $\sum w_i x_i y^7 = 0$

(23) $\sum w_i x_i^5 y_i^3 = 0$

(24) $\sum w_i x_i^3 y_i^7 = 0$

(25) $\sum w_i x_i^5 y^5 = 0$

(26) $\sum w_i x_i^3 y_i^8 = 0$

(27) $\sum w_i x_i^5 y_i^7 = 0$

2nd set of 26 equations treated as non-linear in x_i's and y_i's.

(28) $w_1 = w_2$

(29) $w_3 = w_4$

(30) $w_5 = w_6$

(31) $w_7 = w_8$

(32) $w_9 = w_{10}$

(33) $w_{11} = w_{12}$

(34) $w_{13} = w_1$

(35) $w_2 = w_3$

(36) $w_5 = w_9$

(37) $w_6 = w_7$

(38) $w_{10} = w_{11}$

(39) $w_4 = w_5$

With starting values of:

x_1	$= -0.78$,	y_1	$= -0.78$
x_2	$= 0.0$,	y_2	$= -0.78$
x_3	$= 0.78$,	y_3	$= -0.78$
x_4	$= -0.78$,	y_4	$= 0.78$
x_5	$= 0.0$,	y_5	$= 0.78$
x_6	$= 0.78$,	y_6	$= 0.78$
x_7	$= -7.8$,	y_7	$= 0.0$
x_8	$= 0.0$,	y_8	$= 0.0$
x_9	$= 0.78$,	y_9	$= 0.0$
x_{10}	$= -0.3$,	y_{10}	$= -0.3$
x_{11}	$= 0.3$,	y_{11}	$= -0.3$
x_{12}	$= -0.3$,	y_{12}	$= 0.3$
x_{13}	$= 0.3$,	y_{13}	$= 0.3$

i	w_i	x_i	y_i
1	0.30769230769230767800	-0.76492418809482633755	-0.77961330614441911929
2	0.30769230769229711896	-0.00000000000000003445	-0.799989947422098803
3	0.20769230769230774002	0.76492418809482622035	-0.77961330614441910758
4	0.30769230769230764981	-0.76478878273873230665	0.77964109428407462144
5	0.30769230769230497286	-0.00000000000000003252	0.90480195105775847795
6	0.30769230769230770787	0.76478878273873218175	0.7796401942840746134
7	0.307692307692307527751	-0.93862453720499981497	0.0058691718660076066715
8	0.30769230769231965155	-0.00000000000000282922	-0.21807561721437668665
9	0.30769230769230778160	0.9386245372050008732	0.0058691718660075783340
10	0.30769230749230846757	-0.26554040812044791067	-0.28816497264435434155
11	0.30769230769230750794	0.26554040812044637002	-0.28816497264435351154
12	0.30769230769230955004	-0.21243361490974740905	-0.33890071944012364501
13	0.30769230769230664692	0.21243361490974539734	0.33890071944012322444

It is noted that the monomials for equal weights are not symmetric especially in the higher powers. The resulting formulae therefore have slightly off symmetric points for this reason. These choices were forced upon us by either linear dependence or lack of existence of a solution, as all these formulae depend on the full, exact, solution of the defining equations. Figures (2) to (7) show the spread of points in the region defined by $-1.0 \leq x \leq 1.0$ and $-1.0 \leq y \leq 1.0$ for the respective 5,6,7,9 and 13 point rules.

FIGURE (2)

(a and b) 5 points

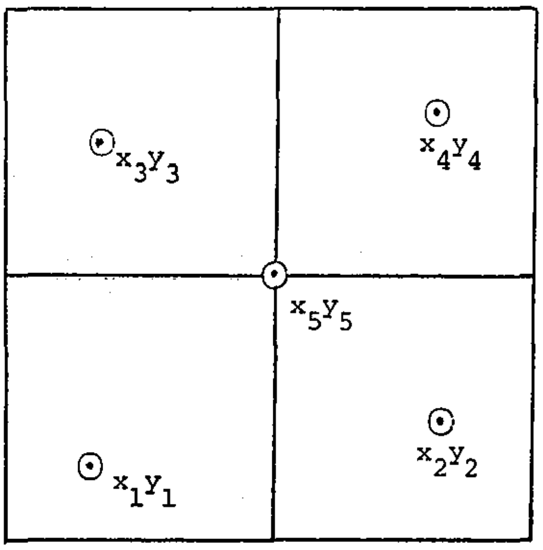


FIGURE (3)

(c) 5 points

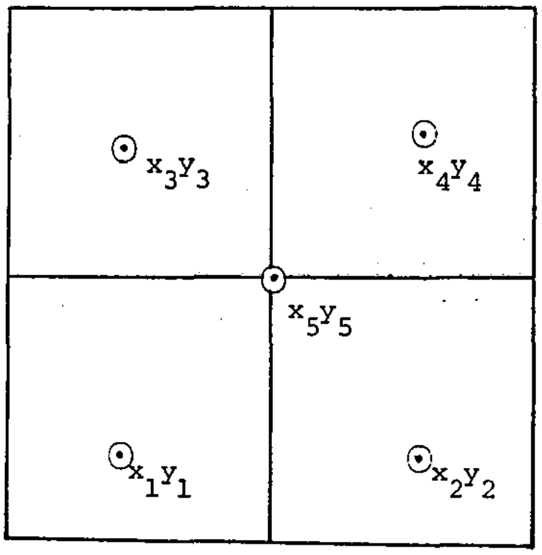


FIGURE (4)

6 points

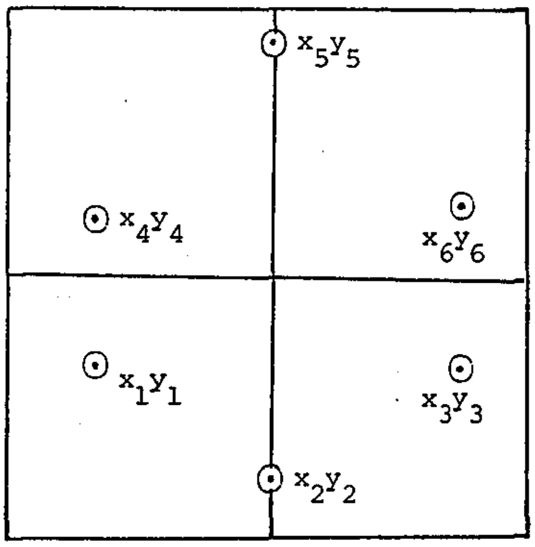


FIGURE (5)

7 points

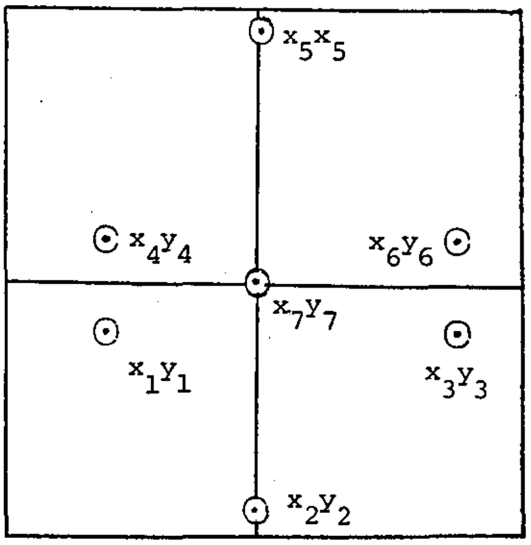


FIGURE (6)

9 points

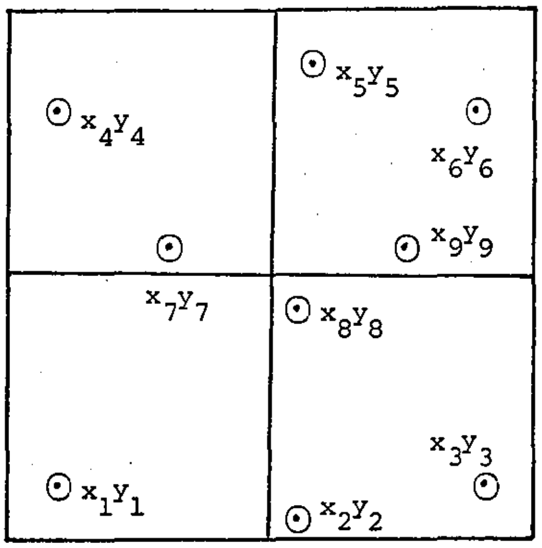
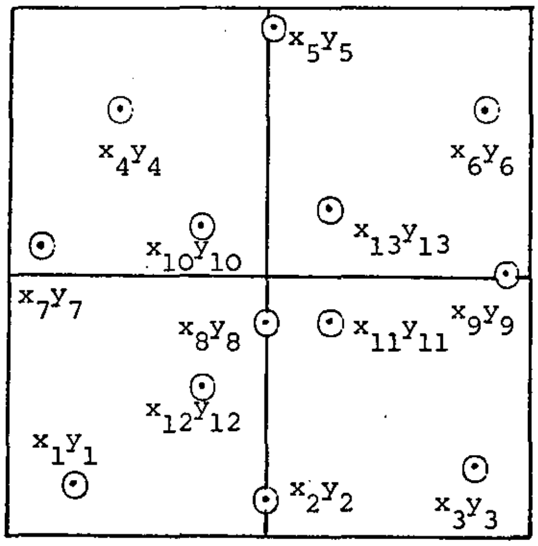


FIGURE (7)

13 points



4.4 EXPERIMENTAL RESULTS AND CONCLUSIONS

The rules developed were applied to various integrals I defined by:

$$I_1 = \int_{-1}^{1}\int_{-1}^{1} \frac{dxdy}{\sqrt{(4-x-2y)}} = 2.0958446$$

$$I_2 = \int_{-1}^{1}\int_{-1}^{1} \frac{dxdy}{(4+x+y)} = 1.0464963$$

$$I_3 = \int_{-1}^{1}\int_{-1}^{1} \frac{dxdy}{\sqrt{(3-x^2-y^2)}} = 2.6555866$$

$$I_4 = \int_{-1}^{1}\int_{-1}^{1} e^x \cos y dxdy = 3.955591$$

$$I_5 = \int_{-1}^{1}\int_{-1}^{1} e^x \sin y dxdy = 0$$

$$I_6 = \int_{-1}^{1}\int_{-1}^{1} e^{xy} dxdy = 4.2290035$$

The results in Table V1 and V2 show that the new rules developed compare favourably with existing ones. The 13 point rule seems to be the best of the new rules. In addition to this the performance of these rules appears to improve with increase in the number of points. This is consistent with the model used in developing the above rules since the $\iint f(x,y)dxdy$ can only be evaluated exactly by the rule when the integrand f is defined as,

$$f(x,y) = x^\alpha y^\beta$$

So the higher the values of the number of points used, the more accurate the approximation.

