STUDIES IN NUMERICAL QUADRATURE

BY

AYSE ALAYLIOGLU, M.Sc.

A DOCTORAL THESIS

Submitted in partial fulfilment of the requirements
for the award of

the degree of Ph.D. of the Loughborough University of Technology
October 1974.

Supervisors: J. HYSLOP, Ph.D. G.A. EVANS, D.Phil.
Department of Mathematics.

Abstract

Various types of quadrature formulae for oscillatory integrals are studied with a view to improving the accuracy of existing techniques. Concentration is directed towards the production of practical algorithms which facilitate the efficient evaluation of integrals of this type arising in applications.

The Newton-Cotes and the Hermite-type quadrature formulae are considered and extended to the case of oscillatory weight functions. Algorithms for automatic generation of formulae of any order are presented. The reasons for the recommended preference for the low order formulae are pointed out. A more powerful and efficient technique based on the use of Chebyshev series is given which represents an extention of the well-known Clenshaw and Curtis procedure to oscillatory integrals. Applications and numerical results are included and comparisons with the related method of Piessens and Poleunis (1971) are made.

A practical method of evaluating oscillatory integrals over semi-infinite ranges is presented. It is based on integration between the zeros of the oscillatory weight functions and the subsequent use of the convergence acceleration technique of Shanks (1955), and is also appropriate for the evaluation of integrals that converge in the mean only.

A general investigation into the structure of Gaussian quadrature formulae is also considered. The notorious instability associated with the algebraic approach of generating quadrature coefficients for certain weight functions is discussed. A non-linear approach that delays the advent of instability is introduced. Further, an accurate and reliable algorithm, based on multilength rational arithmetic, is developed for

the specific weight functions whose associated monomials are expressible as rationals (apart from a suitable multiplying constant). Comparisons with the published tables (Stroud and Secrest (1965)) are also carried out. Algol 68 programs are presented.

Acknowledgements

The author is deeply indebted to her supervisors Dr. J. Hyslop and Dr. G.A. Evans for their generous advice and help in preparing this thesis.

Grateful thanks are given to the Head of the Department, Professor C. Storey, for his reading of the manuscript.

It is also a pleasure to record appreciation to the Scientific and Technical Research Council of Turkey for providing the scholarship.

Thanks are due to Mrs. B. Wright for her skill in typing the manuscript.

Table of Contents

CHAPTER 1

INTRODUCTION AND BACKGROUND

The purpose of the work is discussed with reference to the fields of application, and the content of the thesis is described.

1.1 Introduction

Numerical quadrature forms an important branch of numerical analysis with applications ranging into physics, engineering and applied mathematics. Considerable wealth of methods exists in the literature going back to early work by such authors as Gauss and Simpson (Davis and Rabinowitz (1967)). The methods rely on the analytic formulation of the Riemann integral in which the integrals are converted into summations. In practice this process involves "fitting" the integrand to a suitable and easily integrated function (such as a polynomial) from which a quadrature formula will follow.

The question arises as to whether the whole integrand need be fitted, or whether some part can be left and so be integrated out analytically in the production of the quadrature formula. For instance, an integrand with a trigonometric or exponential factor might be treated in this manner. It is clear that in the former case a highly oscillatory integrand will not be easily approximated by a polynomial over a wide interval, but if the oscillatory part can be treated separately the remaining function may be "smooth" enough for accurate fitting.

Hence the problem reduces to calculating α_i and x_i in the formula

$$\int_a^b F(x)\,dx = \int_a^b W(x)\,f(x)\,dx \simeq \sum_{i=1}^{n} \alpha_i\, f(x_i) \qquad (1)$$

where $F(x)$ represents the complete integrand and $W(x)\,f(x)$ the "factored form", $W(x)$ being a suitable weight function. Of particular interest are integrals with $W(x)$ given by a trigonometric function such as $\sin px$ or $\cos px$ with possibly large p. This type of integral occurs in virtually every branch of applied mathematics and is often generated

by the use of Fourier transforms. Two particular instances of this are discussed below, one from quantum mechanics and the other from fluid mechanics.

1.2 Applications of Oscillatory Integrals in Quantum Mechanics

Integrals having oscillatory integrands may occur in calculations of the energy levels of atomic and molecular systems by means of variational procedures involving Green's functions. The integrals occuring in the variational functionals are capable of simplification in that the kernels may be partially separated when Fourier transform representations of the Green's functions are employed. This method has the advantage of removing the singularities which cause most of the problems in the evaluation of atomic and molecular integrals but introduces the oscillatory integrands as a compensating disadvantage. Consequently an investigation of methods of evaluation of such integrals is of considerable importance in molecular quantum mechanics, as these new variational procedures are being developed.

Thus, the Schrödinger wave equation for the energy E of a system may be written in the form

$$(T + V)\psi = E\psi \tag{1}$$

where T is the kinetic energy differential operator, V is the potential energy of the system and ψ is the eigenfunction corresponding to the eigenvalue E. For example, in the case of an n-electron system the kinetic energy of the electrons may be represented by

$$T = -\frac{1}{2}(\nabla_1^2 + \nabla_2^2 + \ldots + \nabla_n^2) \tag{2}$$

and the potential energy by

$$V = V_{nn} + V_{ne} + V_{ee} \tag{3}$$

where V_{nn}, V_{ne} and V_{ee} represent the nuclear, nuclear-electron and electronic-electrostatic interactions.

Until recently one of the main methods of solving equation (1) was the Rayleigh-Ritz variational procedure (Eyring et al. (1944)). This method suffers from a number of defects and recently a new principle was proposed by Hall (1967), which introduces a Green's function and re-writes (1) as an integral equation. Thus, (1) is expressed as

$$\psi = \mu(E - T)^{-1} V\psi \tag{4}$$

introducing an eigenvalue parameter μ, whose exact physical value is unity. If $G(\underline{r},\underline{r}')$ is the Green's function corresponding to the operator $(E - T)^{-1}$ then the relation

$$\psi(\underline{r}) = \mu \int \psi(\underline{r}') \, V(\underline{r}') \, G(\underline{r},\underline{r}') \, d\underline{r}' \tag{5}$$

leads to consideration of the variational principle based on the functional

$$[\eta] = \frac{\int\int \omega^*(\underline{r}) \, V(\underline{r}) \, G(\underline{r},\underline{r}') \, V(\underline{r}') \, \omega(\underline{r}') \, d\underline{r} \, d\underline{r}'}{\int \omega^*(\underline{r}) \, V(\underline{r}) \, \omega(\underline{r}) \, d\underline{r}} \tag{6}$$

for arbitrary trial functions $\omega(\underline{r})$. This functional has been studied by various authors (Hall (1967), Hall et al. (1969), Hall et al. (1970), Robinson and Epstein (1970) and Robinson et al. (1970)) and results indicate that accurate energy values may be obtained with very simple trial functions in a number of important cases.

The main difficulty associated with the functional η is the evaluation of the integral in the numerator. For example, in the case of a one-electron Green's function

$$G(\underline{r},\underline{r}') = - \frac{1}{2\pi r_{12}} \exp(-kr_{12}) \tag{7}$$

where

$$E = -\frac{1}{2}k^2 \tag{8}$$

and

$$r_{12} = |\underline{r} - \underline{r}'| \tag{9}$$

the double volume integral may be expressed as

$$I = \iint \omega^*(\underline{r})\, V(\underline{r}) \left[-\exp(-kr_{12})/2\pi r_{12}\right] V(\underline{r}')\, \omega(\underline{r}')\, d\underline{r}\, d\underline{r}' \tag{10}$$

This type of integral is notoriously difficult to evaluate, one of the principal reasons being the presence of the r_{12}^{-1} singularity (Harris and Michels (1967)). The use of the Fourier representation of the Green's function (Hall et al. (1970))

$$G(\underline{r},\underline{r}') = -\frac{1}{4\pi^3} \int \frac{e^{-i\,\underline{s}.(\underline{r}-\underline{r}')}}{s^2 + k^2}\, d\underline{s} \tag{11}$$

enables partial separation of the kernel and consequently (10) may be expressed as

$$I = -\frac{1}{4\pi^3} \int |F(\underline{s})|^2 \frac{d\underline{s}}{s^2+k^2} \tag{12}$$

where

$$F(\underline{s}) = \int \omega^*(r)\, V(\underline{r})\, e^{-i\,\underline{s}.\underline{r}}\, d\underline{r} \tag{13}$$

represents the Fourier transform of the function $\omega^*(\underline{r})\, V(\underline{r})$. It is noted that the r_{12} singularity has been removed. This transformation is equivalent to working in momentum space and similar types of integrals occur in many other applications (for example, Mott and Sneddon (1948), Sacks (1953), Coulson and McWeeney (1949), McWeeney (1949)) in quantum mechanics.

In an application to the hydrogen molecular ion H_2^+ (Hall el al. (1970)), the above relation formed the basis of the evaluation of the integrals.

The introduction of elliptical two-centre co-ordinates reduced the integral (12) to the form

$$I = -16R^2 \int_0^\infty \int_0^1 (s^2 + k^2)^{-1} \left| \int_u^v f^*(\lambda) \sin \frac{Rs\xi}{2} \, d\xi \right|^2 ds \, du \qquad (14)$$

in which R is the internuclear distance, $f(\lambda)$ is the wave-function adopted and is defined only for $1 \leqslant \lambda \leqslant L$ and the variables are related by the following equations

$$\lambda = (\xi^2 + 1 - u^2)^{\frac{1}{2}}$$

$$v = (L^2 - 1 + u^2)^{\frac{1}{2}}$$

In the previous work (Hall et al. (1970)) the oscillatory nature of the integrands was avoided by changing the order of integration and proceeding analytically by means of contour integration, thus reducing (14) to the equivalent form

$$I = -\frac{8\pi R^2}{k} \int_0^1 \int_u^v \int_u^\eta f(\lambda) \, f(\lambda_1) \left[e^{-Rk(\eta-\xi)/2} - e^{-Rk(\eta+k)/2} \right] d\xi \, d\eta \, du \qquad (16)$$

where

$$\lambda_1 = (\eta^2 + 1 - u^2)^{\frac{1}{2}} \qquad (17)$$

The integration was then accomplished numerically by means of a product procedure of Gaussian type (Davis and Rabinowitz (1967)).

However, in generalizations to more complicated systems, such analytical procedures may not be feasible and it is therefore essential to be able to deal with integrals such as (12), (13) or (14) directly. Consideration has been given recently to this problem by Blakemore, Evans and Hyslop (1974), where improved versions of the earlier work of Hall et al. (1970) and Hyslop (1973) are presented. Consequently, accurate

and efficient methods for the numerical evaluation of Fourier transforms and also integrals with non-explicitly periodic integrands require to be developed.

In addition, similar variational techniques involving the variation-iteration method (Morse and Feshbach (1953)) are presently being applied to problems involving the Hartree-Fock Self-Consistent Field method (Eyring et al. (1944)) and preliminary calculations have been carried out on the ground state energies of the hydrogen and helium atoms (Hyslop (1972)). Such techniques require the recursive use of quadrature routines including those for oscillatory integrals similar to the above. Once again the development of efficient routines will be absolutely essential in reducing computer time for the large scale iterative investigations implied by the Self-Consistent Field method.

1.3 Applications of Oscillatory Integrals in Fluid Mechanics

Another source of oscillatory integrands occurs in slow viscous flow problems in fluid mechanics. The general Navier-Stokes' equations simplify for slow viscous flows, the important parameter being the Reynolds' number R which is small in these cases. The Reynolds' number arises as follows:

If $\hat{q}$ is the velocity vector at any point in the fluid, $\hat{\rho}$ its density, $\hat{p}_r$ its pressure and $\hat{\nu}$ the kinematic viscosity, then the steady Navier-Stokes equations are given by

$$(\underline{\hat{q}}.\underline{\hat{\nabla}})\ \underline{\hat{q}} = -\frac{1}{\hat{\rho}}\underline{\hat{\nabla}}\,\hat{p}_r + \hat{\nu}\,\underline{\hat{\nabla}}^2\,\underline{\hat{q}} \qquad (1)$$

$$\underline{\hat{\nabla}}.\underline{\hat{q}} = 0 \qquad (2)$$

where $\hat{\nabla}$ is the usual operator with respect to variables $\hat{x}$, $\hat{y}$, $\hat{z}$.