	I_1	I_2	I_3
GAUSS 9 PTS PRODUCT RULE	2.0953358	1.0464731	2.6514268
RADON	2.094937	1.0453342	2.6472813
ALBRECHT AND COLLATZ	2.094441	1.0463702	2.642812
COHEN AND GISMALLA	2.0913961	1.0464845	2.6541559
NEW FORMULAS			
5 POINTS RULE (A)	2.076392958	1.040921313	2.591229547
5 POINTS RULE (B)	2.077946837	1.041465851	2.594184682
5 POINTS RULE (C)	2.093217447	1.04651128	2.635852121
6 POINTS RULE	2.090753844	1.046450669	2.636952733
7 POINTS RULE	2.092437918	1.044414799	2.633849630
9 POINTS RULE	2.093028943	1.046085180	2.638122727
13 POINTS RULE	2.097414898	1.04625322	2.655895646

TABLE V1

NEW FORMULAS	I_4	I_5	I_6
5 POINTS RULE (C)	3.907445679	0.0	4.281819883
6 POINTS RULE	4.083778656	0.0	4.089973626
7 POINTS RULE	4.047420824	-0.000000461	4.233577168
9 POINTS RULE	3.944323412	-0.00007166	4.226890560
13 POINTS RULE	3.960338146	-0.01193567	4.228784968

TABLE V2

CHAPTER 5

CONCLUSION

The purpose of this chapter is to bring together the conclusions reached in the previous chapters and draw attention to possible directions in which further investigation could be made.

In Chapter 2, two methods of transformation - the IMT type of rule and the Error type rule were generalised. The rules developed were applied to various test integrals and their performance compared favourably with existing methods. The results also showed that the new rules are also robust for non-singular integrals but are not competitive with conventional rules such as Patterson or Clenshaw-Curtis. This seems to indicate that rules based on transformations should be used where possible for singular integrals, where conventional rules perform badly. Moreover, since the transformations used were not standard functions, attention had to be paid to effective means of computing them. Previous work had been based primarily on standard functions and as long as methods to compute suitable non-standard functions were available it seemed workable to investigate these possibilities. Hence one of the recommendations of this work is for a further investigation into the use of non-standard functions as a means of dealing with singular integrals. In addition, the performance of the IMT type of rule as revealed by the tests depends on the appropriate choice of c and so far, the way of choosing this beforehand has not been established. So a search for a way of choosing the values of c for different classes of function will certainly be of interest if the full power of the IMT type of rule is to be realised. These investigations could just result in a transformation method competitive with Clenshaw-Curtis or Patterson on a smooth integrand.

Chapter 3 is divided into two sections. While the first part deals with 2-dimensional singular integrals, the second half is devoted to non-singular integrals in 2-dimensions.

In the first part two transformations - the tanh transformation and the Error transform were employed to develop efficient rules for 2-dimensional singular integrals. The performance of the rules was compared with Romberg and a standard Gauss-Legendre product rule which is one of the best known methods and the results show that the new rules are more efficient and more accurate. With regard to the new rules two strategies were adopted. First a further transformation was introduced and this enabled us to combine the efficiency of the trapezoidal and Clenshaw-Curtis rules. So by integrating in concentric circles it was possible to scan the area of significant contribution to the quadrature easily. The alternative approach was to use the trapezoidal product rule in the infinite plane and monitoring the contribution in each node of the grid. The results showed that both strategies have their uses. The first gives very accurate results with integrals having derivative singularities within the region of integration while the second strategy is clearly more efficient and more accurate when applied to integrals with boundary singularities. However these rules have been developed for 2 dimensional integrals with a view to extending them to higher dimensions. So the recommendation here is an investigation of this approach to high dimensional singular integrals. The simplicity of the grid approach may well prove inviting in this context.

In the second part of Chapter 3, the polynomial transformation was used to improve the trapezoidal product rule for non-singular 2-dimensional integrals. This resulted in the reduction of the number of function evaluations as well as improved accuracy. This investigation has been restricted to 2-dimensions and so far it appears to be successful. It will be of interest to know what happens in higher dimensions and the efficiency of other similar transformations should be investigated.

In Chapter 4, various minimum point rules were developed using monomials and the results showed that the efficiency of the rules increases with the number of points. Beyond the 13 points rule it was observed that ill-conditioning begins to set in while using the dynamic Gaussian elimination in the code. The way to deal with this is to employ multiple precision codes which will prove expensive, but no more so than the generation of high order Patterson rules for example.

BIBLIOGRAPHY

ABRAMOWITZ, M. and STEGUN, I.A.,
Handbook of mathematical functions,(Dover, New York, 1965).

ANDERS, E.B., *An extension of Romberg integration procedure to N variables*, JACM 13(1966), 505-510.

ALAYLIOGLU, A., EVANS, G.A. and HYSLOP, J. (1973),
The evaluation of oscillatory integrals with infinite limits, J.Computational Phys., Vol.13, No.3, pp.433-438.

ALAYLIOGLU, A., EVANS, G.A. and HYSLOP, J. (1974),
Some comments on the derivation and structure of Newton-Cotes quadrature formulae, Intern.J.Mathematical Ed., Vol.5, pp.213-217.

ARFKEN, G. *Mathematical methods for physicists*, Academic Press, New York and London (1967).

BABUSHKA, I., *Optimal quadrature formulae*, Dokyl.Akad.Nauk, SSSR, 149(1963), pp.227-229 = Soviet Math.Dokl.4(1963),pp.330-332.

BAKHVALOV,N.S.,*On approximate calculation of multiple integrals*, Ibid.,10(1965), pp.96-129 (in Russian).

BAKHVALOV, N.S., KOROBOV, N.N. and CENCOU, N.N.,
The application of number-theoretic nets to numerical analysis problem (in Russian), in Proc. Fourth All-Union Math.Congr. Leningrad 1961, Vol.11, pp.580-587, IZ det Nauks Leningrad 1964, MR 36# 4776.
On approximate calculation of multiple integrals, Vestnik Moscow Univ.Ser.Math.Mch.Astr.Fz.Hem, 4(1959), pp.3-18.

BAKER, C.T.H. *The numerical treatment of integral equations*, Clarendon Press, Oxford 1977.

BAKHVALOV, N.S. and VASIL'EVA, L.G. (1968), *Evaluation of integrals of oscillatory function by interpolation of nodes of Gaussian quadrature*, Zh. Vychul.Mat.Mat.Fiz., Vol. 8, Part 1, pp.241-249.

BARNHILL, R.E. *The convergence of quadratures in complex contours*, SIAM J.Numer.Anal. 2(1965), pp.321-336.

BLAKEMORE M., EVANS, G.A. and HYSLOP, J. (1975), *The evaluation of two-centre molecular integrals involving one-electron Green's functions*, Theoret.Chim.Acta. 38, pp.311-326.

CERULUS, R. and HAGEDORN, R., *A Monte Carlo method to calculate multiple phase space integrals I, II*, Nuovo Cininto, (10)V, 9, 1958, pp. 646-677, MR.21, 79.

CHISHOLM, J.S.R., GENZ, A. and ROWLAND, G.A., *Accelerated convergence of sequences of quadrature approximations*, J.Comp.Phys., 10(1973), pp.284-307.

CLENSHAW, C.W. and CURTIS, A.R. (1960), *A method of numerical integration on an automatic computer*, Num.Math., Vol. 2, pp.197-205.

COHEN, A.M., *Cautious Romberg extrapolation*, Intern.J. Computer Math. 8, (1980), pp.137-147.

COHEN, A.M. and GISMALLA, D.A.,

The construction of quadrature rules by parameter optimization, Intern.J. Computer Maths. (1985), Vol.17, pp.203-214.

COWDREY, D.R. and REEVES, C.M.,

An application of the Monte Carlo method to the evaluation of molecular integrals, Comput.J. 6 (1963-1964), pp.277-286.

DAVIS, P.J. and RABINOWITZ, P.,

Methods of numerical integration, Academic Press, Inc. (London) Ltd.,

Some Monte Carlo experiments in computing multiple integrals, MTAC 10 (1956), pp.1-8.

DELVES, L.M. and MOHAMED, J.L.,

Computational methods of integral equations, Cambridge University Press, 1985.

ERDELYI, A. (Ed) (1954) TABLES OF INTEGRA TRANSFORMS, VOL. I MCGRAW-HILL. [VIII]

EVANS, G.A., (1969),

Some problems on low Reynolds number flow, D.Phil. Thesis, Oxford.

EVANS, G.A. and OCKENDON (1972),

The drag on a sphere in low Reynolds number flows, J.Aero.Sci., Vol. 3, pp.237-242.

EVANS, G.A. (1973),

The forces on small spheres due to the interactive *inertia effects of neighbouring spheres*, Mathematics Research, No. 21, Loughborough University.

EVANS, G.A., FORBES, R.C. and HYSLOP, J.,
A note on the use of polynomial transformation in evaluating singular integrals, Int.J.Comp.Math. 14(1985), 4, pp.157-170.

EVANS, G.A., FORBES, R.C. and HYSLOP, J.,
The implementation of the Tanh transformation in the evaluation of singular integrals, Int.J.Comp.Math., 15 (1984), pp.339-358.

EVANS, G.A., HYSLOP, J. and MORGAN, A.P.G.,
An extrapolation procedure for the evaluation of singular integrals, Intern.J. Computer Math. 12(1983), pp.251-256.

GAUSS, C.F., *Methods nova integration volores per approximationen inveniendi*, Werke, Vol.3, (1866), pp.163-169.

GRADSHTEYN, I.S. and RYZHIK, I.M., (1965),
Tables of integrals, series and products, New York, Academic Press.

HABER, S., *Numerical evaluation of multiple integrals*, SIAM Review, Vol.12, No. 4, pp.481-526.
The tanh rule for numerical integration, SIAM J.Numer. Anal. 14(1977), pp.668-685.

HALL, G.G. (1967),

A variation principle for continuous wave functions, Chem.Phys.Letters, Vol.1, pp.495-496.

HALL, G.G., HYSLOP, J. and REES, D., (1969),

A minimum principle for atomic systems allowing the use of discontinuous wave functions, Intern.J.Quantum Chem. Vol.3, pp.195-204.

A minimum principle for molecular systems allowing the use of discontinuous wave functions, Intern.J. Quantum Chem., Vol.4, pp.5-20.

HAMMER, P.C. and STROUD, A.H.,

Numerical evaluation of multiple integrals II, Ibid. 12(1958), pp.272-280.

HILDEBRAND, F.B.,

Introduction to numerical analysis, McGraw-Hill Book Company, Inc. 1956.

HURWITZ, H. and ZWEIFEL, P.F., (1959),

Numerical quadrature of Fourier transform integrals, MTAC, Vol.10, pp.140-149.

HURWITZ, H., PFEIFER, R.A. and ZWEIFEL, P.F., (1959),

Numerical quadrature of Fourier transform integrals II, MTAC, Vol.13, pp.87-90.

HYSLOP, J., (1972), *Private Communications.*

(1973), *A six-dimensional quadrature procedure*, Theoretical Chem.Acta., Vol. 31, pp.184-194.

IRI, M., MORIGUTI, S. and TAKASAWA, Y.,
On certain quadrature formulae, (in Japanese), Kokyinoku Res.Inst.Math.Sci., Kyoto Univ. 91(1970), pp.82-118.

JENKINS, L.D., *A note on the paper by A.M. Cohen and D.A. Gismalla on quadrature formulas for symmetric integrands*, Intern. J.Computer Math.(1985), Vol. 17, pp.207-209.