The equations are non-dimensionalized by using the transformations

$$\hat{q} = \hat{U} q \tag{3}$$

$$\hat{x} = \hat{\ell} x \tag{4}$$

$$\hat{y} = \hat{\ell} y \tag{5}$$

$$\hat{z} = \hat{\ell} z \tag{6}$$

$$\hat{p}_r = \hat{\nu} \hat{\rho} \hat{U} p_r / \hat{\ell} \tag{7}$$

yielding

$$\frac{\hat{U}\hat{\ell}}{\hat{\nu}} (q.\nabla)q = - \nabla p_r + \nabla^2 q \tag{8}$$

$$\nabla.q = 0 \tag{9}$$

where $\hat{U}$ and $\hat{\ell}$ are typical physical quantities in the problem under consideration. The non-dimensional constant $\hat{U}\hat{\ell}/\hat{\nu}$ is called the Reynolds' number and symbolized by R.

For very small R the term on the left hand side of equation (8) which contains the non-linearity may be expected to become unimportant and the solution might be close to that of the right hand side alone. The equations formed by the right hand side are called Stokes' equations and are soluble by separation of the variables in many cases.

However, if an attempt is made to set up an asymptotic series for the solution of the form

$$q = q_o + R q_1 + R^2 q_2 + \ldots \tag{10}$$

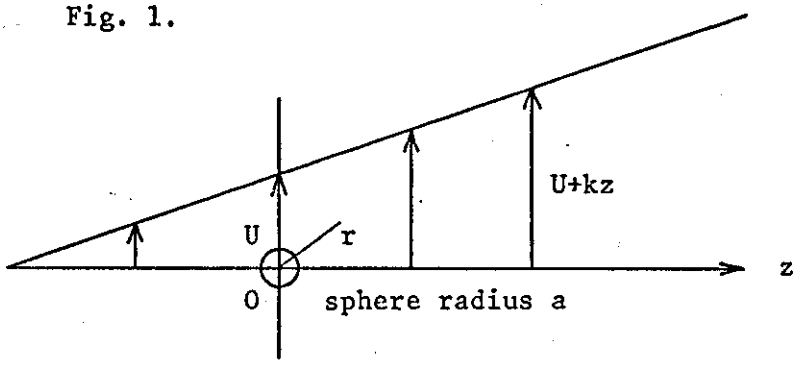
the solutions obtained from the differential equations satisfied by each term of the series are valid only in a limited region. Just what happens away from the region of validity can be discovered by using a different scaling and applying the theory of matched asymptotic expansions

(Van Dyke (1964)). Such expansions give a measure of the influence of the non-linear part even though the equations being solved are themselves linear.

A general way of tackling these problems is to apply a complex Fourier integral transform in three variables to the differential equations for a given term of the asymptotic series (Evans (1969), Evans and Ockendon (1972)). The inversion of this problem is in general a multiple integral of a highly oscillatory nature - often only convergent in the mean for some of the relevant parameter values.

Many practical problems have a further non-dimensional parameter in addition to the Reynolds' number and the interaction between the two parameters results in a long series of problems of the above type. One, which is of current interest, is an extension of the work of Saffman (1964) in which he considers the forces on a sphere in a viscous fluid of infinite extent across which is imposed a linear shear as illustrated in Figure 1.

Fig. 1.



With the notation in the diagram, the full problem satisfies the equations

$$(\underline{q}.\underline{\nabla})\,\underline{q} = -\underline{\nabla} p_r + \underline{\nabla}^2 \underline{q} \tag{11}$$

$$\underline{\nabla}.\underline{q} = 0 \tag{12}$$

with

$$\underline{q} = \underline{0} \quad \text{on} \quad r = \frac{Ua}{\nu} = \varepsilon \tag{13}$$

$$\underline{q} = (1 + \beta z)\underline{i} \quad \text{at} \quad \infty$$

which have been non-dimensionalized using ν/U as a typical length and U as a typical velocity. In the equations ε and β are two parameters and different problems arise depending on their relative sizes. For $\varepsilon \ll 1$ and $1 \ll \beta \ll \frac{1}{\varepsilon^2}$ the inner and outer expansions are those of Saffman. The requirement is to find the force on the sphere. This force follows directly from taking the limit of the outer solution near to the sphere (Evans (1969)). Consider the case where β is order of unity. Then the asymptotic series

$$\underline{q}_{\text{inertia}} = (1 + \beta z)\underline{i} + \varepsilon \underline{q}^{(1)} + \ldots \tag{14}$$

will result in problems for the individual terms which cannot satisfy all the boundary conditions. Hence, the inner problem is introduced and is presented by

$$r = \varepsilon \tilde{r} \tag{15}$$

$$\varepsilon(\underline{q}.\tilde{\underline{\nabla}})\underline{q} = -\tilde{\underline{\nabla}} p_r + \tilde{\nabla}^2 \underline{q} \tag{16}$$

$$\tilde{\underline{\nabla}}.\underline{q} = 0 \tag{17}$$

$$\underline{q} = \underline{0} \quad \text{on} \quad \tilde{r} = 1 \tag{18}$$

$$\underline{q} = (1 + \beta \varepsilon \tilde{z})\underline{i} \quad \text{at} \quad \infty \tag{19}$$

with the series solution of the form

$$\underline{q}_{inner} = \underline{q}_o + \varepsilon\, q_1 \qquad (20)$$

where $\underline{q}_o$ is the Stokes' solution for a uniform stream, that is

$$\underline{q}_o = \underline{i} + 0\left(\frac{1}{r}\right) \quad \text{at infinity.} \qquad (21)$$

Further $q^{(1)}$ must satisfy

$$(1 + \beta z)\frac{\partial \underline{q}^{(1)}}{\partial x} + \beta\, w^{(1)}\,\underline{i} = -\,\underline{\nabla}\, p_r^{(1)} + \underline{\nabla}^2\,\underline{q}^{(1)} \qquad (22)$$

$$\nabla.\underline{q}^{(1)} = 0 \qquad (23)$$

where $w^{(1)}$ is the z-component of $q^{(1)}$, with the limit as $r \to 0$ of $\underline{q}^{(1)}$ being a Stokeslet. This matching can be achieved automatically using a delta function, (Ockendon and Evans) to give the equations

$$(1 + \beta z)\frac{\partial \underline{q}^{(1)}}{\partial x} + \beta\, w^{(1)}\,\underline{i} = -\,\underline{\nabla}\, p_r^{(1)} + \underline{\nabla}^2\,\underline{q}^{(1)} + 6\pi\, i\, \delta(x)\,\delta(y)\,\delta(z) \qquad (24)$$

$$\underline{\nabla}.\underline{q}^{(1)} = 0 \qquad (25)$$

The complex Fourier transform of this equation is

$$-k^2\,\bar{\underline{q}} + i\,k\,\bar{p}_r = \beta\,\bar{w}\,\underline{i} - i\,k_1\,\bar{\underline{q}} - \beta\,k_1\frac{\partial \bar{\underline{q}}}{\partial k_3} + 6\pi\,\underline{i} \qquad (26)$$

$$\underline{k}.\bar{\underline{q}} = 0 \qquad (27)$$

$$\therefore\ \underline{k}\frac{\partial \bar{\underline{q}}}{\partial k_3} + \bar{w} = 0 \qquad (28)$$

and hence the solution for the force in the z-direction is

$$\text{force} = \frac{3}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{o}^{\infty}\frac{k_1}{k_2}\exp\left[-t(k^2 - ik_1)\right]$$
$$\left\{(k_3 + k_1 t\beta)\exp\left[-t^2\left(k_1 k_3\beta + \frac{k_1^2\,\beta^2 t}{3}\right)\right] - k_3\right\} dt\, dk_1\, dk_2\, dk_3 \qquad (29)$$

This force is 6π times the limit of the fluid velocity at the centre of the sphere. The integral (29) can be reduced to a real one given by

$$\text{force} = \frac{3}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{o}^{\infty}\frac{k_1}{k^2}\exp(-tk^2)\cos kt_1\left\{(k_3 + k_1 t\beta)\right.$$

$$\left.\exp\left[-t^2(k_1k_3\beta + \frac{k_1^2\beta^2 t}{3})\right] - k_3\right\}dt\ dk_1\ dk_2\ dk_3 \qquad (30)$$

where the term $\cos tk_1$ causes the oscillatory difficulties in the numerical evaluation of the integral. This integral is typical of the sort which result in pursuing other cases of β and ϵ. It is clear that an efficient algorithm for evaluating oscillatory integrals is necessary so that this four dimensional integral can become tractable numerically. The integral (30) bears marked similarity to the two centre quantum mechanical Fourier transforms arising from equation (1.2.13).

A further problem, in the same field, arises in the study of the interaction of sets of small spheres and results in an integral, convergent in the mean. The basic problem, in this case, is to find the force on one sphere due to the presence of another when both have general velocities in the fluid stream, which has a uniform velocity U. By extending the work of Evans, the transformed equations are obtained in the form

$$-ik_1\,\bar{u}_{01}\,R + k^2\,\bar{u}_{01} = ik_1\,\bar{p}_{r_{01}} - 6\pi\left(1 + \lambda u_{r1}\,e^{i(k_1 \sin\alpha + k_2\cos\alpha)}\right) \qquad (31)$$

$$-ik_1\,\bar{u}_{01}\,R + k^2\,\bar{u}_{01} = ik_2\,\bar{p}_{r_{01}} - 6\pi\,\lambda\,u_{r2}\,e^{i(k_1 \sin\alpha + k_2\cos\alpha)} \qquad (32)$$

$$-ik_1 \bar{w}_{01} R + k^2 \bar{w}_{01} = ik_3 \bar{p}_{r_{01}} \tag{33}$$

$$k_1 \bar{u}_{01} + k_2 \bar{v}_{01} + k_3 \bar{w}_{01} = 0 \tag{34}$$

where the bars indicate transforms of fluid velocity components and pressure, $\bar{u}_{01}$, $\bar{v}_{01}$, $\bar{w}_{01}$ and p_r respectively, u_{r1} and u_{r2} are components of relative velocities of the two spheres, λ is the ratio of their radii and

$$R = U\ell/\nu$$

where ℓ is distance between spheres. The solution of these equations is

$$\bar{p}_{r_{01}} = 6\pi k_1 \left[1 + \lambda u_{r1} e^{i(k_1 \sin\alpha + k_2 \cos\alpha)}\right] + 6\pi \lambda u_{r2} e^{i(k_1 \sin\alpha + k_2 \cos\alpha)} k_2 \tag{35}$$

$$\bar{u}_{01} = \left\{-6\pi(k_2^2 + k_3^2) \left[1 + \lambda u_{r1} e^{i(k_1 \sin\alpha + k_2 \cos\alpha)}\right] + 6\pi \lambda u_{r2} k_1 k_2 e^{i(k_1 \sin\alpha + k_2 \cos\alpha)}\right\} / k^2(k^2 - ik_1 R) \tag{36}$$

$$\bar{v}_{01} = \left\{6\pi k_1 k_2 \left[1 + \lambda u_{r1} e^{i(k_1 \sin\alpha + k_2 \cos\alpha)}\right] - 6\pi \lambda u_{r2} (k_1^2 + k_3^2) e^{i(k_1 \sin\alpha + k_2 \cos\alpha)}\right\} / k^2(k^2 - ik_1 R) \tag{37}$$

where the forces required can again be obtained directly from the limits of the inverse transforms of these expressions as $x, y, z \to 0$. All,

except the second integral for v_{01} can be found analytically by using rotations in the complex plane and contour integration. The integral for the second part of v_{01} is

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{(k_1^2 + k_3^2)\, e^{i(k_1 \sin\alpha + k_2 \cos\alpha) - i(\underline{k}.\underline{r})}}{k^2(k^2 - i\, k_1\, R)}\, d\underline{k} \qquad (38)$$

and it can be reduced, by considerable manipulation, to

$$2\pi^2(\sin\alpha - 1 + \sin\alpha\cos\alpha)/R\cos^2\alpha$$

$$+ \pi^2 \exp(-R\sin\alpha/2)\int_0^{\infty} a \exp(-y\sin\alpha)\, c^{-1}\, y^{-1}$$

$$\left[2(c + a^2)J_0(a\cos\alpha) - 2a\, J_1(a\cos\alpha)/\cos\alpha\right]da \qquad (39)$$

where

$$y = (a^2 + R^2/4)^{\frac{1}{2}} \qquad (40)$$

$$c = Ry + R^2/2 \qquad (41)$$

for $\sin\alpha > 0$, and a similar form for $\sin\alpha < 0$. Except when α is close to zero this integral is perfectly well-behaved and can be integrated numerically with little trouble. However the integral converges only in the mean when $\alpha = 0$. A method is suggested in Chapter 5 to deal with this difficulty and also the non-trigonometric nature of the periodicity.