JOHNSON, W.W., *On Cartesian numbers; their history, computation and values to n=20*, Quart.J. Pure.Appl.Math. Vol.46(1915), pp.52-65.

KNOPP, K., *Infinite series and application*, Blackie and Sons Ltd., Glasgow, (1951).

KRYLOV, V.I.,(1962),
Approximate calculation of integrals, New York, Macmillan.

LEDERMANN, W., *Multiple integrals*, Routledge and Kegan Paul Ltd.

LEVIN, D., (1973),
Development of non-linear transformation for improving convergence of sequences, Intern.J.Computer Maths. Section B, Vol.3, pp.371-388.

LINDSEY, C.H. and Van der MEULEN, S.G.,
Informal introduction to Algol 68, North-Holland Publishing Company, Amsterdam-London.(1971).

LYNESS, J.N. *Symmetric integration rules for hypercubes error coefficients*, Math.Comp., 19(1965), pp.260-276.

" *Symmetric integration rules for hypercubes II: rules, projection and rule extension*, Ibid, 19(1965), pp.394-407.

" *Integration rules of hypercube symmetry over a certain spherically symmetric space*, Ibid, 19(1965), pp.471-476.

" *Symmetric integration rules for hypercubes III: construction of integration rules using null rules*, Ibid, 19(1965), pp.625-637.

" *Limits on the number of function evaluations required by certain high-dimensional integration rules of hypercube symmetry*, Ibid, 19(1965), pp.638-643.

MCNAMEE, J. and STENGER, F., *Construction of fully symmetric integration formulas*, Numer.Math. 10(1967), pp.327-344.

MORI, M., *An IMT-type double exponential formula for numerical integration*, Publ. RIMS, Kyoto Univ. 14(1978), pp.713-729.

" *Quadrature formulae obtained by variable transformations and the D-E rule*, J. of Comp. and App.Maths. 12 and 13 (1985), pp.119-130.

MORSE, P.M. and FESHBACH, H. (1953), *Methods of theoretical physics*, New York, McGraw-Hill.

MUROTA, K. and IRI, M., *Parameter turning and repeated application of the IMT-*

type transformation in numerical quadrature, Num. Maths. 38(1982), pp.347-363.

MYSOVSKIH, I.P.,

Proof of the minimality of the number of nodes in the cubature formulas for a hypersphere, Zh. Vychist.Mat. Mat.Fiz., 6(1966), pp.621-630 = U.S.S.R. Comp.Math.and Math.Phys. 6 (1966), No. 4, pp.15-27.

" *On the construction of cubature formulas with fewest nodes*, Dokh.Akad.Nauk. U.S.S.R., 178(1968), pp.1252-1254=Soviet Math.Dokyl. 9(1968), pp.277-280.

O'HARA, H. and SMITH, F.J.,

Error estimate in the Clenshaw-Curtis quadrature formula, Computer J. 12(1969), pp.179-182.

PATTERSON, T.N.L.,

The optimal addition of points to quadrature formulae, Math.Comp. 22(1968), pp.843-856.

PIERCE, W.H., *Numerical integration over spherical shells*, Math. Table Aids Comput. 11 (1957), pp.244-249.

PIESSENS, R., (1970),

Gaussian formulas for the integration of oscillatory functions, ZAMN, Vol. 50, pp.698-700.

PIESSENS, R. and HAEGEMANS, A. (1973),

Numerical calculation of Fourier transform integrals, Electron Lett., Vol. 9, No.5, pp.108-109.

RABINOWITZ, P. (1967),

Gaussian integration in the presence of singularity, SIAM J.Numer.Anal., Vol. 4, pp.191-201.

SAG, T.W. and SZEKERES, G.,

Numerical evaluation of high-dimensional integrals, Math.Comp. 18(1964), pp.245-253.

SCHWARTZ, C., *Numerical integration of analytic function*, J.Comp. Phys. 4(1969), pp.19-29.

Variational principle of integrals, J.Computer.Phys. 3(1969), pp.512-522.

SQUIRE, W., *Integration for engineers and scientists*, American Elsevier Publishing Company, Inc., New York (1970).

SQUIRE, W., *A quadrature method for finite intevals*, Int.J.Num. Math.Engng., 10(1976), pp.137-712.

" *A comment on quadrature in the presence of end point singularities*, Intern. J. Computer Maths. 7(1979), pp.239-241.

STENGER, F., *Integration formulae based on the trapezoidal formula*, J.Int.Math.Applic. 12(1973), pp.102-114.

STROUD, A.H., *Numerical integration formulas of degree two*, Math.Comp. 14(1960), pp.21-26.

STROUD, A.H. and SECREST, D.,

Gaussian quadrature formulas, Prentice-Hall Inc., Englewood Cliffs, N.J. (1966).

TALBOT, A., (1979),

The accurate numerical inversion of Laplace transform, Inst.Maths. Applics. 23, pp.97-120.

TAKAHSI, H. and MORI, M.,

Quadrature formulae obtained by variable transformations, Num.Math. 21(1973), pp.206-219.

" *Double exponential formulae for numerical integration,* Publ. Res.Inst.Math.Sci., Kyoto Univ. 9(1974), pp.721-741.

" *Error estimate in the numerical integration of analytic functions,* Res.Comput. Centre Univ. Tokyo, 3(1970), pp.41-108.

THA CHER, H.C. *Optimum quadrature formulas in s-dimension,* Maths. Table Aids Comput. 11(1957), pp.189-194.

" *An efficient composite formula for multi-dimensional quadrature,* Comm. ACM, 7(1964), pp.189-194.

TYLER, G.W. *Numerical integration of functions of several variables,* Canada J.Math. 5(1953), pp.393-412.

ZAREMBA, S.K., *The mathematical basis of Monte Carlo and Quasi-Monte Carlo methods,* SIAM Preview Vol.10 (1968), pp.303-314, MR 38# 1810.

APPENDIX

```
PROGRAM getest
BEGIN
    INT n,ip,ic,enn;
    REAL x,gw,c,gc0,gcw,gw0,capc;
    BOOL con;
    [1:5] REAL te;
    PROC psi=(INT n)REAL:
    BEGIN
       REAL s,gamma:=0.5772156649;
       IF n=1 THEN -gamma
       ELSE
          s:=-gamma;
          FOR i1 TO n-1 DO s PLUSAB 1.0/i1 OD;
          s
      FI
    END;

    PROC en=(INT n,REAL x)REAL:
    BEGIN
      IF ABS x >1.0 THEN
         INT nd:=8+ ENTIER(80/x);
         REAL x0,y0:=1.0;
         FOR i1 FROM 0 TO nd-1 DO
            x0:=(nd-i1)/(x+y0);
            y0:=(nd+n-i1-1)/(1.0+x0)
         OD;
         exp(-x)*(1.0/(x+y0))
      ELSE
       IF ABS x<1.0e-8 AND (n=0 OR n=1) THEN
          print((newline,"Singular E0 or E1",outline));
          0.0
       ELSE
         REAL s,xm:=-x,t:=1.0,tt;
         IF ABS x<1.0e-8 THEN
             s:=0.0
         ELSE
             s:=psi(n)-ln(x);
             tt:=s*10.0;
             FOR i1 TO n-1 DO s TIMESAB xm/i1 OD
          FI;
          (n/=1!s MINUSAB 1.0/(1.0-n));
          FOR i1 WHILE ABS tt>ABS s*1.0e-10 DO
             t TIMESAB xm/i1;
             (i1/=n-1!tt:=t/(i1-n+1);s MINUSAB tt)
          OD;
          s
       FI
       FI
    END;

    PROC coef=(INT m,n)REAL:
    BEGIN
       IF n=1 THEN 1.0
       ELIF m=0 THEN 1.0
       ELIF m=1 THEN n
       ELSE
      REAL s:=0.0,t:=1.0;
      FOR i TO (n<m!n!m) DO
         t TIMESAB (n-i+1)/i;
         s PLUSAB t*coef(m-i,i)
      OD;
      s
   FI
END;

 PROC curtclena=( PROC ( REAL ) REAL f, REAL a,b,prec,sc,econ,
      INT max,BOOL rel,REF BOOL con) REAL:
 BEGIN
    REAL sa:=1.0,sb:=0.0,h1:=b-a,w:=b,s1;
    REAL fac:=0.0;
   con:= FALSE ;
    INT ic:=1,icc;
   [0:max] REAL xa,wa,fa;
    BOOL outb;
   outb:= TRUE ;
    WHILE
      ic TIMESAB 2;
       IF ic=2 OR ic=4 THEN  TRUE
       ELSE
         con:=con OR  ABS (sa-sb)<=fac;
         ic<=max AND  NOT con
       FI
    DO
    BEGIN
      sa:=sb;
       IF ic=2 THEN xa[0]:=1.0;xa[1]:=0.0;xa[2]:=-1.0
       ELSE
          FOR i1 FROM icc BY -1 TO 1 DO xa[2*i1]:=xa[i1] OD ;
          FOR i1 TO icc DO xa[2*i1-1]:=cos(pi*(i1+i1-1)/ic) OD
       FI ;
       FOR i1 FROM 0 TO ic DO
       BEGIN
         s1:=0.0;
          FOR i2  FROM 0 TO ic DO
             IF i2=(i2%2)*2 THEN
               IF i2=0 OR i2=ic
                THEN s1 PLUSAB cos(pi*i1*i2/ic)/(1-i2*i2)/2
                ELSE s1 PLUSAB cos(pi*i1*i2/ic)/(1-i2*i2)
                FI
             FI  OD ;
         wa[i1]:=(i1=0 OR i1=ic!2!4)*s1/ic
       END  OD ;
       IF ic=2 THEN fa[0]:=f(w);fa[1]:=f(w-h1/2);fa[2]:=f(a)
       ELSE
          FOR i1 FROM icc BY -1 TO 1 DO fa[2*i1]:=fa[i1] OD ;
          FOR i1 TO icc DO fa[2*i1-1]:=f(w+(xa[2*i1-1]-1)*h1/2) OD
       FI ;
      sb:=0.0;
       FOR i1 FROM 0 TO ic DO sb PLUSAB wa[i1]*fa[i1] OD ;
      icc:=ic;
      (outb!printf(($l"ic="+3zd5x,"int="+d.9de+zd$,ic,sb*h1/2)))
    END  OD ;
    sb*h1/2
   END ;


  FORMAT for1=$318x"X"14x"1"20x"3"20x"10"20x"20"
               2155(5x+2zd.3d,4(5x+d.7de+zd)l)$;

  PROC ge=(REAL t)REAL:
  BEGIN
     INT uplim:=200;
     IF t<-0.991 THEN 0.0
     ELSE
     REAL enc,tt,s,z:=1.0/(1.0-t*t),s0,tw,tw0,gm,rem,ep:=1.0e-10;
     BOOL con:=FALSE,dir:=FALSE;INT m:=1;
  print((newline,"infinite series and application by knops",
      outline));
     enc:=en(2,z);
     tt:=1.0/z;
     s:=enc*tt;s0:=s;tw0:=s;
     FOR i1 TO uplim WHILE NOT con DO
        (i1=uplim!print((newline,"i1 limit reached in ge",outline)));
        IF dir THEN
           enc:=en(i1+2,z)
        ELSE
           enc:=(exp(-z)-z*enc)/(i1+1)
        FI;
        tt TIMESAB (2.0*i1-1)/2.0/i1/z;
        CO
        print((newline,"enc=",enc,space,"tt=",tt));
        CO
        tw:=enc*tt;
        s PLUSAB tw;
        gm:=ABS(tw/tw0);
        IF gm>1.0 THEN con:=FALSE
        ELSE
           rem:=ABS(tw)/(1.0-gm);
           con:=rem<ABS s *ep
        FI;
        CO
        print((space,space,"gm=",gm,outline));
        CO
        tw0:=tw
     OD;
     s/2.0
     FI
  END;


  PROC ge1=(REAL t)REAL:
  BEGIN
    REAL a0:=0.6034502,a1:=1,a2,s,tw,a3,a4,a5,gm,rem,ep:=1.0e-12;
    BOOL con:=FALSE;
    a3:=8.0*a1/3.0;a2:=3.0*a0;a4:=(10.0*a2-4.0*a0)/4.0;
    s:=a0+a1*t+a2*t*t+a3*t*t*t+a4*t*t*t*t;
    tw:=t*t*t*t;
     FOR i1 FROM 4 TO 100 WHILE NOT con DO
```

```
      (i1=100!print((newline,"i1 limit reached in ge1",outline)));
      a5:=((2.0*i1+4.0)*a3-(i1+1)*a1)/(i1+1);
      CO
      print((newline,"i1=",i1,space,"a=",a5,outline));
      CO
      tw TIMESAB t;
      s PLUSAB a5*tw;
      a1:=a2;a2:=a3;a3:=a4;a4:=a5;
      gm:=ABS(a4*t/a3);
      IF  gm>1.0 THEN con:=FALSE
      ELSE
         rem:=ABS(tw)/(1.0-gm);
         con:=rem<ABS s*ep
      FI
   OD;
   tw:=(1.0-t*t);
   s TIMESAB tw*tw*exp(-1.0/tw);
   s
END;

PROC ge2=(REAL t)REAL:
BEGIN
   REAL g0:=0.22199691,s1:=t,tw1:=1.0,s2,tw2,ts:=t*t,te2,te1;
   INT i1w,i2w;
   BOOL con1:=FALSE,con2:=FALSE;
   FOR i1 TO 100 WHILE NOT con1 DO
      tw1:=-tw1/i1;
      s2:=t;tw2:=t;
      FOR i2 TO 100 WHILE NOT con2 DO
         i2w:=i2;
         tw2 TIMESAB ts;
         te2:=coef(i2,i1)*tw2/(2*i2+1);
         s2 PLUSAB te2;
         con2:=ABS te2 <1.0e-8*ABS s2
      OD;
      IF i2w=100 THEN print((newline,"i2 loop not converged  te2="
                             te2,outline)) FI;
      te1:=s2*tw1;
      i1w:=i1;
      s1 PLUSAB te1;
      con1:= ABS te1<1.0e-8*ABS s1
   OD;
   print((newline,i1w,space,te1,outline));
   g0-s1
END;

PROC gec=(REAL c,t)REAL:
BEGIN
   INT uplim:=200;
   IF t<-0.991 THEN 0.0
   ELSE
   REAL enc,tt,s,z:=1.0/(1.0-t*t),s0,tw,tw0,gm,rem,ep:=1.0e-10;
   REAL cz:=c*z;
   BOOL con:=FALSE,dir:=FALSE;INT m:=1;
   enc:=en(2,cz);
   tt:=1.0/z;
   s:=enc*tt;s0:=s;tw0:=s;
   FOR i1 TO uplim WHILE NOT con DO
  (i1=uplim!print((newline,"i1 limit reach in gec",outline)));
      IF dir THEN
         enc:=en(i1+2,cz)
      ELSE
         enc:=(exp(-cz)-cz*enc)/(i1+1)
      FI;
      tt TIMESAB (2.0*i1-1)/2.0/i1/z;
      CO
      print((newline,"enc=",enc,space,"tt=",tt));
      CO
      tw:=enc*tt;
      s PLUSAB tw;
      gm:=ABS(tw/tw0);
      IF gm>1.0 THEN con:=FALSE
      ELSE
         rem:=ABS(tw)/(1.0-gm);
         con:=rem<ABS s *ep
      FI;
      CO
      print((space,space,"gm=",gm,outline));
      CO
      tw0:=tw
   OD;
   s/2.0
   FI
END;

PROC gcf0=(REAL x)REAL:
BEGIN
   REAL y:=1.0e-36;
   REAL xl:=-sqrt(1.0-c/ln(1.0/y));
   IF  x <xl THEN
      0.0
   ELSE
      exp(-c/(1.0-x*x))
   FI
END;

PROC setgc0=(REAL c, REF REAL gc0,g0) VOID :
BEGIN
   g0:=curtclena(gcf0,-1.0,0.0,1.0e-7,1.0,1.0,64,FALSE,con);
   gc0:=exp(c)*g0
END;

PROC gec1=(REAL c,t)REAL:
BEGIN
  REAL a0:=gc0,a1:=1,a2,s,tw,a3,a4,a5,gm,rem,ep:=1.0e-12;
  BOOL con:=FALSE;
  a2:=(4.0+2.0*c)*a0/2.0;
  a3:=(6.0+2.0*c)*a1/3.0;
  a4:=((8.0+2.0*c)*a2-4.0*a0)/4.0;
  s:=a0+a1*t+a2*t*t+a3*t*t*t+a4*t*t*t*t;
  tw:=t*t*t*t;
   FOR i1 FROM 4 TO 100 WHILE NOT con DO
```

```
        (i1=100!print((newline,"i1 limit reached in gec1",outline)));
        a5:=((2.0*i1+2.0+2.0*c)*a3-(i1+1)*a1)/(i1+1);
        CO
        print((newline,"i1=",i1,space,"a=",a5,outline));
        CO
        tw TIMESAB t;
        s PLUSAB a5*tw;
        a1:=a2;a2:=a3;a3:=a4;a4:=a5;
        gm:=ABS(a4*t/a3);
        IF  gm>1.0 THEN con:=FALSE
        ELSE
           rem:=ABS(tw)/(1.0-gm);
           con:=rem<ABS s*ep
        FI
     OD;
     tw:=(1.0-t*t);
     s TIMESAB tw*tw*exp(-c/tw);
     s
  END;

  PROC spexp=(REAL x)REAL:
  BEGIN
     IF x>-87.0 THEN
        exp(x)
     ELSE
        0.0
     FI
  END;

  PROC gn=(INT n,REAL t)REAL:
  BEGIN
    REAL tw,s,t2n;
    BOOL con:=FALSE;
    INT j1;
    t2n:=t**(2*n);
    s:=t;