1.4 Outline of Present Work

In the present work, therefore, concentration has been largely on the development of practical techniques for the evaluation of integrals with oscillatory integrands, although some attention has been paid to more general weight functions. For instance, a suitable choice of weight function can often deal directly with a singular integrand, or an integrand with some singular derivatives in which convergence of a numerical quadrature may be slow.

The work is arranged as follows. Chapter 2 is concerned with the comparison and systematization of the classical quadrature methods, starting with a description of the work on the non-oscillatory case. The classical methods such as that of Filon (1928) are in some cases inadequate to deal accurately with the oscillatory integrals encountered in the applications and a method of improving the results of the low-order quadrature formulae is investigated.

This procedure is not generally successful and a systematic method of extending the order of the quadrature prescriptions is investigated in Chapter 3. An algorithm for the automatic generation of Filon-type formulae of any order is presented and numerical instabilities associated with the calculation of the higher order quadrature coefficients are discussed.

These instabilities stimulated the search for a more powerful and efficient quadrature technique based on the use of Chebyshev series and this method is presented in Chapter 4, and represents an extension of the well-known Clenshaw and Curtis (1960) procedure to oscillatory integrands. Critical comparisons are carried out with the Newton-Cotes based methods of Chapter 3 and also with the related earlier work of Piessens and Poleunis (1971), Bakhvalov and Vasil'eva (1968).

In Chapter 5 a practical method of evaluating oscillatory integrals over semi-infinite ranges is given. This method is based on the convergence acceleration procedure provided by the non-linear transformations of Shanks (1955) and it is shown that the method applies also to integrals which converge only in the mean.

In the remainder of the thesis, a rather more general class of integrals is studied. An effective method for the systematic generation of the Gaussian quadrature formulae is presented for integrals with weight functions whose associated monomials are expressible as rationals. Using rational arithmetic, accurate quadrature coefficients are obtained for integrals for which the traditional methods are notoriously unstable.

In the course of this work several papers based on the contents have already been prepared and published and are listed in the References (Alaylioglu, Evans and Hyslop (1973, 1974a, 1974b, in press)).

CHAPTER 2

PRELIMINARY DISCUSSION

The structure of certain basic quadrature formulae is investigated by considering their derivation using both interpolatory polynomials and Taylor expansions, particular attention being paid to the forms of the respective error terms. Initially, the familiar Newton-Cotes formulae for non-oscillatory integrals are treated and then generalizations to integrals with oscillatory components are considered. Numerical results are given to illustrate the accuracy of the two different approaches in this case and the possibility of improving the results by means of the extrapolation technique of Romberg (1955) is also considered.

2.1 Introduction

Quadrature formulae of the interpolatory type for general weight functions are usually produced by the integration of interpolation formulae (most commonly of Lagrangian type), the general formulation together with error terms being quoted, for instance, by Davis and Rabinowitz (1967). In the case of oscillatory integrands, an approach based on the interpolation property of the non-oscillatory factor for the calculation of integrals over finite intervals, has been used by Filon(1928), Luke(1954), Flinn(1960), Clendenin(1966) and Tuck(1967). Filon has fitted a second order interpolation polynomial to the middle and the end points of each double section of the sub-divided interval, whereas Luke has considered an n-th order polynomial and has given formulae primarily suitable for hand calculations using tabulated functions. Flinn has modified Filon's method by fitting a fifth order polynomial to the values of the function and of its first derivative at the Filon abscissae. Clendenin has derived integration formulae based on the linear interpolation property. An independent development leading to a similar result has been given by Tuck.

Taylor series may also be employed to produce alternative quadrature rules (for instance, Squire (1970)) but these obviously suffer from the defect that derivatives are involved in the final forms, whereas only function values appear in the interpolatory rules (although derivatives appear in the error terms for both classes of formulae). However, the use of the more familiar Taylor series, as opposed to an interpolatory polynomial whether of the Lagrange, Newton, Bessel or Stirling type (Hildebrand, 1956), has the great advantage of directness.

In the present chapter a preliminary investigation is carried out into the derivation and structure of both Taylor and interpolatory types of formulae.

Initially, attention is confined to the derivation of the familiar Newton-Cotes formulae in the case of non-oscillatory integrands. Particular attention is paid to the derivation of error terms using Taylor series expansions.

A similar investigation is then performed on the structure of the first and second order formulae developed by Clendenin and Filon for the oscillatory case, with a view to setting up a quadrature extrapolation scheme of the Romberg type.

2.2 Investigation of the Structure of the Newton-Cotes Formulae

To begin with, the derivation of Newton-Cotes quadrature rules is considered using both the usual Newton interpolating polynomial and, for comparison purposes, a corresponding Taylor polynomial.

a) The Use of Interpolating Polynomials

The standard method (Hildebrand (1956) or Scheid (1968)) of obtaining Newton-Cotes quadrature formulae for the evaluation of the integral

$$I_n = \int_{x_o}^{x_n} f(x)\, dx \qquad (1)$$

using $x_i = x_o + ih$ with $i = 1,2,\ldots n$, is to replace $f(x)$ by the n-th order Newton polynomial

$$P_n(x) = \sum_{i=0}^{n} (x-x_o)(x-x_1)\ldots(x-x_{i-1})\, \Delta^i f_o/(i!h^i)$$

$$= \sum_{i=0}^{n} \binom{t}{i} \Delta^i f_o \qquad (2)$$

where $f_i = f(x_i)$ and $x = x_o + th$, with the usual notation.

The result is the n-th order Newton-Cotes approximation

$$N_n = h \sum_{i=0}^{n} C_{n,i} \Delta^i f_o \tag{3}$$

where

$$C_{n,i} = \int_o^n \binom{t}{i} dt \tag{4}$$

The error term is conveniently obtained by successive applications of Rolle's Theorem to a function of the form

$$f(x) - P_n(x) - c\Pi_n(x) \tag{5}$$

in which

$$\Pi_n(x) = (x-x_o)(x-x_1).....(x-x_n). \tag{6}$$

The final expression may be written as

$$I_n = N_n + \varepsilon(N_n) \tag{7}$$

where the error $\varepsilon(N_n)$ in N_n is expressed as

$$\varepsilon(N_n) = h^{n+2} C_{n,n+1} f^{(n+1)}(\xi_n) \tag{8}$$

with $0 < \xi_n < x_n$, assuming the derivatives exist.

For example, the familiar trapezoidal rule

$$I_1 = \frac{1}{2} h(f_o + f_1) - \frac{1}{12} h^3 f^{(2)}(\xi_1) \tag{9}$$

follows immediately.

As is well known, the case of even n requires special treatment, in that it is easily seen that $C_{n,n+1}$ vanishes when n is even. The implication is that equation (8) is valid only for odd values of n and modification

is required in the even case. In fact, in equation (5), $c\Pi_n(x)$ is replaced by $c\Pi_{n+1}(x)$ and gives a non-vanishing error term of the form $h^{n+3} C_{n,n+2} f^{(n+2)}(\xi_n)$ for even n.

Explicitly, the error is given by

$$\varepsilon(N_{2n}) = h^{2n+3} C_{2n,2n+2} f^{(2n+2)}(\xi_{2n}), \tag{10}$$

an example of this modification being the much used Simpson's rule

$$I_2 = \frac{1}{3}h(f_o + 4f_1 + f_2) - \frac{1}{90} h^5 f^{(4)}(\xi_2) \tag{11}$$

b) The Use of Taylor Series

Instead of the Newton interpolating polynomial of the previous section, a Taylor polynomial is used and yields the formula

$$I_n = T_n + \varepsilon(T_n) \tag{12}$$

where

$$T_n = \sum_{i=0}^{n} \int_{x_o}^{x_n} \frac{(x-x_o)^i}{i!} f^{(i)}(x_o)\, dx \tag{13}$$

and

$$\varepsilon(T_n) = \int_{x_o}^{x_n} \frac{(x-x_o)^{n+1}}{(n+1)!} f^{(n)}(\xi_n)\, dx \tag{14}$$

These expressions may be reduced to the forms

$$T_n = h \sum_{i=0}^{n} B_{n,i} h^i f^{(i)}(x_o) \tag{15}$$

and

$$\varepsilon(T_n) = h^{n+2} B_{n,n+1} f^{(n+1)}(\xi_n) \tag{16}$$

with

$$B_{n,i} = \frac{n^{i+1}}{(i+1)!} \tag{17}$$

For instance, the first order result may be quoted as

$$I_1 = h \{f_o + \frac{1}{2} h f_o^{(1)}\} + \frac{1}{6} h^3 f^{(2)}(\xi_1) \tag{18}$$

and it will be noted that, by comparison with equation (9),

$$| \varepsilon (T_1) | = 2 | \varepsilon (N_1) | \tag{19}$$

showing that the magnitude of the error in the Taylor series result is twice that for the trapezoidal rule. A comparison of the error terms for general n is now considered.

c) Comparison of the Error Terms

A direct comparison of the error coefficients $C_{n,n+1}$ and $B_{n,n+1}$ is not meaningful in general, since, as mentioned above $C_{n,n+1}$ is effectively replaced by $C_{n,n+2}$ when n is even. Also, symmetry has not been taken into account in the derivation of the Taylor series results, since expansion has been carried out about the end-point x_o. For this reason, attention is confined to even order formulae which, in the case of the Newton-Cotes interpolatory results, involve the coefficient $C_{2n,2n+2}$ of equation (10). In addition, the Taylor expansion is symmetrized by expanding about the mid-point x_n of the interval $[x_o, x_{2n}]$.

The modified result is written as

$$T_{2n} = \sum_{i=0}^{2n} \int_{x_o}^{x_{2n}} \frac{(x-x_n)^i}{i!} f^{(i)}(x_n)\, dx \tag{20}$$

and, noting that all odd terms in the summation vanish on integration, it follows immediately that

$$T_{2n} = h \sum_{i=0}^{n} B_{2n,2i}\, h^{2i} f^{(2i)}(x_n) \tag{21}$$

where the coefficients are now defined as

$$B_{2n,2i} = 2n^{2i+1} / (2i+1)! \tag{22}$$

The error term is also readily obtainable in the form

$$\varepsilon(T_{2n}) = h^{2n+3} B_{2n,2n+2} f^{(2n+2)}(\xi_{2n}) \tag{23}$$

which may now be compared directly with the corresponding Newton-Cotes result (10).

A useful example of this formula is the case n = 1 which yields the "Taylor-Simpson" result

$$I_2 = h\{ 2f_1 + \frac{1}{3} h^2 f_1^{(2)} \} + \frac{1}{60} h^5 f^{(4)}(\xi_2) \tag{24}$$

in contrast with equation (11). It will be noted that the magnitude of the error term is once again larger than the corresponding Newton-Cotes result, the factor this time being 3/2, as opposed to 2 for the trapezoidal case.

The first few values of the coefficients are shown in the following table for comparison purposes.

Table 1. Comparison of error coefficients in Newton-Cotes and Taylor series quadrature formulae.