    tw:=t;
    FOR i1 TO 200 WHILE NOT con DO
        twTIMESAB t2n/(-i1); j1:=i1;
        s PLUSAB tw/(2.0*n*i1+1);
        con:= ABS (tw/(2.0*n*i1+1))<=1.0e-8*ABS (s+1.0e-6)
    OD;
    (j1=100!print((newline,"series halted in gn",outline)));
     s*capc
  END;

  PROC gn1=(INT n,REAL t) REAL:
  BEGIN
    REAL tw,s,t2n,tw1;
    BOOL con:=FALSE;
    INT j1;
    t2n:=t**(2*n);
    tw:=1.0/2/n/t**(2*n-1);
     s:=tw;
    FOR i1 TO 100 WHILE NOT con DO
        tw1:=tw;
```

```
         tw TIMESAB (2.0*n*i1-1)/(-2.0)/n/t2n;
         s PLUSAB tw;j1:=i1;
         con:= ABS tw <= 1.0e-8*ABS(s+1.0e-8)
      OD;
      s TIMESAB exp(-t2n);
      (j1=100!print((newline,"series limit reached in gn1",outline)))
      (ABS tw> ABS tw1!print((newline,"series not asymptotic in gn1",
      outline)));
      print((newline,"s=",s,outline));
      capc*s
   END;

   PROC gn2=(INT n,REAL t) REAL:
   BEGIN
      REAL s,tw,tp;
      BOOL con :=FALSE;
      INT j1,n2:=n+n;
      tp:=1.0;
      FOR i1 TO n2 DO tp TIMESAB t OD;
      s:=t;
      tw:= t;
      FOR i1 FROM n2+1 BY n2 TO 200 WHILE NOT con DO
         tw TIMESAB n2/i1*tp;
         s PLUSAB tw;
         j1:=i1;
         con := ABS tw <= 1.0e-10 * ABS (s+1.0e-8)
      OD;
      s TIMESAB exp(-tp);
      (j1>=147!print((newline,"series limit reached in gn2",
      outline)));
     s*capc
   END;

  PROC gnq=(INT n,REAL t)REAL:
  BEGIN
     INT m;
     REAL capn:=exp(ln(38*ln(10.0))/2.0/n);
     CO
     print((newline,"capn=",capn,outline));
     CO

     PROC gngf=( REAL y) REAL:
     BEGIN
        REAL w,pr:=1.0;
        w:=(capn-t)/2.0*y+(capn+t)/2.0;
        FOR i1 TO 2*n DO pr TIMESAB w OD;
        exp(-pr)
     END;

     m:=9;
     [1:m]REAL wt,ab;
     wt:=(0.17944647,0.17656271,0.168004102,0.15404576,0.13513637,
          0.11188385,0.085036148,0.055459529,0.024148303);
     ab:=(0.0,0.17848418,0.35123176,0.51269054,0.65767116,0.78151400,
          0.88023915,0.95067552,0.99057548);
     REAL s:=wt[1]*gngf(ab[1]);
   FOR i1 FROM 2 TO m DO
      s PLUSAB wt[i1]*(gngf(ab[i1])+gngf(-ab[i1]))
   OD;
   s TIMESAB (capn-t)/2.0*capc
END;

PROC setcapc=(INT n) REAL:
BEGIN
   REAL ul;

   PROC gnf0=(REAL x)REAL:
   BEGIN
      exp(-(x**(n+n)))
   END;

   ul:=exp(ln(10.0*ln(10.0))/2.0/n);
   capc:=1.0/curtclena(gnf0,0.0,ul,1.0e-7,1.0,1.0,64,FALSE,con);
   print((newline,"capc=",capc,space,"for n=",n,outline));
   capc
END;

 PROC spgen=(REAL t,INT n)REAL:
 BEGIN
   REAL tt,wt;
       IF t<0.0 THEN
          tt:=-t
       ELSE
          tt:=t
       FI;
       IF tt<(n!1.5,1.0,1.0) THEN
          wt:=1.0-gn2(n,tt)
       ELSE
          wt:=gnq(n,tt)
       FI;
   wt
 END;

 PROC spfe=(PROC(REAL,REAL)REAL f,REAL a,b,t,INT n,REF REAL w1,
 w2)VOID:
 BEGIN
   REAL em:=(b-a)/2.0,ce:=(b+a)/2.0,w0;
   print((newline,"spfe    t=",t));
   w0:=em*spgen(t,n);
   print(("      w0=",w0));
   w1:=f((t>0.0!b-w0!a+w0),w0);
   print(("      w1=",w1));
   w2:=exp(-(t**(2*n)))
   ;print(("     w2=",w2,outline))
 END;

 PROC eint=(PROC(REAL,REAL)REAL f,REAL a,b,prec,INT n)REAL:
 BEGIN
   REAL sum1,sum2,quad1,quad2,tj,h0,h,w1,w2,odd;
   INT nn,npts;
   BOOL con;
   h0:=0.5;
   h:=h0;
   setcapc(n);
   spfe(f,a,b,0.0,n,w1,w2);
   sum1:=w1*w2;con:=FALSE;
   FOR i1 TO 100 WHILE NOT con DO
       spfe(f,a,b,i1*h,n,w1,w2);
       tj:=w1*w2;
       sum1 PLUSAB tj;
      spfe(f,a,b,-i1*h,n,w1,w2);
      tj:=w1*w2;
      sum1 PLUSAB tj;
       nn:=i1;
       con:=ABS(tj) < prec* ABS (0.5*sum1)
    OD;
   print((newline,"First search complete   ",nn,outline));
    IF nn=100 THEN print((newline,"h too small",outline)) FI;
    npts:=2*nn+1;
    quad1:=h*(b-a)*capc*sum1/2.0;
    print((newline,"npts=",npts,space,"quad1=",quad1,outline));
    con:=FALSE;
    FOR i1 FROM 2 TO 8 WHILE NOT con DO
       nnTIMESAB 2;
       h DIVAB 2.0;
       odd:=0.0;
       FOR i2 BY 2 TO nn-1 DO
          spfe(f,a,b,i2*h,n,w1,w2);
          odd PLUSAB w1*w2;
          spfe(f,a,b,-i2*h,n,w1,w2);
          odd PLUSAB w1*w2
       OD;
       sum2:=sum1+odd;
       quad2:=h*(b-a)*capc*sum2/2.0;
       npts:=2*nn+1;
       print((newline,"npts=",npts,space,"quad2=",quad2,outline));
       con:=(ABS(quad1-quad2)<ABS(prec*quad2));
       quad1:=quad2;
       sum1:=sum2
    OD;
    quad2
END;


PROC spge=(REAL t)REAL:
BEGIN
    REAL tt,g0:=0.22199691,wge;
    IF t<-1.0 THEN
       print((newline,"t<-1.0",outline));
       0.0
    ELIF t>1.0 THEN
       print((newline,"t>1.0",outline));
       0.0
    ELSE
       IF t>0.0 THEN tt:=-t
       ELSE tt:=t
       FI;
       IF tt<-0.5 THEN
```

```
            wge:=ge(tt)
         ELSE
            wge:=ge1(tt)
         FI;
         CO
         IF t>0.0 THEN
            (g0-wge)/g0
         ELSE
            (wge-g0)/g0
         FI
         CO
         wge/g0
      FI
   END;

   PROC gint=(PROC(BOOL,REAL)REAL f,REAL a,b,INT n,ip,ic)REAL:
   BEGIN
      REAL h,s,em,ce,t,g0:=0.22199691,w1,w2,h1,s1;

   PROC spf=(REAL t,REF REAL w1,w2) VOID:
   BEGIN
      REAL w0:=spge(t);
      w1:=f(t>0.0,em*w0);
      w2:=spexp(-1.0/(1.0-t*t));
      print((newline,"t=",t,space,"w1=",w1,space,"w2=",w2,space,
             "w0=",w0,outline))
   END;

      em:=(b-a)/2.0;
      ce:=(b+a)/2.0;
      h:=2.0/n;
      s:=0.0;
      FOR i1 TO n-1 DO
         t:=i1*h-1.0;
         spf(t,w1,w2);
         s PLUSAB w1*w2
      OD;
      s TIMESAB h*em/g0;
      CO
      h1:=h/(ip+1);
      s1:=0.0;
      FOR i1 TO ic*(ip+1)-1 DO
         t:=i1*h1-1.0;
         spf(t,w1,w2);
         s1 PLUSAB w1*w2
      OD;
      t:=ic*h-1.0;
      spf(t,w1,w2);
      s1 PLUSAB w1*w2/2.0;
      s PLUSAB w1*w2/2.0;
      FOR i1 FROM ic+1 TO n-ic-1 DO
         t:=i1*h-1.0;
         spf(t,w1,w2);
         s PLUSAB w1*w2
      OD;
      t:=(n-ic)*h-1.0;
```

```
    spf(t,w1,w2);
    s PLUSAB w1*w2/2.0;
    s TIMESAB h;
    s1 PLUSAB w1*w2/2.0;
    FOR i1 FROM 1 TO ic*(ip+1)-1 DO
       t:=(n-ic)*h+i1*h1-1.0;
       spf(t,w1,w2);
       s1 PLUSAB w1*w2
    OD;
    s1 TIMESAB h1;
    s PLUSAB s1;
    s TIMESAB em/g0;
    CO
    s
END;

 PROC spgec=(REAL c,t)REAL:
 BEGIN
    REAL tt,g0:=gw0,wge;
    IF t<-1.0 THEN
       print((newline,"t<-1.0",outline));
       0.0
    ELIF t>1.0 THEN
       print((newline,"t>1.0",outline));
       0.0
    ELSE
       IF t>0.0 THEN tt:=-t
       ELSE tt:=t
       FI;
       IF tt<-0.5 THEN
          wge:=gec(c,tt)
       ELSE
          wge:=gec1(c,tt)
       FI;
       CO
       IF t>0.0 THEN
          (g0-wge)/g0
       ELSE
          (wge-g0)/g0
       FI
       CO
       wge/g0
    FI
 END;

 PROC gintc=(PROC(BOOL,REAL)REAL f,REAL a,b,INT n,ip,ic)REAL:
 BEGIN
    REAL h,s,em,ce,t,g0:=gw0,w1,w2,h1,s1;

 PROC spf=(REAL t,REF REAL w1,w2) VOID:
 BEGIN
    REAL w0:=spgec(c,t);
    w1:=f(t>0.0,em*w0);
    w2:=spexp(-c/(1.0-t*t));
    print((newline,"t=",t,space,"w1=",w1,space,"w2=",w2,space,
          "w0=",w0,outline))
 END;

    em:=(b-a)/2.0;
    ce:=(b+a)/2.0;
    h:=2.0/n;
    s:=0.0;
    FOR i1 TO n-1 DO
       t:=i1*h-1.0;
       spf(t,w1,w2);
       s PLUSAB w1*w2
    OD;
    s TIMESAB h*em/g0;
    s
END;

 PROC spfc=(PROC(REAL)REAL f,REAL t,REF REAL w1,w2) VOID:
 BEGIN
    REAL w0:=spge(t);
    w1:=((ABS t -1.0)<1.0e-6!0.0!f(w0));
    w2:=((ABS t -1.0)<1.0e-6!0.0!spexp(-1.0/(1.0-t*t)));
    print((newline,"t=",t,space,"w1=",w1,space,"w2=",w2,space,
           "w0=",w0,outline))
 END;


 PROC out1=(REF [ ] REAL t,REF [ , ] REAL b,INT no) VOID:
 BEGIN
    FORMAT fur1=$+d.8de+zd3x$;
    INT nn:=no%2+1;
    newline(stand out);
    FOR i1 TO no DO
       printf((fur1,t[i1]));
       FOR i2 TO
          IF i1<6 AND i1<nn THEN i1-1