Order (2n)	Newton-Cotes $C_{2n,2n+2}$	Taylor Series $B_{2n,2n+2}$	Ratio $\lvert B_{2n,2n+2}/C_{2n,2n+2} \rvert$
2	-1/90	1/60	3/2
4	-8/945	16/315	6
6	-9/1400	243/2240	135/8

d) Generation of Newton-Cotes Formulae Using Taylor Series

Although the error terms in the Taylor series quadrature formulae are larger than the corresponding Newton-Cotes results the analysis required in their derivation is considerably simpler. Consequently the possibility of using the simple Taylor series approach to generate the Newton-Cotes results is now investigated. As an example, if the Taylor-Trapezoidal rule (18) is considered and the first derivative term $h\, f_o^{(1)}$ is effectively replaced by its finite difference approximation $(f_1 - f_o)$, according to the relation

$$h\, f_o^{(1)} = (f_1 - f_o) - \frac{1}{2} h^2\, f^{(2)}(\xi_1), \tag{25}$$

then the trapezoidal rule (9) is obtained immediately.

Again, when the Taylor-Simpson rule (24) has the term $h^2\, f_1^{(2)}$ replaced according to the relation quoted by Abramowitz and Stegun (1965)

$$h^2\, f_1^{(2)} = (f_2 - 2f_1 + f_o) - \frac{1}{12} h^4\, f^{(4)}(\xi_2), \tag{26}$$

Simpson's rule (11) follows at once.

Care is needed with the higher order formulae to ensure that finite difference forms of the correct order are employed for the derivatives. Thus, the case n = 2 yields

$$I_4 = 4h\, (f_2 + \frac{2}{3} h^2\, f_2^{(2)} + \frac{2}{15} h^4\, f_2^{(4)}) + \frac{16}{315} h^7\, f_2^{(6)}(\xi_4) \tag{27}$$

The fourth derivative term is replaced according to the natural relation quoted by Abramowitz and Stegun as

$$h^4\, f_2^{(4)} = (f_4 - 4f_3 + 6f_2 - 4f_1 + f_o) - \frac{1}{6} h^6\, f^{(6)}(\xi_4) \tag{28}$$

However, in the case of the second derivativate term, a three-point formula of a similar form to equation (26) is <u>not</u> applicable and must

be replaced by the five-point formula

$$h^2 f_2^{(2)} = (f_3 - 2f_2 + f_1) - \frac{1}{12}(f_4 - 4f_3 + 6f_2 - 4f_1 + f_o) + \frac{1}{90} h^6 f^{(6)}(\xi_4) \quad (29)$$

(Abramowitz and Stegun)

When equations (28) and (29) are substituted in (3) the result is

$$I_4 = \frac{4}{90} h (7f_o + 32f_1 + 12f_2 + 32f_3 + 7f_4) - \frac{8}{945} h^7 f^{(6)}(\xi_4) \quad (30)$$

which is the usual four-strip or Boole's rule (Abramowitz and Stegun).

The possibility exists of generalizing this method to integrals involving oscillatory weight functions as in the integral

$$\int_a^b f(x) \begin{matrix} \sin \\ \cos \end{matrix} px \, dx \quad (31)$$

The resulting formulae are generalizations of those quoted by Squire (1970) and it will be shown that they are entirely analogous to the interpolatory procedures established by Filon, Tuck and Clendenin.

2.3 The Generalized Quadrature Rule with Oscillatory Weight Functions

Thus, on sub-dividing the interval [a, b] into m sub-intervals each of width 2h where

$$h = (b-a)/2m \quad (1)$$

and using a linear Taylor expansion (n = 1) about the mid-point x_i of the i-th sub-interval, where

$$x_i = a + (2i-1)h \quad (2)$$

it is apparent that

$$\int_a^b f(x) \cos px \, dx = \sum_{i=1}^{m} \int_{-h}^{h} f(x_i+t) \cos p(x_i+t) \, dt \tag{3}$$

$$\simeq \sum_{i=1}^{m} f(x_i) \int_{-h}^{h} \cos p(x_i+t) \, dt$$

$$+ \sum_{i=1}^{m} f^{(1)}(x_i) \int_{-h}^{h} t \cos p(x_i+t) \, dt \tag{4}$$

$$= \frac{2 \sin ph}{p} \sum_{i=1}^{m} f(x_i) \cos px_i$$

$$- \frac{2}{p^2} (\sin ph - ph \cos ph) \sum_{i=1}^{m} f^{(1)}(x_i) \sin px_i \tag{5}$$

This quadrature rule has been quoted by Squire (1970) and the method of generalization by introducing higher order Taylor series is obvious. For example, if a quadratic expansion (n = 2) is utilized about the mid-point x_i, the result is

$$\int_a^b f(x) \cos px \, dx \simeq \frac{2 \sin ph}{p} \sum_{i=1}^{m} f(x_i) \cos px_i$$

$$- \frac{2}{p^2} (\sin ph - ph \cos ph) \sum_{i=1}^{m} f^{(1)}(x_i) \sin px_i$$

$$+ \frac{1}{p^3} (2 ph \cos ph + p^2h^2 \sin ph - 2 \sin ph) \sum_{i=1}^{m} f^{(2)}(x_i) \cos px_i \tag{6}$$

The finite difference approximations

$$f^{(1)}(x_i) \simeq \{f(x_i+h) - f(x_i)\}/h \tag{7}$$

$$f^{(2)}(x_i) \simeq \{f(x_i+h) - 2f(x_i) + f(x_i-h)\}/h^2 \tag{8}$$

yield the Newton-Cotes type quadrature formula

$$\begin{aligned}\int_a^b f(x)\cos px\,dx \simeq h\alpha\left[f(b)\sin pb - f(a)\sin pa\right] \\ + h\gamma\sum_{i=1}^{m} f(x_i)\cos px_i \\ + 2h\beta\left[\frac{1}{2}f(a)\cos pa + \frac{1}{2}f(b)\cos pb\right. \\ \left. + f(a+2h)\cos p(a+2h) + f(a+4h)\cos p(a+4h) + \ldots\right]\end{aligned} \tag{9}$$

which is, of course, the well-known formula of Filon. The parameters appearing are given by

$$\alpha = (\theta^2 + \theta\sin\theta\cos\theta - 2\sin^2\theta)/\theta^3, \tag{10}$$

$$\beta = \left[\theta(1 + \cos^2\theta) - 2\sin\theta\cos\theta\right]/\theta^3, \tag{11}$$

$$\gamma = 4(\sin\theta - \theta\cos\theta)/\theta^3, \tag{12}$$

with

$$\theta = ph \tag{13}$$

Clendenin's formula which is a first order quadrature rule of the Newton-Cotes type results from the linear approximation

$$\int_a^b f(x)\cos px\,dx \simeq \sum_{i=0}^{m-1}\int_o^h \left\{f(a+ih) + t\,f^{(1)}(a+ih)\right\}\cos p(a+ih+t)\,dt \tag{14}$$

together with the finite difference representation

$$f^{(1)}(a+ih) \simeq \left\{f(a+ih+h) - f(a+ih)\right\}/h \tag{15}$$

where the step-size is defined by the equation

$$h = (b-a)/m \tag{16}$$

It is expressed by the following relation which is analogous to equation (9)

$$\int_a^b f(x) \cos px \, dx \simeq f(a) \left\{ - \sin pa/p + z \sin p(a+h/2) \right\} + f(b) \left\{ \sin pb/p - z \sin p(b-h/2) \right\} + y \sum_{i=1}^{m-1} f(a+ih) \cos p(a+ih) \tag{17}$$

where

$$y = (4/hp^2) \sin^2(ph/2) \tag{18}$$

$$z = (2/hp^2) \sin (ph/2) \tag{19}$$

For reference purposes, the formulae derived by Taylor expansions of order n = 1 and n = 2 and depicted in equations (5) and (6) respectively are referred to as T_1 and T_2. The corresponding formulae of the Newton-Cotes type derived by Clendenin and Filon shown in equations (17) and (9) are referred to as N_1 and N_2.

The local truncation errors associated with these formulae are denoted by $\varepsilon(T_1)$, $\varepsilon(T_2)$, $\varepsilon(N_1)$ and $\varepsilon(N_2)$. Clearly, the higher order terms left out in the Taylor expansion of the integrand provide estimates of the errors in (5) and (6). Thus, quoting only the leading terms, the estimates of the errors are

$$\varepsilon(T_1) \simeq f^{(2)}(x_i) \left[\frac{h^2}{2p} \sin p(x_i+h) + \frac{h}{p^2} \cos p(x_i+h) - \frac{1}{p^3} \sin p(x_i+h) - \frac{h^2}{2p} \sin p(x_i-h) + \frac{h}{p^2} \cos p(x_i-h) + \frac{1}{p^3} \sin p(x_i-h) \right] \tag{20}$$

and

$$\varepsilon(T_2) \simeq f^{(3)}(x_i) \left[\frac{h^3}{6p} \sin p(x_i+h) + \frac{h^2}{2p^2} \cos p(x_i+h) - \frac{h}{p^3} \sin p(x_i+h) \right.$$

$$- \frac{1}{p^4} \cos p(x_i+h) + \frac{h^3}{6p} \sin p(x_i-h) - \frac{h^2}{2p^2} \cos p(x_i-h)$$

$$- \frac{h}{p^3} \sin p(x_i-h) + \frac{1}{p^4} \cos p(x_i-h)\Big] \qquad (21)$$

The error estimates $\varepsilon(N_1)$ and $\varepsilon(N_2)$ are based on the difference between the interpolatory formula and the corresponding exact formula obtained by integrating the Taylor's expansion of the integrand.

Manipulation of the coefficients of the formulae yields the required estimates,

$$\varepsilon(N_1) \simeq f^{(2)}(x_i) \left[\frac{h}{p^2} \cos p(x_i+h) - \frac{1}{p^3} \sin p(x_i+h) + \frac{1}{p^3} \sin px_i \right.$$

$$\left. - \frac{h}{2p^2} \cos p(x_i+h) + \frac{h}{2p^2} \cos px_i \right] \qquad (22)$$

and

$$\varepsilon(N_2) \simeq f^{(3)}(x_i) \left[\frac{h^2}{3p^2} \cos p(x_i+h) - \frac{h}{p^3} \sin p(x_i+h) - \frac{1}{p^4} \cos p(x_i+h) \right.$$

$$\left. - \frac{h^2}{3p^2} \cos p(x_i-h) - \frac{h}{p^3} \sin p(x_i-h) + \frac{1}{p^4} \cos p(x_i-h) \right] \qquad (23)$$

2.4 Numerical Results

The four quadrature rules were applied to the integral

$$\int_0^1 e^x \cos px \, dx \qquad (1)$$

for $p = 10^i$, $i = 0\ (1)\ 4$ using sub-division formulae with $m = 5,10,20,40$. The calculations were carried out in double precision arithmetic, and the relative errors are tabulated in Table 2. It is apparent from the

Table 2. Relative Errors in the Evaluation of $\int_0^1 e^x \cos px\, dx$

Rule	p \ m	5	10	20	40
T_1	1	16639574	4164977	1041561	260410
N_1		- 8336996	- 2083562	- 520848	- 130209
T_2		- 13543	- 852	- 53	- 3
N_2		9031	568	37	2
T_1	10	13917390	4000589	1031376	259775
N_1		- 8373977	- 2085898	- 520993	- 130218
T_2		625461	37215	2299	143
N_2		- 438930	- 25210	- 1537	- 95
T_1	10^2	55942187	10095786	1293879	196852
N_1		- 2594599	- 4505568	- 587402	- 133868
T_2		2188942	- 155962	- 21602	- 874
N_2		- 52441	- 57911	21518	619
T_1	10^3	4599306	10159454	1227177	- 1004217
N_1		- 1706786	- 1737642	- 1745312	- 1747227
T_2		1166842	316106	80029	19527
N_2		11037	2261	- 13	- 590
T_1	10^4	59383630	15906375	3062169	730171
N_1		135039	- 5232	- 14746	- 16750
T_2		- 4699039	- 1636898	29455	18372
N_2		14144	3548	897	218

The entries have a multiplying factor of 10^{-10}.

T_1, N_1, T_2, N_2 indicate the quadrature rules defined by equations (2.3.5), (2.3.17), (2.3.6) and (2.3.9) respectively.

table that for small p ($p \sim 1$) the magnitude of the error in the Clendenin formula is a half of that of T_1. Similarly it is noted that the error in the Filon formula is 2/3 that of T_2. This result is to be expected as the present formulae reduce to the Newton-Cotes formulae as p tends to zero. In fact it can be easily shown that the error terms quoted in equations (2.3.20)-(2.3.23) reduce as $p \to 0$ to the results given in section 2.2. On the other hand for large p it is clear from formulae (2.3.20) - (2.3.23) that the prescriptions N_1 and N_2 are to be preferred to T_1 and T_2 respectively, and that this is borne out by the errors shown in the table. Indeed it is apparent that the interpolatory formulae are to be preferred for all values of p, the second order Filon formulae being, of course, preferable to the first order Clendenin formula. The interpolatory formulae have the added practical advantage of not needing analytic forms of the derivatives of f(x). Extension of the interpolatory formula approach to higher orders is analytically extremely involved and, in addition, numerical instabilities arise in a manner analogous to the usual Newton-Cotes formulae. This is the well-known effect in which some of the higher order coefficients become negative. This problem is discussed further in the next chapter. Here, attention is confined to the low order formulae (n = 1,2) and an attempt to improve the results is made by employing the extrapolation technique of Romberg.

In the limit as $p \to 0$ the errors in the formulae of Filon and Clendenin tend to the classical Newton-Cotes errors. Hence it is reasonable to expect a Romberg extrapolation technique to be viable when p is small, say order 1. The Romberg process can be defined by the recurrence relation

$$S_{i+1}^{(\ell)} = (q_i S_i^{(\ell)} - S_i^{(\ell-1)})/(q_i-1) \qquad (2)$$

in which $S_o^{(\ell)}$ is the value of the required integral with step-size $h/2^{\ell}$, and q_o is 2^L where the error of the integration formula is of order h^L. The constant q_i is defined by $2^{2i}\ q_o$ and, a triangular array of the form

$$\begin{array}{cccc} S_o^{(0)} & & & \\ S_o^{(1)} & S_1^{(1)} & & \\ S_o^{(2)} & S_1^{(2)} & S_2^{(2)} & \\ S_o^{(3)} & S_1^{(3)} & S_2^{(3)} & S_3^{(3)} \end{array} \qquad (3)$$

is generated, in which the first column is calculated using the underlying integration formula. From Tables 3 and 4 it is clear that the natural choice of $q_o = 4$ and $q_1 = 16$ give the relevant Romberg schemes for the Clendenin and the Filon formulae respectively. However, for larger values of p the Romberg scheme gives no improvement. In Table 5 the reverse process of using the exact value of the integral to determine the value of q_i for convergence is illustrated. The previous results with q_i being 4 and 16 respectively appears for p = 1, 10 whereas for large p the Filon quadrature formula appears to behave in a way in which $q_1 = 4$ gives good results.

As can be seen from the error formula (2.3.23) no simple error form exists for intermediate p so explaining the random results in this range. However for large p one of the terms of (2.3.23) becomes dominant and an error of the form

$$-\frac{2h^2}{3p^2} \sin ph \sin px_i \qquad (4)$$

arises, which confirms the choice of $q_1 = 4$ above.

Table 3. Successive entries in the "Romberg" Table based on Clendenin's results.

p	S_o	S_1	S_2	S_3
1	1.3791734721421101			
	1.3783117335795986	1.3780244873920948		
	1.3780963876398794	1.3780246056599730	1.3780246135444982	
	1.3780425567036133	1.3780246130581913	1.3780246135514058	1.3780246135515155
10	-0.1790494129925933			
	-0.1789369195149376	-0.1788994216890524		
	-0.1789089234281732	-0.1788995913992517	-0.1788996027132650	
	-0.1789019324672310	-0.1788996021469169	-0.1788996028634279	-0.1788996028658115
10^2	-0.0136322158640257			
	-0.0136348202625880	-0.0136356883954421		
	-0.0136294803192005	-0.0136277003380713	-0.0136271678009133	
	-0.0136288622118381	-0.0136286561760506	-0.0136287198985826	-0.0136287445350535
10^3	0.0022486018085984			
	0.0022486087458047	0.0022486110582068		
	0.0022486104701918	0.0022486110449875	0.0022486110441062	
	0.0022486109006772	0.0022486110441723	0.0022486110441180	0.0022486110441182
10^4	-0.0000831093631055			
	-0.0000831105289031	-0.0000831109175023		
	-0.0000831106079747	-0.0000831106343319	-0.0000831106154539	
	-0.0000831106246301	-0.0000831106301819	-0.0000831106299052	-0.0000831106301346

[a] The accuracy of the approximation is indicated by exhibiting inaccurate figures underlined.

[b] "Romberg factor", $q_o = 4$ is used to generate S_1 column.

Table 4. Successive entries in the "Romberg" Table based on Filon's results.

p	S_1	S_2	S_3	S_4
1	1.3780233689966924			
	1.3780245352574705	1.3780246130081890		
	1.3780246086389876	1.3780246135310887	1.3780246135331393	
	1.3780246132329714	1.3780246135392370	1.3780246135392689	1.3780246135392704
10	-0.1789074553099737			
	-0.1789000538868263	-0.1788995604586165		
	-0.1788996303698235	-0.1788996021353566	-0.1788996022987948	
	-0.1788996045747147	-0.1788996028550408	-0.1788996028578631	-0.1788996028579996
10^2	-0.0136293944731869			
	-0.0136287586923291	-0.0136287163069386		
	-0.0136286504419882	-0.0136286432252988	-0.0136286429387041	
	-0.0136286789235519	-0.0136286808223228	-0.0136286809697621	-0.0136286809790493
10^3	0.0022482156046106			
	0.0022482175777242	0.0022482177092651		
	0.0022482180888616	0.0022482181229374	0.0022482181245597	
	0.0022482182185763	0.0022482182272239	0.0022482182276329	0.0022482182276581
10^4	-0.0000831103678694			
	-0.0000831104559307	-0.0000831104618015		
	-0.0000831104779658	-0.0000831104794348	-0.0000831104795040	
	-0.0000831104836067	-0.0000831104839828	-0.0000831104840006	-0.0000831104840017

[a] The accuracy of the approximation is indicated by exhibiting inaccurate figures underlined.

[b] "Romberg factor", $q_1 = 16$ is used to generate S_2 column.

Table 5. "Romberg factors" for Clendenin's and Filon's results.

Rule	p	Romberg factor		
N_1	1	4.0013	4.0003	4.0001
N_2		15.8967	15.9503	15.6123
N_1	10	4.0146	4.0037	4.0009
N_2		17.4108	16.4045	16.1919
N_1	10^2	0.5759	7.6703	4.4879
N_2		9.0556	-2.6913	4.3737
N_1	10^3	0.9822	0.9956	0.9989
N_2		4.8823	-175.0600	0.0219
N_1	10^4	-25.8097	0.3548	0.8804
N_2		3.9864	3.9567	4.1138

[a] "Romberg factors" are calculated using (2.4.2) to yield the exact value of the integral in the S_1 and S_2 columns of Tables 3 and 4 respectively.

[b] For each value of p, the first row implies the value of q_o to be used in Table 3, and the second row implies the q_1 value of Table 4.

It is thought that a useful algorithm could be developed if ph was chosen to keep sin ph sensibly constant. The error is then available in a Romberg form. In practice, however, this algorithm proved unsatisfactory, again because of the involved dependence on p of the error formula in general.

Hence, it was considered worthwhile to find systematic methods for extending the order of the quadrature rules for oscillatory integrals and this topic is investigated in the following Chapter.

CHAPTER 3

AUTOMATIC GENERATION OF QUADRATURE FORMULAE FOR OSCILLATORY INTEGRALS

An effective method of automatically generating higher order Filon-type formulae is presented, and is based on the technique of systematically producing product combinations arising from the Lagrangian interpolation formulae employed. The familiar Newton-Cotes quadrature formulae for the special non-oscillatory case are reproduced for checking purposes (Abramowitz and Stegun (1965)), and examples of Filon-type quadrature coefficients are quoted. An application of the method to the evaluation of an oscillatory integral is presented. Hermite-Filon type quadrature formulae are also considered and numerical results are given for the test integral. Algol 68 versions of both Filon and Filon-Hermite type quadrature methods are presented in the Appendix.

3.1 Introduction

In the preceding chapter it was pointed out that the formulae of Clendenin and Filon are not always adequate for the accurate evaluation of oscillatory integrals. However, generalization of Filon's method of derivation in order to produce higher order quadrature formulae (for example, Flinn (1960)) involves tedious analysis. The present chapter is concerned with obtaining a rapid method for the systematic generation of such higher order formulae. The derivations are based on the use of Lagrangian interpolation formulae with equally-spaced abscissae. The Newton-Cotes formulae arise as special cases and hence are treated initially and act as a check for the accuracy of the method. Numerical results are given for Filon-type formulae of orders $n \leqslant 10$ and the accuracy is discussed. A Hermite-Filon type quadrature procedure arising from the use of Hermite's interpolation formula which involves derivatives is also generated by the same technique.

3.2 Systematic Generation of the Newton-Cotes Formulae

To evaluate the integral

$$I = \int_a^b f(x)\, dx \tag{1}$$

an attempt is made to develop an algorithm for the rapid and systematic computation of the quadrature coefficients, associated with the Lagrangian polynomial $L_j(x)$ in the approximation of the integrand according to

$$f(x) = \sum_{j=0}^{n} L_j(x)\, f(x_j) + \varepsilon_n \tag{2}$$

Here ε_n is the associated error

$$\varepsilon_n = \Pi_{n+1}(x)\, f^{(n+1)}(\xi)/(n+1)! \qquad (a < \xi < b) \tag{3}$$

and

$$\Pi_{n+1}(x) = (x-x_0)(x-x_1)\ldots(x-x_n) \tag{4}$$

(Hildebrand (1956)). Integral (1) may be expressed as

$$I = \sum_{j=0}^{n} C_{n,j}\, f(x_j) + \varepsilon_n \tag{5}$$

where $\{x_j\}$ denotes the (n+1) distinct points in $[a,b]$. The numbers $C_{n,j}$ $(0 \leqslant j \leqslant n)$ are the Cote's coefficients, given by

$$C_{n,j} = \int_a^b L_j(x)\, dx = \int_a^b \frac{\Pi_n'(x)}{\Pi_n'(x_j)}\, dx \tag{6}$$

where the primes denote that the terms $(x-x_j)$ and (x_j-x_j) are omitted respectively from the products $\Pi_n'(x)$ and $\Pi_n'(x_j)$. Introducing the symbol $\sigma_{n-\ell}(X)$ to represent the sum of all possible different products of the n quantities $\{X\} = \{x_0, x_1, \ldots x_{j-1}, x_{j+1}, \ldots x_n\}$ taken $(n-\ell)$ at a time,

the coefficients (6) can be expressed in a form suitable for automatic computation, namely,

$$C_{n,j} = \frac{1}{\Pi_n'(x_j)} \int_a^b \sum_{\ell=0}^{n} (-1)^{n-\ell} x^{\ell} \sigma_{n-\ell}(X)\, dx \tag{7}$$

$$= \frac{1}{\Pi_n'(x_j)} \sum_{\ell=0}^{n} (-1)^{n-\ell} \frac{b^{\ell+1} - a^{\ell+1}}{\ell + 1} \sigma_{n-\ell}(X) \tag{8}$$

3.3 Extension to Oscillatory Integrals

The result of 3.2 can be extended to derive quadrature coefficients for the oscillatory integrals of the form

$$I_c = \int_a^b f(t) \cos pt\, dt = \sum_{i=0}^{m} \int_{a+ih}^{a+(i+1)h} f(t) \cos pt\, dt \tag{1}$$

The subinterval width h is given by

$$h = (b-a)/(m+1) \tag{2}$$

The transformation

$$t = a + ih + x \tag{3}$$

leads to

$$I_c = \sum_{i=0}^{m} \left\{ \cos p(a+ih) \int_o^h f(a+ih+x) \cos px\, dx \right.$$

$$\left. - \sin p(a+ih) \int_o^h f(a+ih+x) \sin px\, dx \right\} \tag{4}$$

Lagrangian type interpolation formulae applied to the integrals in (4) yield a result of the form

$$I_c \simeq \sum_{i=0}^{m} \left\{ \cos p(a+ih) \sum_{j=0}^{n} C_{n,j}^{\cos} f(a+ih+x_j) \right.$$

$$\left. - \sin p(a+ih) \sum_{j=0}^{n} C_{n,j}^{\sin} f(a+ih+x_j) \right\} \qquad (5)$$

The location of the abscissae, x_j, in (5) is not restricted and may be specified quite generally. For example, equally spaced points may be chosen (Clendenin, Filon, Flinn), or, alternatively, prescriptions of the Gaussian type may be employed (Bakhvalov and Vasil'eva (1969)). In this chapter attention is confined to equally spaced ordinates so that direct comparison with existing quadrature procedures of the Filon type is possible. Generalizations involving automatic generation of Chebyshev and Gaussian type formulae are discussed in Chapters 4 and 6.