          ELIF i1>no-5 AND i1>=nn THEN no-i1
          ELSE 5
          FI
     : DO
         printf((fur1,b[i1-i2,i2]))
       OD;
      newline(stand out)
    OD
 END;

  PROC epal=( INT k, REF [ ] REAL t) REAL :
  BEGIN
     INT no:=2*k+1;
     REAL eps;
    eps:=1.0e-8;
    [0:no] REF [ ] REAL ep;
    [0:no] REF [ ] BOOL bp;
     INT ic:=0;
 l1:ep[ic]:= LOC [-1:no-ic-1] REAL ;
    bp[ic]:= LOC [-1:no-ic-1] BOOL ;
    ((ic PLUSAB 1)<=no! GOTO l1);
     FOR i1 FROM 0 TO no DO
```

```
   FOR i2 FROM -1 TO no-i1-1 DO bp[i1][i2]:= FALSE  OD  OD ;
   FOR i1 FROM 0 TO no DO ep[i1][-1]:=0.0 OD ;
   FOR i1 FROM 0 TO no-1 DO ep[i1][0]:=t[i1+1] OD ;
   FOR i1 TO no-1 DO  FOR i2 FROM 0 TO no-i1-1 DO
      IF bp[i2+1][i1-1] OR bp[i2][i1-1] OR bp[i2+1][i1-2]
      THEN ep[i2][i1]:=ep[i2+1][i1-2];bp[i2][i1]:= TRUE
      ELIF  ABS (ep[i2+1][i1-1]-ep[i2][i1-1])<1.0e-16
      THEN
         IF  ODD i1 THEN ep[i2][i1]:=1.0e+37;bp[i2][i1]:= TRUE
         ELSE ep[i2][i1]:=ep[i2+1][i1-2];bp[i2][i1]:= TRUE  FI
      ELSE
     ep[i2][i1]:=1.0/(ep[i2+1][i1-1]-ep[i2][i1-1])+ep[i2+1][i1-2]
      FI  OD  OD ;
  ep[0][2*k]
END ;

PROC eptable=( INT no, REF [ ] REAL t) REAL :
BEGIN
   INT nc:=(no+1)%2;
  [0:no] REF [ ] REAL ep;
  [0:no] REF [ ] BOOL bp;
  [1:no-2,1:nc] REAL b;
   INT ic:=0;
l1:ep[ic]:= LOC [-1:no-ic-1] REAL ;
  bp[ic]:= LOC [-1:no-ic-1] BOOL ;
  ((ic PLUSAB 1)<=no! GOTO l1);
   FOR i1 FROM 0 TO no DO
   FOR i2 FROM -1 TO no-i1-1 DO bp[i1][i2]:= FALSE  OD  OD ;
   FOR i1 FROM 0 TO no DO ep[i1][-1]:=0.0 OD ;
   FOR i1 FROM 0 TO no-1 DO ep[i1][0]:=t[i1+1] OD ;
   FOR i1 TO no-1 DO  FOR i2 FROM 0 TO no-i1-1 DO
      IF bp[i2+1][i1-1] OR bp[i2][i1-1] OR bp[i2+1][i1-2]
      THEN ep[i2][i1]:=ep[i2+1][i1-2];bp[i2][i1]:= TRUE
      ELIF  ABS (ep[i2+1][i1-1]-ep[i2][i1-1])<1.0e-16
      THEN ep[i2][i1]:=1.0e+37;bp[i2][i1]:= TRUE
      ELSE
     ep[i2][i1]:=1.0/(ep[i2+1][i1-1]-ep[i2][i1-1])+ep[i2+1][i1-2]
      FI  OD  OD ;
   FOR i1 TO nc DO
   FOR i2 TO no-2*i1 DO
     b[i2,i1]:=ep[i2-1][2*i1] OD  OD ;
   IF ip>=2 THEN out1(t,b,no) FI ;
   FOR i1 FROM 2 TO nc DO  FOR i2 TO no-i1-i1 DO
     b[i2,i1]:=epal(1,b[i2:i2+2,i1-1]) OD  OD ;
   IF ip>=1 THEN out1(t,b,no) FI ;
  (ip=-1!print((newline,b[1,nc])));
  b[1,nc]
END ;


PROC levinv=( INT k, REF [ ] REAL acap,asm) REAL :
BEGIN
   INT nk:=k+1,fac:=1;
   REAL s1:=0.0,s2:=0.0,w,ww;
   FOR i1 TO nk DO
   BEGIN
```

```
            w:=i1/nk;
            ww:=1.0;
             FOR i2 TO k-1 DO ww:=ww*w OD ;
            w:=fac*ww*(asm[i1+1]-asm[i1])/asm[i1]/asm[i1+1];
            s2 PLUSAB w;
            s1 PLUSAB w*acap[i1];
            fac:=-fac*(nk-i1)%i1
          END  OD ;
         s1/s2
       END ;


   PROC insp=(REAL y)REAL:
     BEGIN
        REAL s:=y,t:=y;
        FOR i FROM 2 TO 30 WHILE ABS t > ABS s*1.0e-8 DO
        t TIMESAB y; s PLUSAB t/i
        OD;
        s
      END;
     PROC f1=(BOOL t0,REAL x) REAL:((t0!exp(1.0-x)!exp(-1.0+x)));

     PROC f11=(REAL x)REAL: (exp(-x));

 CO
     PROC fe1=(REAL x,d)REAL:(exp(-x));
 CO
  PROC fe1=(REAL x,d)REAL:
    BEGIN
         REAL w;
         IF x < 0.0 THEN
         cos(pi*x)/sqrt(2.0-d)
         ELSE
         cos(pi*x)/sqrt(d)
       FI
   END;
     PROC f2=(BOOL t0,REAL x )REAL:((t0!sin(pi/4-x)!sin(x)));

     PROC f22=(REAL x) REAL:(sin(x));

     PROC f3=(BOOL t0,REAL x )REAL:
     BEGIN
        REAL w:= ln((t0!x!1.0-x));
        w*w/(1.0+(t0!x*x!(1.0-x)*(1.0-x)))
     END;

 CO
     PROC fe3=(REALx,d)REAL:
     BEGIN
        REAL w;
        IF x<0.5 THEN
           w:=ln(d);
           w*w/(1.0+x*x)
        ELSE
           w:=ln(1.0-d);
           w*w/(1.0+x*x)
      FI
   END;

PROC fe3=(REAL x,d)REAL:
BEGIN
  REAL w,w1;
  IF x<0.5 THEN
     ln(ln(1.0/x))/(1.0+x)/(1.0+x)
    ELIF d < 1.0e-02
      THEN ln(insp(d))/(2.0-d)/(2.0-d)
  ELSE
     ln(ln(1.0/(1.0-d)))/(2.0-d)/(2.0-d)
  FI
END;

CO
PROC fe3=(REAL x,d)REAL:
  BEGIN
     REAL w,w1;
        IF x <0.5 THEN
       insp(1.0-cos(d))
         ELSE
        ln(1.0-cos(x))
        FI
  END;

   PROC f33=(REAL x)REAL:
   BEGIN
      REAL w:=ln(x);
      w*w/(1.0+x*x)
   END;

   PROC f3c=(REAL t) REAL:
   BEGIN
      REAL w1,w2;
      spfc(f33,t,w1,w2);
      w1*w2
   END;

   PROC f4=(BOOL t0,REAL x)REAL:
   BEGIN
      1.0/sqrt(x)/sqrt(1.0-x)
   END;

      CO
   printf(for1);
   FOR i2 TO 55 DO
      x:=(i2<10!i2*0.1!(i2-10)*0.2+1.0);
      printf(x);
      FOR i3 TO 4 DO
         n:=(i3!1,3,10,20);
         printf(en(n,x))
      OD
   OD;
   printf(for1);
   FOR i2 FROM 0 TO 20 DO
```

```
     x:=i2*0.01;
     printf(x);
     FOR i3 TO 4 DO
        n:=(i3!1,3,10,20);
        printf(en(n,x))
     OD
  OD;
     CO
  FOR i1 FROM 0 TO 20 DO
     x:=-1.0+i1*0.05;
     gw:=ge(x);
     print((newline,x,space,space,gw,outline))
  OD;
  CO
  c:=1.0;
  FOR i1 FROM 0 TO 20 DO
     x:=-1.0+i1*0.05;
     gw:=gec(c,x);
     print((newline,x,space,gw,outline))
  OD;
  setgc0(c,gc0,gw0);
  FOR i1 FROM 20 BY -1 TO 5 DO
     x:=-1.0+i1*0.05;
     gw:=gec1(c,x);
     print((newline,x,space,gw,outline))
  OD;
  c:=2.0;
  FOR i1 FROM 0 TO 20 DO
     x:=-1.0+i1*0.05;
     gw:=gec(c,x);
     print((newline,x,space,space,gw,outline))
  OD;
  print((newline,"True Value=",exp(1.0)-1.0,outline));
  curtclena(f11,-1.0,0.0,1.0e-7,1.0,1.0,64,FALSE,con);
  setgc0(c,gc0,gw0);
  FOR i1 FROM 20 BY -1 TO 0 DO
     x:=-1.0+i1*0.05;
     gw:=gec1(c,x);
     print((newline,x,space,space,gw,outline))
  OD;
 FOR i1 FROM -20 TO 20 DO
     x:=i1*0.05;
     print((newline,"x=",x,space,space,spge(x),outline))
  OD;
  c:=2.0;
  setgc0(c,gc0,gw0);
  FOR i1 FROM -20 TO 20 DO
     x:=i1*0.05;
     print((newline,"x=",x,space,space,spgec(c,x),outline))
  OD;
  FOR i1 TO 8 DO
     c:=1.0+i1*0.25;
     print((newline,"c=",c,outline));
     setgc0(c,gc0,gw0);
     enn:=1;
     FOR i2 TO 5 DO
```

```
      enn TIMESAB 2;
      te[i2]:=gintc(f1,-1.0,1.0,enn,ip,ic);
      print((newline,i2, space,te[i2],outline))
   OD;
   eptable(5,te);
   enn:=1;
   FOR i2 TO 5 DO
      enn TIMESAB 2;
      te[i2]:=gintc(f3,0.0,1.0,enn,ip,ic);
      print((newline,i2,space,te[i2],outline))
   OD;
   eptable(5,te)
OD;
   ic:=1;
n:=2;
print((newline,"True value=",2.350402389,outline));
curtclena(f11,-1.0,1.0,1.0e-7,1.0,1.0,64,FALSE,con);
FOR i1 TO 5 DO
  ip:=n;
   n TIMESAB 2;
   print((newline,gint(f1,-1.0,1.0,n,ip,ic),outline))
OD;
n:=2;
print((newline,"True value=",0.2928932186,outline));
FOR i1 TO 5 DO
   ip:=n;
   n TIMESAB 2;
   print((newline,gint(f2,0.0,pi/4.0,n,ip,ic),outline))
OD;
n:=2;
print((newline,"True value=",1.9378932186,outline));
curtclena(f3c,-1.0,1.0,1.0e-7,1.0,1.0,64,FALSE,con);
FOR i1 TO 5 DO
   ip:=n;
   n TIMESAB 2;
   print((newline,gint(f3,0.0,1.0,n,ip,ic),outline))
OD;
n:=2;
FOR i1 TO 5 DO
   ip:=n;
   n TIMESAB 2;
   print((newline,gint(f4,0.0,1.0,n,ip,ic),outline))
OD;
FOR i1 FROM 1 TO 3 DO
  setcapc(i1);
  FOR i2 FROM 0 TO 140 DO
     x:=i2*0.05;
     gw:=gnq(i1,x);
     print((newline,"x=",x,space,"gnq=",gw,outline))
  OD;
   print((newline,"i1=",i1,outline));
   FOR i2 FROM 0 TO (i1!80,60,40) DO
      x:=i2 *0.05;
      gw:=gn(i1,x);
      print((newline,"x=",x,space,"gn=",gw,outline))
   OD;
```

```
      FOR i2 FROM 140 BY -1 TO (i1!80,40,35) DO
         x:=i2*0.05;
         gw:=gn1(i1,x);
         print((newline,"x=",x,space,"gn1=",gw,outline))
      OD;
      FOR i2 FROM 0 TO (i1!100,40,40) DO
         x:=i2*0.05;
         gw:=gn2(i1,x);
         print((newline ,"x=",x,space,"gn2=",gw,outline))
      OD
   OD;
   print((newline,eint(fe1,-1.0,1.0,1.0e-7,1),outline));
 CO
   print((newline,eint(fe3,0.0,1.0,1.0e-7,1),outline));
   SKIP
END
FINISH
```