Thus, assuming equidistant abscissae

$$x_j = jh/n \qquad (6)$$

the Lagrangian coefficients are expressed by

$$\int_o^h \frac{\Pi_n'(x)}{\Pi_n'(x_j)} w(x)\, dx \qquad (7)$$

where the weight function $w(x)$ is taken to be $\cos px$ for $C_{n,j}^{\cos}$ and $\sin px$ for $C_{n,j}^{\sin}$.

It is noticed that in (5) the terms $\cos p(a+ih)$ and $\sin p(a+ih)$ appear outside the summations. Consequently the coefficients $C_{n,j}^{\cos}$ and $C_{n,j}^{\sin}$ may be calculated independently of i. This has the effect of reducing considerably the computation time.

The computation of the coefficients is facilitated by the following relations (Gradshteyn and Ryzhik (1963), p.183).

$$C_{n,j}^{\cos} = \frac{1}{\Pi_n'(x_j)} \int_o^h \sum_{\ell=0}^{n} (-1)^{n-\ell} \sigma_{n-\ell}(X)\, x^{\ell} \cos px\, dx$$

$$= \frac{1}{\Pi_n'(x_j)} \sum_{\ell=0}^{n} (-1)^{n-\ell} \sigma_{n-\ell}(X)\, \ell! \left[\sum_{k=0}^{\ell} \frac{x^{\ell-k}}{(\ell-k)!\, p^{k+1}} \sin\left(px + \frac{k\pi}{2}\right) \right]_o^h \qquad (8)$$

and

$$C_{n,j}^{\sin} = \frac{1}{\Pi_n'(x_j)} \int_o^h \sum_{\ell=0}^{n} (-1)^{n-\ell} \sigma_{n-\ell}(X)\, x^{\ell} \sin px\, dx$$

$$= -\frac{1}{\Pi_n'(x_j)} \sum_{\ell=0}^{n} (-1)^{n-\ell} \sigma_{n-\ell}(X)\, \ell! \left[\sum_{k=0}^{\ell} \frac{x^{\ell-k}}{(\ell-k)!\, p^{k+1}} \cos\left(px + \frac{k\pi}{2}\right) \right]_o^h \qquad (9)$$

where

$$\{X\} = \left\{x_0, x_1, \ldots x_{j-1}, x_{j+1}, \ldots x_n\right\} \qquad (10)$$

$$\Pi_n'(x_j) = (x_j - x_0)(x_j - x_1)\ldots(x_j - x_{j-1})(x_j - x_{j+1})\ldots(x_j - x_n) \qquad (11)$$

Similar analysis shows that the integral

$$I_s = \int_a^b f(t) \sin pt\, dt \qquad (12)$$

may be evaluated from the formula

$$I_s \simeq \sum_{i=0}^{m} \left\{ \sin p(a+ih) \sum_{j=0}^{n} C_{n,j}^{\cos} f(a+ih+x_j) \right.$$

$$\left. + \cos p(a+ih) \sum_{j=0}^{n} C_{n,j}^{\sin} f(a+ih+x_j) \right. \qquad (13)$$

In a similar way, a Hermite-Filon quadrature formula is derived using a Hermite interpolation formula (Hildebrand (1956)). The interpolation polynomial of degree (2n+1) now collocates with both f(x) and its derivative f'(x) at the points x_j, j = 0(1)n, as in

$$f(x) = \sum_{j=0}^{n} h_j(x)\, f(x_j) + \sum_{j=0}^{n} \bar{h}_j(x)\, f'(x_j) + \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\, \Pi_{2n+2}^2(x) \tag{14}$$

where

$$h_j(x) = \left[1 - 2\, L_j'(x_j)(x-x_j)\right] \left[L_j(x)\right]^2 \tag{15}$$

$$\bar{h}_j(x) = (x-x_j) \left[L_j(x)\right]^2 \tag{16}$$

$L_j(x)$ is the polynomial of degree n in the Lagrange formula defined by

$$L_j(x) = \Pi_n'(x)/\Pi_n'(x_j) \tag{17}$$

and ξ is in the range bounded by the extreme values of {X}.

The oscillatory integral may be expressed as

$$I_c = \int_a^b f(x)\, w(x)\, dx = \sum_{j=0}^{n} H_j\, f(x_j) + \sum_{j=0}^{n} \bar{H}_j\, f'(x_j) + E \tag{18}$$

where

$$H_j = \int_a^b h_j(x)\, w(x)\, dx \quad , \tag{19}$$

$$\bar{H}_j = \int_a^b \bar{h}_j(x)\, w(x)\, dx \tag{20}$$

and E is the error term in the approximate quadrature formula. From (15) and (19) we get

$$H_j = \frac{1}{\left[\Pi_n'(x_j)\right]^2}\left\{\left(1 + 2x_j \sum_{\substack{r=0\\ r\neq j}}^{n} \frac{1}{x_j - x_r}\right)\sum_{\ell=0}^{n}\sum_{i=0}^{n}(-1)^{2n-\ell-i}\sigma_{n-\ell}(X)\sigma_{n-i}(X)\int_a^b x^{\ell+i}\, w(x)\, dx\right.$$

$$\left. -2\sum_{\substack{r=0\\ r\neq j}}^{n}\frac{1}{x_j - x_r}\sum_{\ell=0}^{n}\sum_{i=0}^{n}(-1)^{2n-\ell-i}\sigma_{n-\ell}(X)\sigma_{n-i}(X)\int_a^b x^{\ell+i+1}\, w(x)\, dx\right\} \qquad (21)$$

Similarly, (16) and (20) give

$$\overline{H}_j = \frac{1}{\left[\Pi_n'(x_j)\right]^2}\left\{\sum_{\ell=0}^{n}\sum_{i=0}^{n}(-1)^{2n-\ell-i}\,\sigma_{n-\ell}(X)\,\sigma_{n-i}(X)\int_a^b x^{\ell+i+1}\, w(x)\, dx\right.$$

$$\left. - x_j\sum_{\ell=0}^{n}\sum_{i=0}^{n}(-1)^{2n-\ell-i}\,\sigma_{n-\ell}(X)\,\sigma_{n-i}(X)\int_a^b x^{\ell+i}\, w(x)\, dx\right\} \qquad (22)$$

The quadrature coefficients H_j and $\overline{H}_j$ have been written in a convenient form for programming. The integrals that appear in these equations are computed making use of the results

$$\int x^r \cos px\, dx = \sum_{\ell=0}^{r} \ell!\binom{r}{\ell}\frac{x^{r-\ell}}{p^{\ell+1}}\sin\left(px + \frac{1}{2}\ell\pi\right) + C \qquad (23)$$

$$\int x^r \sin px\, dx = -\sum_{\ell=0}^{r} \ell!\binom{r}{\ell}\frac{x^{r-\ell}}{p^{\ell+1}}\cos\left(px + \frac{1}{2}\ell\pi\right) + C \qquad (24)$$

(Gradshteyn and Ryzhik (1965)).

The subdivision formula for the integral I_c may be expressed in the form

$$I_c \simeq \sum_{i=0}^{m}\left\{\cos p(a+ih)\left[\sum_{j=0}^{n} H_{n,j}^{\cos} f(a+ih+x_j) + \sum_{j=0}^{n}\overline{H}_{n,j}^{\cos} f'(a+ih+x_j)\right]\right.$$

$$\left. - \sin p(a+ih)\left[\sum_{j=0}^{n} H_{n,j}^{\sin} f(a+ih+x_j) + \sum_{j=0}^{n}\overline{H}_{n,j}^{\sin} f'(a+ih+x_j)\right]\right\} \qquad (25)$$

where the coefficients are

$$H_{n,j}^{\cos} = \int_o^h h_{n,j}(x) \cos px \, dx \quad , \quad H_{n,j}^{\sin} = \int_o^h h_{n,j}(x) \sin px \, dx \tag{26}$$

and

$$\overline{H}_{n,j}^{\cos} = \int_o^h \overline{h}_{n,j}(x) \cos px \, dx \quad , \quad \overline{H}_{n,j}^{\sin} = \int_o^h \overline{h}_{n,j}(x) \sin px \, dx \tag{27}$$

with $h_{n,j}$ and $\overline{h}_{n,j}$ as defined by (15) and (16).

Similarly, the subdivision formula for I_s is expressed by

$$I_s = \sum_{i=0}^{m} \left\{ \sin p(a+ih) \left[\sum_{j=0}^{n} H_{n,j}^{\cos} f(a+ih+x_j) + \sum_{j=0}^{n} \overline{H}_{n,j}^{\cos} f'(a+ih+x_j) \right] \right.$$

$$\left. + \cos p(a+ih) \left[\sum_{j=0}^{n} H_{n,j}^{\sin} f(a+ih+x_j) + \sum_{j=0}^{n} \overline{H}_{n,j}^{\sin} f'(a+ih+x_j) \right] \right\} \tag{28}$$

3.4 Results and Discussion

To begin with, for checking purposes, the Newton-Cotes coefficients were calculated for $n \leqslant 10$ using formula (3.2.8) and comparison was made with the standard coefficients quoted, for example, by Abramowitz and Stegun (1965).

In Table 1, representative quadrature coefficients for oscillatory integrands are presented. These coefficients depend, of course, on the angular frequency p and the interval $[a,b]$, and are given, by way of illustration, for the special case $p = 10$ and the interval $[0,\pi/2]$. It is noted that the results need only be quoted for $j \leqslant [n/2]$ because of the symmetry relations

$$C_{n,j}^{\cos} = - C_{n,n-j}^{\cos} \tag{1}$$

$$C_{n,j}^{\sin} = C_{n,n-j}^{\sin} \tag{2}$$

which hold in this case. The further relations

$$\sum_{j=0}^{n} C_{n,j}^{\cos} = 0 \tag{3}$$

$$\sum_{j=0}^{n} C_{n,j}^{\sin} = 2/p \tag{4}$$

are also true here and were extensively employed for checking purposes. Instability appears in the higher order coefficients (n = 9,10) due to cancellation effects, though this could be eliminated by using higher precision arithmetic. However, it is well known that such high order formulae of the Newton-Cotes type exhibit instability in use and, in practice, are unlikely to be employed. The use of the coefficients appearing in the columns with n = 1,2 and 5 will give rise to the existing quadrature formulae of Clendenin, Filon and Flinn, respectively.

Table 1. Quadrature coefficients $C_{n,j}^{cos}$ and $C_{n,j}^{sin}$ for the case p = 10 and interval $[\,0,\ \pi/2\,]$.

n		j=0	j=1	j=2	j=3	j=4	j=5
1	c	0.127324					
	s	1.000000					
2	c	0.127324	0.000000				
	s	0.967577	0.064846				
3	c	0.399870	-0.817639				
	s	0.963524	0.036376				
4	c	0.450342	-0.646036	0.000000			
	s	0.759835	0.830968	-1.181606			
5	c	0.603496	-1.119071	0.976355			
	s	0.714746	0.754441	-0.469187			
6	c	0.643445	-0.974299	0.400236	0.000000		
	s	0.409050	2.299471	-4.171137	4.925232		
7	c	0.535871	0.116622	-2.840417	5.078313		
	s	0.326363	2.232637	-2.854676	1.295676		
8	c	0.509818	0.382441	-2.770556	2.863816	0.000000	
	s	0.220719	2.405823	-2.821639	1.583885	-0.777578	
9	c	0.431563	1.018550	-4.286255	5.403157	-4.646193	
	s	0.183226	2.087470	-1.333411	-0.685093	0.747835	
10	c	0.406737	1.065879	-3.382395	2.329711	-0.172780	-0.000084
	s	0.135033	2.003227	-1.262378	1.021923	-3.964286	6.133096
		$\times 10^{-1}$	$\times 10^{-1}$	$\times 10^{-1}$	$\times 10^{-1}$	$\times 10^{-1}$	$\times 10^{-1}$

(c denotes $C_{n,j}^{cos}$ and s denotes $C_{n,j}^{sin}$)

As an example of the use of the quadrature formulae (3.3.5) and (3.3.13) and also to provide a check on the accuracy of the computed coefficients, consideration is given to the evaluation of the integral

$$\int_0^1 e^x \cos px \, dx = \left[e (\cos p + p \sin p) - 1\right](p^2 + 1)^{-1} \qquad (5)$$

for $p = 10^i$, $i = 0(1)4$ and $n \leq 10$, by the method described.

In practice, it is customary to investigate empirically the effect of truncation and round-off errors by increasing either the number of subdivisions of the range of integration, or the order of the formulae used. To demonstrate these effects, the number of points at which the integrand is evaluated is chosen so that, within any block of Table 2, the number of function evaluations is approximately constant, (10, 20 and 30). In general, the formulae of order four, five and six give the best results for this example. It is clear that the accuracy falls off for the higher order formulae. This is mainly due to the increased oscillations of the weights of the quadrature formulae as the order n increases. Moreover, for small p the evaluation of the monomials introduces instability. This effect is more pronounced when n is large, as seen in the case of $p = 1$ using the formula of order 10, and 30 function evaluations. Also, for large p subsequent subtractions of multiples of 2π involved in the evaluation of trigonometric functions introduce substantial errors in the results. However, some improvement in the accuracy of the formula can be experienced if the trigonometric functions and the monomials (3.3.23) and (3.3.24) are evaluated using higher precision. These effects arise again in the algorithm presented in Chapter 4.

Table 2. Relative errors in the numerical evaluation of

$$\int_0^1 e^x \cos px \, dx$$ by Filon type rule

n	m+1	p=1	p=10	p=100	p=1000	p=10000
1	10	8337	8374	2595	1707	- 134
2	5	- 9	439	524	- 11	- 14
3	3	19	5	- 9	- 13	1
4	3	0	1	1	0	1
5	2	0	0	1	0	1
6	2	0	- 1	0	0	- 1
7	1	- 1	- 5	- 1	- 1	- 7
8	1	- 1	- 3	16	- 13	3
9	1	- 16	99	127	43	213
10	1	- 314	- 1311	- 310	- 804	- 1140
1	20	2084	2086	4506	1738	12
2	10	- 1	25	58	- 2	- 3
3	7	1	1	0	0	4
4	5	0	0	0	0	1
5	4	0	0	0	0	0
6	3	0	0	0	1	1
7	3	1	1	- 2	4	0
8	3	2	- 3	- 9	0	5
9	2	2	63	66	51	446
10	2	- 116	- 2295	- 1644	- 910	- 6099
1	30	926	926	1171	- 108	- 145
2	15	0	5	224	- 1	- 3
3	10	0	0	1	0	1
4	8	0	0	0	0	1
5	6	0	0	0	0	- 4
6	5	0	0	1	0	0
7	4	- 1	- 1	3	- 2	- 15
8	4	1	- 5	8	12	28
9	3	- 82	3	- 76	- 275	- 113
10	3	- 3791	22	- 1878	- 1470	- 565

The entries have a multiplying factor of 10^{-7}.

(n = order, (m+1) = number of subdivisions)

The calculations outlined in Table 2 were also performed using the Hermite formulae defined by (3.3.25), (3.3.26) and (3.3.27), as shown in Table 3. The quadrature results of order $n \geq 4$ are inferior to those obtained from the Lagrangian interpolation formulae (3.3.5) and (3.3.13) and serious instability occurs in the quadrature coefficients beyond order $n = 4$. Thus the use of the Hermite formulae is not recommended, except for low order, especially when it is recalled that the derivatives $f'(x_j)$ are also required.

Table 3. Relative errors in the numerical evaluation of

$$\int_0^1 e^x \cos px \, dx$$ by Hermite-Filon type rule.

n	m+1	p=1	p=10	p=100	p=1000	p=10000
1	10	-1	-1	1	0	1
2	5	0	0	0	0	1
3	3	0	0	0	0	1
4	3	-1	-6	-2	14	7
5	2	48	-190	93	-102	-114
6	2	113612	-10230	-4631	-2218	-20678
1	20	0	0	0	5	7
2	10	0	0	0	0	1
3	7	0	0	0	0	4
4	5	-143	2	4	-6	-14
5	4	59912	-194	-115	222	-1924
1	30	0	0	0	0	0
2	15	0	0	0	0	-2
3	10	-1	0	0	0	0
4	8	-233	3	-20	-17	-41

The entries have a multiplying factor of 10^{-7}.

(n = order, (m+1) = number of subdivisions).

Appendix

The heart of the computational method is the routine to evaluate $\sigma_{n-\ell}(X)$, the coefficient of $x^{\ell}(-1)^{n-\ell}$ in $(x-x_o)(x-x_1)\ldots(x-x_{j-1})(x-x_{j+1})\ldots(x-x_n)$ in equation (3.2.7).

There is no loss of generality if the coefficient of $x^{\ell}\ (-1)^{n-\ell}$ in $(x-x_1)(x-x_2)\ldots(x-x_n)$ can be found. This latter product expands to give:

$$x^n - (x_1+x_2+\ldots+x_n)\ x^{n-1} + (x_1x_2+x_1x_3+\ldots+x_2x_3+\ldots)\ x^{n-2}$$

$$- (x_1x_2x_3+\ldots)\ x^{n-2} + \ldots + x_1x_2x_3\ldots x_n$$

and it is clear that the required coefficient is the sum of the n quantities $\{X\} := \left\{ x_1, x_2, \ldots, x_n \right\}$ taken $(n-\ell)$ at a time.

With a computational language, such as Algol 68, available, which can efficiently compile recursive algorithms, the calculation of the above coefficients can be elegantly programmed. There are $(n-\ell)$ x's in each product, which suggests that to scan through all the combinations required in the final summation one of the $(n-\ell)$ x's could be chosen at each level of the recursion. The process may be represented diagrammatically as

Enumeration of elements in each product

Stages of process	1	2.....(n-ℓ) level of recursion		
1	x_1	x_2 . . .	$x_{n-\ell-1}$	$x_{n-\ell}$
2	x_1	x_2 . . .	$x_{n-\ell}$	$x_{n-\ell+1}$
⋮	⋮	⋮		
	x_1	x_2 . . .	$x_{n-\ell-1}$	x_n
	x_1	x_2 . . .	$x_{n-\ell}$	x_n
	⋮	⋮		
	x_1	x_3 . . .	x_{n-1}	x_n
	x_2	x_3 . . .	$x_{n-\ell}$	$x_{n-\ell+1}$
	⋮	⋮		
$\binom{n}{n-\ell}$	$x_{\ell+1}$	$x_{\ell+2}$ . . .	x_{n-1}	x_n

That is to say, the array of x's,X, would be global to the scanning routine but the indices would be fixed according to the local variables, I1, at each level of the recursion. A for loop at each level would then effect the search for the combinations, and only when the level of the recursion had reached (n-ℓ), would the complete product be added into the summation accumulator.

Hence the following Algol 68 procedure is produced.

```
proc sigma = (ref[ ]real X, int n,k) real:
begin
  int pd; [1:k] int itrans, id;
```

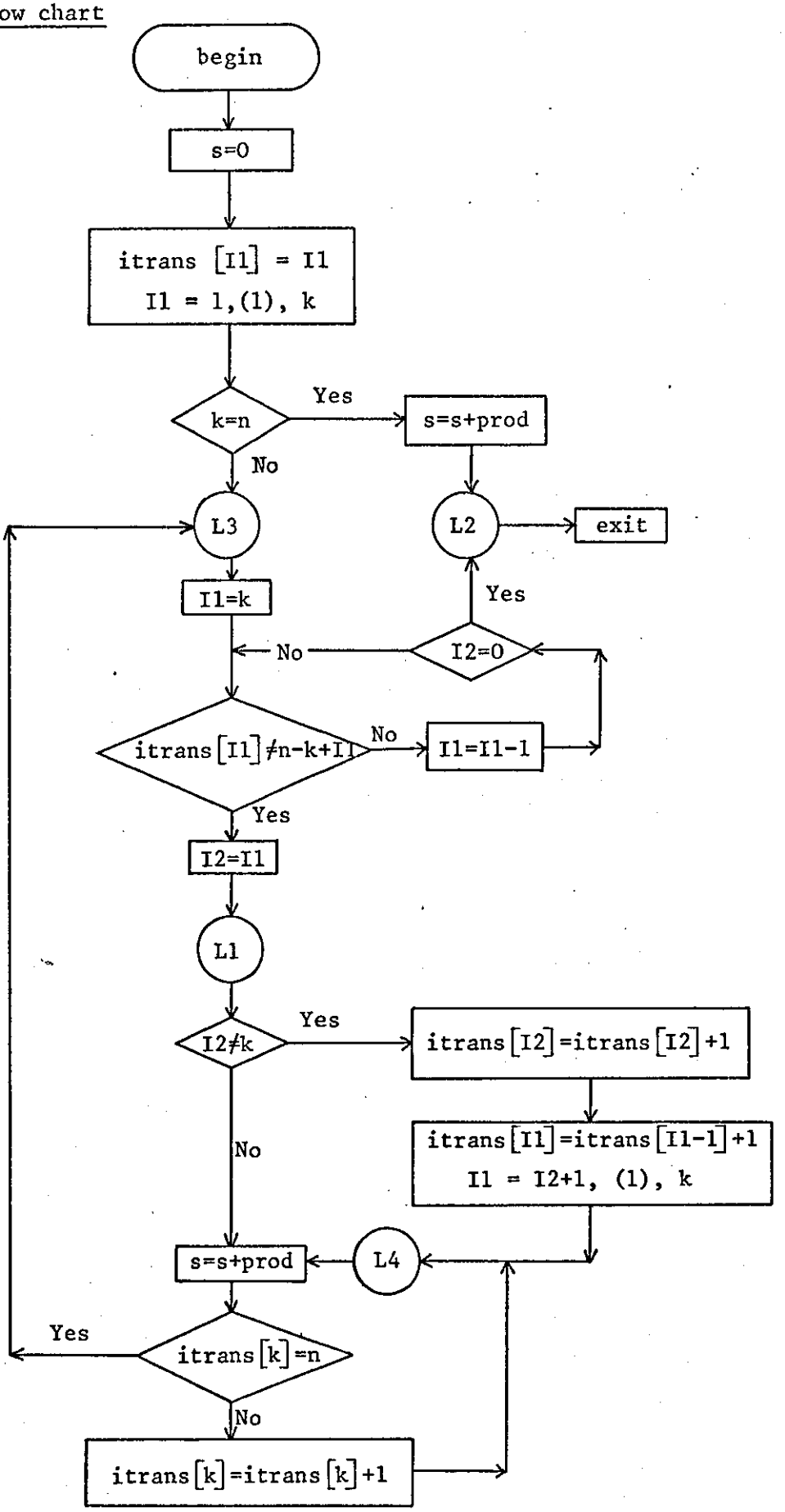
```
proc prod = real: (real pr;
  for I1 to k do (pr times X[itrans[I1]];pr);
proc sum = real: (real s←0.0; pd minus 1;
  for I1 from pd+1 to id[pd+1] do
  (itrans [pd+1]←I1; (pd≠0 | id[pd]←I1-1);
  s plus (pd=0 | prod | sum)); pd plus 1; s);
pd←k; (k≠0 | id[pd]←n); (k=0 | 1.0 | sum)
end;
```

The parameters of the procedure sigma are: X, the array of x's, n, the number of x's in the array X and k, the number of x's in each product, i.e. (n-ℓ). The procedure prod simply forms the product of the relevant x's whose k subscripts are stored in the array itrans. The elements of the array itrans are set in the procedure sum, one at each level of the recursion. The upper limit of the for loop, id[pd+1], is fixed in the previous level of the recursion and effects the correct search for the required combinations, the final product being added into the summation accumulator s. The level of the recursion is counted by pd. It starts at the value k and becomes zero when the indices of k x's are fixed. Up to this point the procedure sum calls itself and when pd is zero the procedure prod is called. The case k=0 is a trivial special case dealt with in the final line where the required real is delivered by sum.