(III)

```
PROGRAM twodint
BEGIN
   REAL h,s1,al,be,w1,w2,w3,c1;
   INT en,ie,count:=0,em;
   BOOL con ;
   h:=0.25;
   en:=5;
   em:=8;

PROC tanh=(REAL x)REAL:
BEGIN
   REAL w;
   IF x<0.0 THEN
      w:=exp(x+x);
      -1.0+2.0*w/(1.0+w)
   ELSE
      w:=exp(-x-x);
      1.0-2.0*w/(1.0+w)
   FI
END;

PROC sech=(REAL x)REAL:
BEGIN
   REAL w;
   IF x<0.0 THEN
      w:=exp(x);
      2.0*w/(w*w+1.0)
   ELSE
      w:=exp(-x);
      2.0*w/(1.0+w*w)
   FI
END;

OP P = (REAL x,INT n)REAL:
BEGIN
   REAL p:=1.0;
   FOR i1 TO n DO p TIMESAB x OD;
   p
END;

PROC f1=(REAL x,y)REAL:
BEGIN
    count PLUSAB 1;
    REAL w,w1;
           w:=(1.0+x)/2.0;
           w1:=(1.0+y)/2.0;
      ABS (x*x+y*y-0.25)
END;

PROC g=(REAL th)REAL:
BEGIN
   REAL al,be,w1,w2,w3,w4;
   al:=ie*h*cos(th);
   be:=ie*h*sin(th);
   w1:=al P en;
   w2:=be P en;
   w3:=sech(w1);
   w4:=sech(w2);
 en*en*w3*w3*w4*w4*f1(tanh(w1),tanh(w2))*(al P (en-1))*(be P (en-1))
END;
PROC curtclena=( PROC ( REAL )REAL f,REAL a,b,prec,sc,econ,INT max,
 BOOL rel,REF BOOL con)REAL:
BEGIN
   REAL sa:=1.0,sb:=0.0,h1:=b-a,w:=b,s1;
   REAL fac:=0.0;
   con:= FALSE;
   INT ic:=1,icc;
   [0:max] REAL xa,wa,fa;
   BOOL outb;
   outb:= TRUE;
   WHILE
      ic TIMESAB 2;
      IF ic=2 OR ic=4 THEN TRUE
      ELSE
        con:=con OR ABS (sa-sb)<=fac;
        ic<=max AND NOT con
      FI
   DO
   BEGIN
     sa:=sb;
      IF ic=2  THEN xa[0]:=1.0;xa[1]:=0.0;xa[2]:=-1.0
      ELSE
         FOR i1 FROM icc BY -1 TO 1 DO xa[2*i1]:=xa[i1] OD;
         FOR i1 TO icc DO xa[2*i1-1]:=cos(pi*(i1+i1-1)/ic) OD
      FI;
      FOR i1 FROM 0 TO ic DO
      BEGIN
        s1:=0.0;
         FOR i2 FROM 0 TO ic DO
            IF i2=(i2%2)*2 THEN
              IF i2=0 OR i2=ic
               THEN s1 PLUSAB cos(pi*i1*i2/ic)/(1-i2*i2)/2
               ELSE s1 PLUSAB cos(pi*i1*i2/ic)/(1-i2*i2)
               FI
            FI
       OD;
       wa[i1]:=(i1=0 OR i1=ic!2!4)*s1/ic
      END OD;
      IF ic=2 THEN fa[0]:=f(w);fa[1]:=f(w-h1/2);fa[2]:=f(a)
      ELSE
         FOR i1 FROM icc BY -1 TO 1 DO fa[2*i1]:=fa[i1] OD;
         FOR i1 TO icc DO fa[2*i1-1]:=f(w+(xa[2*i1-1]-1)*h1/2) OD
      FI;
      sb:=0.0;
       FOR i1 FROM 0 TO ic DO sb PLUSAB wa[i1]*fa[i1] OD;
       icc:=ic;
         (outb!printf(($l"ic="+3zd5x,"int="+d.9de+zd$,ic,sb*h1/2.0)))
        END OD;
         sb*h1/2.0
      END;

  PROC g1=(REAL x)REAL:(exp(-x));
 curtclena(g1,0.0,2.0*pi,1.0e-06,1.0,1.0,64,TRUE,con);
 print((newline,"I=",1.0-exp(-2.0*pi),outline));
 FOR i2 TO 3 DO
   ie:=5*i2;
 FOR i1 FROM 0 TO 50 DO
    w1:=i1/50*2.0*pi;
    print((newline,w1,space,g(w1),outline))
 OD
 OD;
 s1:=0.0;
   FOR i1 TO em-1 DO
   ie:=i1;
s1 PLUSAB curtclena(g,0.0,2.0*pi,1.0e-6,1.0,1.0,128,TRUE,con)*ie
   OD;
   ie:=em;
s1 PLUSAB curtclena(g,0.0,2.0*pi,1.0e-6,1.0,1.0,128,TRUE,con)*em;
   s1 TIMESAB h*h;
   print((newline,"count=",count,outline));
   print((newline,"s1=",s1,outline))
   END
   FINISH
```

```
PROGRAM twotrap
BEGIN
   INT count;

   PROC tanh=(REAL x)REAL:
   BEGIN
      REAL w:=exp(x);
      (w-1.0/w)/(w+1.0/w)
   END;

   PROC sech=(REAL x)REAL:
   BEGIN
      REAL w:=exp(x);
      2.0/(w+1.0/w)
   END;

   PROC tint2=(PROC(REAL,REAL,REAL,REAL)REAL f,REAL a1,b1,a2,b2,h1,h2,
               REAL prec,INT n)REAL:
   BEGIN
     REAL s:=0.0,daph:=0.0,gw,ld,rd,ep:=1.0e-9,val,h11,h22,h15,h25,
          caph,ulim,llim,fmax;
     INT np,flim,iw1,iw2,nsub,jw1,jw2;
      np:=1000;flim:=50;nsub:=1;
      [-np:np]REAL lar,rar;
      BOOL dontrib,girst,dlog,out2;
      out2:=TRUE;

      PROC floc=(REAL al,be)REAL:
      BEGIN
         REAL alw,bew,alw1,bew1,ex,ey,alf,bet,xx,yy,sw1,sw2;
         REAL dal,dbe,exa,exb,cal,cbe;
         alw:=1.0;bew:=1.0;
         FOR i1 TO n-1 DO
            alw TIMESAB al;
            bew TIMESAB be
         OD;
         alw1:=alw*al;
         bew1:=bew*be;
         ex:=tanh(alw1);
         ey:=tanh(bew1);
         exa:=exp(alw1);
         exb:=exp(bew1);
         dal:=(b1-a1)/(1.0+exa*exa);
         dbe:=(b2-a2)/(1.0+exb*exb);
         cal:=(b1-a1)/(1.0+1.0/exa/exa);
         cbe:=(b2-a2)/(1.0+1.0/exb/exb);
         xx:=(b1-a1)*ex/2.0+(b1+a1)/2.0;
         yy:=(b2-a2)*ey/2.0+(b2+a2)/2.0;
         sw1:=sech(alw1);
         sw2:=sech(bew1);
         alw*bew*sw1*sw1*sw2*sw2*f(cal,cbe,dal,dbe)
      END;

      PROC scan=(REAL h0,bet0,REF REAL lend,rend,fmax)VOID:
      BEGIN
         REAL caph,fw,ep:=1.0e-9;
      BOOL contrib,first,clog,out1,out2;
      INT elim:=100;
      fmax:=0.0;
      out1:=TRUE;
      out2:=TRUE;
      (out1!print((newline,"bet0=",bet0,"     ")));
      contrib:=TRUE;caph:=0.0;first:=FALSE;out1:=TRUE;
      TO elim WHILE contrib DO
         caph PLUSAB h0;
         fw:=floc(caph,bet0);
         (ABS fw > fmax!fmax:=ABS fw);
         (out2!print((newline,"caph=",caph,"   fw=",fw,outline)));
         s PLUSAB fw;
         contrib:=NOT(first AND (clog:=ABS(fw)<=ep));
         first:=clog
      OD;
      rend:=caph;
      caph:=0.0;contrib:=TRUE;
      first:=FALSE;
      TO elim WHILE contrib DO
         caph MINUSAB h0;
         fw:=floc(caph,bet0);
         (ABS fw > fmax!fmax:=ABS fw);
         s PLUSAB fw;
         contrib:=NOT(first AND (clog:=ABS(fw)<=ep));
         first:=clog
      OD;
      lend:=caph;
      (out1!print(("rend=",rend,"    lend=",lend,outline)))
   END;

   s:=floc(0.0,0.0);
   dontrib:=TRUE;
   girst:=FALSE;
   scan(h2,0.0,ld,rd,fmax);
   lar[0]:=ld;rar[0]:=rd;iw1:=0;
   TO flim WHILE dontrib DO
      daph PLUSAB h1;
      gw:=floc(0.0,daph);
      s PLUSAB gw;
      scan(h2,daph,ld,rd,fmax);
      dontrib:=NOT(girst AND (dlog:=ABS(fmax)<=ep));
      girst:=dlog;
      iw1 PLUSAB 1;
      (out2!print((newline,"iw1=",iw1,outline)));
      lar[iw1]:=ld;
      rar[iw1]:=rd
   OD;
   ulim:=daph;
   (out2!print((newline,"ulim=",ulim,outline)));
   daph:=0.0;
   dontrib:=TRUE;
   girst:=FALSE;
   iw2:=0;
   TO flim WHILE dontrib DO
      daph MINUSAB h1;
      gw:=floc(0.0,daph);
      s PLUSAB gw;
      scan(h2,daph,ld,rd,fmax);
      dontrib:=NOT(girst AND (dlog:=ABS(fmax)<=ep));
      girst:=dlog;
      iw2 MINUSAB 1;
      (out2!print((newline,"iw2=",iw2,outline)));
      lar[iw2]:=ld;
      rar[iw2]:=rd
   OD;
   llim:=daph;
   (out2!print((newline,"llim=",llim,outline)));
   print((newline,"End of first scan",outline));
   val:=n*n*(b1-a1)*(b2-a2)*h1*h2*s;
   print((newline,"First value=",val,outline));
   h11:=h1;h22:=h2;
   TO nsub DO
      h15:=h11;
      h25:=h22;
      h11 DIVAB 2.0;
      h22 DIVAB 2.0;
      caph:=h22;
      WHILE
         gw:=floc(caph,0.0);
         s PLUSAB gw;
         caph PLUSAB h25;
         caph<rar[0]
      DO SKIP OD;
      caph:=-h22;
      WHILE
         gw:=floc(caph,0.0);
         s PLUSAB gw;
         caph MINUSAB h25;
         caph>lar[0]
      DO SKIP OD;
      daph:=h11;jw1:=0;
      WHILE daph<ulim DO
         gw:=floc(0.0,daph);
         s PLUSAB gw;
         scan(h22,daph,ld,rd,fmax);
         jw1 PLUSAB 1;
         (out2!print((newline,"jw1=",jw1,outline)));
         FOR i1 FROM iw1 BY -1 TO jw1+jw1-1 DO
            lar[i1+1]:=lar[i1];
            rar[i1+1]:=rar[i1]
         OD;
         lar[jw1+jw1-1]:=ld;
         rar[jw1+jw1-1]:=rd;
         (out2!print((newline,"Check positive lar and rar",newline,
                lar[0:iw1+1],newline,rar[0:iw1+1],outline)));
         daph PLUSAB h11;iw1 PLUSAB 1;
         CO New intermediate line done
            onto  corresponding existing line
            for fill in CO
         caph:=h22;
         WHILE
```

```
            gw:=floc(caph,daph);
            s PLUSAB gw;
            caph PLUSAB h25;
            caph<rar[jw1+jw1]
         DO SKIP OD;
         caph:=-h22;
         WHILE
            gw:=floc(caph,daph);
            s PLUSAB gw;
            caph MINUSAB h25;
            caph>lar[jw1+jw1]
         DO SKIP OD;
         daph PLUSAB h11
      OD;
      daph:=-h11;
      jw2:=0;
      WHILE daph>llim DO
         gw:=floc(0.0,daph);
         s PLUSAB gw;
         scan(h22,daph,ld,rd,fmax);
         jw2 MINUSAB 1;
         (out2!print((newline,"jw2=",jw2,outline)));
         FOR i1 FROM iw2 TO jw2+jw2+1 DO
            lar[i1-1]:=lar[i1];
            rar[i1-1]:=rar[i1]
         OD;
         lar[jw2+jw2+1]:=ld;
         rar[jw2+jw2+1]:=rd;
         (out2!print((newline,"Check negative lar and rar",newline,
               lar[iw2-1:0],newline,rar[iw2-1:0],outline)));
         daph MINUSAB h11;
         iw2 MINUSAB 1;
         caph:=h22;
         WHILE
            gw:=floc(caph,daph);
            s PLUSAB gw;
            caph PLUSAB h25;
            caph<rar[jw2+jw2]
         DO SKIP OD;
         caph:=-h22;
         WHILE
            gw:=floc(caph,daph);
            s PLUSAB gw;
            caph MINUSAB h25;
            caph>lar[jw2+jw2]
         DO SKIP OD;
         daph MINUSAB h11
      OD;
      val:=n*n*(b1-a1)*(b2-a2)*h11*h22*s/4.0;
      print((newline,"Val=",val,outline))
   OD;
   val
END;

PROC f1=(REAL cx,cy,dx,dy)REAL:
BEGIN
```

```
      REAL w:=cx*cx*cy*cy;
      count PLUSAB 1;
      CO
      (ABS w<1.0e-5!
      1.0+w/2.0!
      1.0/sqrt(1.0-w))
      CO
      1.0/(dx+dy-dx*dy)
   END;

   PROC f2=(REAL cx,cy,dx,dy) REAL:
   BEGIN
       REAL w,w1;
         count PLUSAB 1;
         w:=-1.0+cx;
         w1:=-1.0+cy;
1.0/sqrt(cx*cy)
   END;

   count:=0;
   print((newline,tint2(f2,0.0,1.0,0.0,1.0,0.55,0.55,1.0e-6,1),
   outline));
   print((newline,"count=",count,outline))
END
FINISH
```

```
PROGRAM twaint
BEGIN
   INT n,m,nss,ns,nsw,uip,intger,ifail,ia,iw;
  m:=9;
  n:=18;
  nss:=19;
  nsw:=18;
  ns:=5;
  uip:=3;
  LONG REAL x1,y1,z1,v1,u1;
 [1:n] LONG REAL ex ;
  [1:n] INT col;

    FORMAT ft2=$14(+d.19de+zd3x)$;
   MODE  EFPROC = PROC ( REF [ ] LONG  REAL ,INT,INT) LONG  REAL ;
   MODE  ARPROC = PROC ( REF [ ] LONG  REAL ,INT) LONG  REAL ;
  [1:m+n] LONG  REAL b,bb;
  [1:m] LONG REAL ae;
  [1:nss,1:2] INT pair;
  LONG REAL w1,t ;
  [1:n] LONG REAL r,e;
 PROC (REF [,] LONG REAL,REF INT,REF INT,REF [] LONG REAL,REF []
 LONG REAL,REF INT)VOID f02aaf=ALIEN"f02aaf";
  ia:=18;
intger:=18;
  ifail:=0;
  [1:n,1:n] LONG REAL u;
PROC tp=(REF [,] LONG REAL a,u)VOID:
  BEGIN
  [1:n,1:n] LONG REAL a,u;
  FOR i TO n DO
  FOR j TO n DO
  u[i,j]:=LONG 0.0;
  FOR k TO n DO
  u[i,j] PLUSAB a[i,k]*a[j,k]
  OD
  OD
 OD
  END;
    PROC lgaussp=( REF [,] LONG  REAL a, REF [ ] LONG  REAL b,x) VOID
    BEGIN
       INT n:= UPB x,ml;
     LONG REAL ep= LONG 1.0e-18;
      BOOL cont,fout;
      fout:= TRUE ;
      cont:=TRUE;
       LONG  REAL max,w;
      print((newline,"Enter lgaussp",newline,"n=",n,newline));
    CO
  (n=18!tp(a,u);
    f02aaf(u,ia,n,r,e,ifail);
    FOR i TO n DO
    print((newline,"r[i]=",r[i],newline)) OD);
        CO
       FOR r TO n-1  WHILE cont DO
      max:= ABS a[r,r];ml:=r;
     CO
    (n=18!iw:=col[ml];
      col[ml]:=col[r];
      col[r]:=iw;
      print((newline,"col[r]=",col[r],outline)));
      CO
       FOR i1 FROM r+1 TO n DO
          IF  ABS a[i1,r]>max THEN
             max:= ABS a[i1,r];ml:=i1
          FI
       OD ;
       IF ml/=r THEN
          FOR i1 FROM r TO n DO
            w:=a[ml,i1];a[ml,i1]:=a[r,i1];a[r,i1]:=w
          OD ;
         w:=b[ml];b[ml]:=b[r];b[r]:=w
       FI ;
     IF ABS a[r,r]<= ep THEN
         print((newline,"Singular set",outline));
         FOR i1 TO n DO x[i1]:= LONG 0.0 OD;
         cont:=FALSE
      ELSE
       FOR i1 FROM r+1 TO n DO
         w:=-a[i1,r]/a[r,r];
          FOR i2 FROM r TO n DO a[i1,i2] PLUSAB w*a[r,i2] OD ;
         b[i1] PLUSAB w*b[r]
       OD
       FI
    OD ;
    IF ABS a[n,n]<= ep THEN
      print((newline,"Singular set",newline));
      FOR i1 TO n DO x[i1]:= LONG 0.0 OD
    ELSE
   x[n]:=b[n]/a[n,n];
    FOR i1 FROM n-1 BY -1 TO 1 DO
      x[i1]:=b[i1];
       FOR i2 FROM i1+1 TO n DO x[i1] MINUSAB x[i2]*a[i1,i2] OD ;
      x[i1] DIVAB a[i1,i1]
    OD
     FI
END ;

PROC lndnewt=( ARPROC f, REF [ ] LONG REAL xg, LONG REAL err,
h) VOID:
BEGIN
    INT n:= UPB xg,nn,stepno,count;
   [1:n,1:n] LONG  REAL a;
   [1:n] LONG  REAL b,xgg,xggg,fg;
   BOOL fout;
     fout:=TRUE;
   stepno:=18;
   count:=0;
   nn:=n%3+1;
   FORMAT ft2=$14(+d.19de+zd3x)$;
    FORMAT ft1=$ln(nn)(+d.18de+zd,2(+d.18de+zd)l)$;
```

```
        FOR i1 TO n DO xgg[i1]:=xg[i1]+ LONG 1.0 OD ;
        WHILE
           BOOL bb:= FALSE ;
          count PLUSAB 1;
           FOR i1 TO n WHILE  NOT bb DO
              IF  ABS (xg[i1]-xgg[i1])>err* ABS xgg[i1]
              THEN bb:= TRUE  FI
           OD ;
          bb AND count<=stepno
        DO
          xgg:=xg;xggg:=xg;
           FOR i1 TO n DO fg[i1]:=f(xgg,i1) OD ;
          print((newline,"aet=",ae));
           FOR i2 TO n DO
             xggg[i2] PLUSAB h;
              FOR i1 TO n DO a[i1,i2]:=(f(xggg,i1)-fg[i1])/h OD ;
             xggg[i2] MINUSAB h
           OD ;
           FOR i1 TO n DO
             b[i1]:=-fg[i1];
              FOR i2 TO n DO b[i1] PLUSAB a[i1,i2]*xgg[i2] OD
           OD ;
          lgaussp(a,b,xg);
          print((newline,"ae=",ae))
        OD
     END ;


     EFPROC ef;
        PROC ce=( INT n,m) LONG  REAL :
        BEGIN
           REAL rn,rm;
          rn:=n+1;rm:=m+1;
  IF ODD n OR ODD m THEN
        LONG 0.0
      ELSE
      LONG 4.0/ LENG rn/LENG rm
     FI
        END ;

             ef:=( REF [ ] LONG  REAL xd,INT i1,i2) LONG  REAL :
              BEGIN
                IF i1<=nss THEN
                 LONG  REAL p:= LONG 1.0;
                 INT ie,ja,en,em;
                ie:=2*i2-1;ja:=2*i2;
                en:=pair[i1,1];em:=pair[i1,2];
                 FOR i3 TO en DO p TIMESAB xd[ie] OD ;
                 FOR i3 TO em DO p TIMESAB xd[ja] OD ;
                p
CO
ELIF i1=7 AND i2=2 THEN LONG 1.0
  ELIF i1=7 AND i2=3 THEN -LONG 1.0
ELIF i1=8 AND i2=1 THEN LONG 1.0
   ELIF i1=8 AND i2=4 THEN - LONG 1.0
CO
```

```
 ELIF i1=20 AND i2=1 THEN LONG 1.0
   ELIF i1=20 AND i2=2 THEN -LONG 1.0
   ELIF i1=21 AND i2=3 THEN LONG 1.0
   ELIF i1=21 AND i2=4 THEN -LONG 1.0
   ELIF i1=22 AND i2=5 THEN LONG 1.0
ELIF i1=22 AND i2=6 THEN -LONG 1.0
  ELIF i1=23 AND i2=7 THEN LONG 1.0
  ELIF i1=23 AND i2=8 THEN -LONG 1.0
  ELIF i1=24 AND i2=9 THEN LONG 1.0
  ELIF i1=24 AND i2=1 THEN -LONG 1.0
  ELIF i1=25 AND i2=3 THEN LONG 1.0
  ELIF i1=25 AND i2=5 THEN -LONG 1.0
  ELIF i1=26 AND i2=6 THEN LONG 1.0
  ELIF i1=26 AND i2=7 THEN -LONG 1.0
  ELIF i1=27 AND i2=8 THEN LONG 1.0
  ELIF i1=27 AND i2=2 THEN -LONG 1.0
  ELSE
     LONG 0.0
     FI
            END;
       pair[1, ]:=(0,0);
         pair[2, ]:=(1,0);
         pair[3, ]:=(0,1);
         pair[4, ]:=(1,1);
         pair[5, ]:=(2,0);
         pair[6, ]:=(0,2);
         pair[7, ]:=(2,1);
         pair[8, ]:=(1,2);
         pair[9, ]:=(2,2);
         pair[10, ]:=(3,0);
         pair[11, ]:=(0,3);
         pair[12, ]:=(3,1);
         pair[13, ]:=(1,6);
         pair[14, ]:=(1,7);
         pair[15, ]:=(1,5);
         pair[16, ]:=(3,2);
           pair[17, ]:=(3,3);
           pair[18, ]:=(2,3);
           pair[19, ]:=(5,1);
   FOR i1 TO nss DO bb[i1]:=ce(pair[i1,1],pair[i1,2]) OD;
 FOR i1 FROM nss+1 TO n+m DO bb[i1]:=LONG 0.0 OD;
   ARPROC f;
     f:=( REF [ ] LONG REAL xd,INT i1) LONG REAL :
                 BEGIN
                    IF i1<=nsw THEN
                      LONG REAL s;
                   s:=-bb[i1+m];
                    IF i1=1 THEN
                      [1:m,1:m] LONG  REAL ar;
                       FOR i2 TO m DO
                          FOR i3 TO m DO
                            ar[i2,i3]:=ef(xd,i2,i3)
                          OD
                       OD ;
                     b:=bb;
                      lgaussp(ar,b[1:m],ae)
```

```
                         FI ;
                         FOR i2 TO m DO
                           s PLUSAB ae[i2]*ef(xd,i1+m,i2)
                         OD ;
                        s
CO
ELIF i1=2 THEN xd[2]-xd[4]
ELIF i1=3 THEN xd[1]-xd[2]
  ELIF i1=4 THEN xd[3]+xd[4]
  ELIF i1=5 THEN xd[5]+xd[6]
ELIF i1=6 THEN xd[7]-xd[8]
  ELIF i1=7 THEN xd[1]-xd[5]
  ELIF i1=8 THEN xd[3]-xd[7]
  ELIF i1=9 THEN xd[2]+xd[6]
  ELSE
xd[4]+xd[8]
CO
  ELSE
   LONG 0.0
 FI
       END;
       y1:=LONG 0.0;
        x1:=LONG 0.78;
ex:=(-x1,-x1,y1,-x1,x1,-x1,-x1,x1,y1,x1,x1,x1,-x1,y1,y1,y1,x1,y1);
   lndnewt(f,ex,LONG 1.0e-15,LONG 0.001);
    FOR i1 TO n DO print((newline,ex[i1],outline)) OD
 END
 FINISH
```