For comparison purposes, the same algorithm was programmed in Algol 60. A test program, using sigma repeatedly, took 65 time units using Algol 68 and 120 time units using Algol 60.

Clearly the algorithm can be programmed in a non-recursive manner, although the resulting programs are less elegant. A version of this

The flow chart



approach was coded in both Algol 60 and Algol 68. In these programs the scanning order for the x combinations was the same as in the recursive program. The Algol 68 version took 75 time units against 80 time units for the Algol 60 version. It is thought that these time differences are explained in the main by the relatively poor procedure and parameter mechanism of Algol 60 compared with Algol 68. Small language refinements such as the operators plus and times also cut the time factor a little.

An equivalent non-recursive iterative program is described below. It involves an explicit count to determine the number of repetitions. The parameters are as follows: n denotes the number of x's which form the array X and k is the number of x's in every product. The products are calculated by the procedure prod, when the current subscripts of the x's in each combination are fixed. This choice of subscripts is made using the array itrans, containing k elements each of which sets a subscript for x in the current combination. Initially, the k elements of the array itrans [I1] are set to I1. In the do-loop labelled L3 the upper limits for each element itrans [I1] of itrans are checked starting at itrans [k], and I2 is set up to indicate the highest index still to reach its upper limit. At this point control moves to label L1 which updates the I2-th element, and the succeeding do-loop resets itrans[I1] for I1 > I2, i.e. the elements which have previously reached their maximum. If the element is still below n-k+I1 for itrans[I1] this subscript is further incremented until the range of possibilities is exhausted. The products of the combinations are added into the summation accumulator, s, and the final result is delivered at the label L2. The Algol 68 code for this routine is:

```
proc sigma = (ref[ ]real X, int n,k) real:
begin
  real s←0.0; [1:k] int itrans; int I2;
  proc prod = real: (real pr;
    for I1 to k do pr times X[itrans[I1]]; pr);
  for I1 to k do itrans[I1]←I1;
  if k=n then s plus prod; goto L2 fi;
L3:for I1 from k by -1 to 1 do
  if itrans[I1]≠ n-k+I1 then I2←I1; goto L1 fi;
  goto L2;
L1:if I2≠k then itrans[I2]←itrans[I2]+1;
  for I1 from I2+1 to k do itrans[I1]←itrans[I1-1]+1 fi;
L4:s plus prod;
  if itrans[k]=n then goto L3 else itrans[k] plus 1; goto L4 fi;
L2: s
end;
```

The recursive version of the procedure sigma is used in the following programs.

```
proc generalized filon = (proc(real)real f,real a,b,p,int n1,bool type)
                                                    [ ]real:
begin
c This is a procedure to evaluate ∫_a^b f(x) cos/sin px dx for n=1,2,....,n1
order quadrature formula of Filon-type with step-size h, and the boolean
type is true if cos px is the weight function and false if sin px is
the weight function. c
  int m,n; real partc,parts,h,h1,ax,fax,ah,cs1,cs2; [1:n1] real integ;
  [1;n1] ref[ ]real cnjc,cnjs,x;
  c cnjc and cnjs refer to the coefficients C^cos_{n,j} and C^sin_{n,j} defined
  as in § 3.3 c
  proc ge =(int j,n,bool sorc,ref[ ]ref[ ]real x)real:
  begin
  c Self-generation of C^cos_{n,j} or C^sin_{n,j} for the boolean sorc being true
  and false respectively c
    [1:n] real xx; real c←0.0; int k;
    proc sigma = (ref[ ]real x,int n,k)real: begin  . . .  end;
    proc pr=real:
      (real r←1.0;
      for i1 from 0 to n do ( i1=j | skip | r times x[n][j] - x[n][i1] ));
      c This procedure evaluates Π'_n(x_j) defined by (3.3.11). c
      r);
```

```
proc fun =(int r, real p,q) real:
begin
c This is a procedure to evaluate ∫ x^r w(x) dx at x=q, using
equations (3.3.23) and (3.3.24) with sorc being true for
w(x)=cos px, and false for w(x)=sin px. c
  int i,iz←0; real y←0.0,p1←1.0,p2←1.0; [0:r] real yy;
  proc msin=(real x) real: (real s← -sin(x); s);
  proc mcos=(real x) real: (real s← -cos(x); s);
  proc tsin=(int i) proc(real) real:(i | sin,cos,msin,mcos | skip);
  proc tcos=(int i) proc(real) real:(i | cos,msin,mcos,sin | skip);
  c Procedures tsin and tcos deliver one of the trigonometric
  procedures sine, cosine, -sine, or -cosine for the evaluation of
  {cos / sin} [px + 1/2 (i-1)π]}  i = 1,2,...r in equations (3.3.23) and
  (3.3.24). c
 (r=0 | y←(sorc | sin(p*q)/p | -cos(p*q)/p); goto ℓ1);
  c The series is summed in the reverse order. c
  for im to r do (p2 times p; p1 times im); p2 times p;
  yy[r]←p1/p2;
  for i2 from r by -1 to 0 do
  begin
    (i2≠r | yy[i2]←yy[i2+1]*(p*q)/(r-i2));
    (i2<4 | i←i2+1 | iz←(i2+1)-((i2+1)÷4)*4; i←(iz=0 | 4 | iz));
    (sorc | y plus yy[i2]*tsin(i)(p*q) | y minus yy[i2]*tcos(i)(p*q))
  end;
ℓ1 :y
 end; c end of procedure fun c
```

```
  for i to n do
  begin
  c The array xx, is formed by leaving out the x_j'th term, as in
  equation (3.3.10), and enumerating the terms from 1 to n. c
    if i=j then if j ⩽ n then xx[i]←x[n][i-1]fi
          else if i < j then xx[i]←x[n][i-1] else xx[i]←x[n][i] fi
    fi
  end;
  for ℓ from 0 to n-1 do
  begin
    k←n-ℓ;
    c plus (fun(ℓ,p,h))-fun(ℓ,p,0.0)*sigma(xx,n,k)*(k=(k÷2)*2|1.0|-1.0)
  end;
  c plus (fun(n,p,h)-fun(n,p,0.0));
  c div pr;
  c
end; c end of procedure ge c
for n to n1 do
c This do loop generates n=1,2,....n1 order formula. c
begin
  integ[n]← 0.0; c the integral c
  h1←(b-a)/ round (10/n); c initially estimated step-size c
  m←entier ((b-a)/h1+0.5);
  h←(b-a)/(m+1);
  c Hence h now divides (b-a) exactly m times. c
  cnjc[n]←loc[0:n] real; c C^cos_{n,j} c
  cnjs[n]←loc[0:n] real; c C^sin_{n,j} c
```

```
    x[n]←loc[0:n]real; c the abscissae c
    for j from 0 to n do
    begin
      x[n][j]←h*j/n
      c if equally spaced abscissae are used c
    end;
    for j from 0 to n do
    begin
      cnjc[n][j]←ge(j,n,true,x);
      cnjs[n][j]←ge(j,n,false,x)
    end;
    for i from 0 to m do
    begin
      partc←0.0; parts←0.0; ah←a+i*h; cs1←cos(p*ah); cs2←sin(p*ah);
      for j from 0 to n do
      begin
        ax←ah+x[n][j]; fax←f(ax);
        partc plus cnjc[n][j]*fax; parts plus cnjs[n][j]*fax;
      end;
      integ[n]plus if type then
                          c the weight function is cos px c
                          partc*cs1-parts*cs2
                          else
                          c the weight function is sin px c
                          partc*cs2+parts*cs1
                  fi
    end
  end; c end of n do loop c
  integ
end
```

```
proc hermite filon=(proc(real)real f,fd,real a,b,p,int nl,bool type)
                                                    []real:
begin
  c This is a procedure to evaluate  ∫_a^b f(x) cos/sin px dx using each of
  the n=2,....,nl point quadrature formulae of Hermite-Filon type with
  step-size h, and the boolean type is true if cos px is the weight
  function and false if sin px is the weight function.  fd represents
  the first derivative of f(x). c
  int m,j,n,ml,m3,m4;
  real part1,part2,h,hl,sm3,sm4,den,fax,fdax,ax,ah,bar,fcr1,fcr2,cs1,cs2,ad,
                                                              d1,d2;
  c d1 and d2 refer to the double summations in (3.3.2.2). c
  [2:nl] real integ; [2:nl] ref[]real x,hj,hbj,hjs,hbjs;
  c hj and hbj refer to the coefficients H^cos_{n,j} and H̄^cos_{n,j}, and hjs and
  hbjs refer to H^sin_{n,j} and H̄^sin_{n,j} respectively. c
  proc he=(int j,n,bool sorc,ref[]ref[]real x,ref real c2) real:
  begin
    c Self-generation of the coefficients (3.3.26) and (3.3.27).  If the
    boolean sorc is true the coefficient H^cos_{n,j} is delivered while H̄^cos_{n,j}
    is assigned to c2, and if sorc is false H^sin_{n,j} is delivered and
    H̄^sin_{n,j} is assigned to c2. c
    [0:ml] real sigarray; c ml is n-1 c
    [1:ml] real xx; real c1;
    proc sigma=(ref[]real x,int n,k)real: begin ... end;
    proc fun=(int r,real p,q)real: begin ... end;
```

```
proc sum= real:
begin
  c This is a procedure to evaluate L'_j (x_j) in (3.3.15) from the
  relation  L'_j (x_j) = Σ_{i=1, i≠j}^{n} 1/(x_j-x_i)  c
  real s←0.0; for i to n do
  (i≠j | s plus 1.0/(x[n][j]-x[n][i]) | s plus 0.0); s
end;

proc pr2= real:
begin
  c This procedure calculates the product
  Π'_n (x_j)=(x-x_1)... (x-x_{j-1})(x-x_{j+1})... (x-x_n)  c
  real r1←1.0; for i to n do
  (i=j | skip r1 times (x[n][j] -x[n][i])); r1
end;

for i2 to m1 do
begin
  if i2=j then if j<n then xx[i2]←x[n][i2+1] else skip fi
          else if i2<j then xx[i2]←x[n][i2] else xx[i2]←x[n][i2+1] fi
  fi
  c The array xx is formed by leaving out the x_j 'th term and
  enumerating the terms from 1 to n-1  c
end;
d1←d2←0.0;
for in from 0 to m1 do sigarray[in]←sigma(xx,m1,in);
c The product combinations are stored in an array, sigarray.  c
```

```
for i3 from 0 to m1 do
begin
  m3←m1-i3; sm3←sigarray[m3]; fcr1← (m3≠(m3÷2)*2 | -sm3 | sm3);
  for i4 from 0 to m1 do
  begin
    int ub1,ub2;
    m4←m1-i4; sm4←sigarray[m4]; ub1←i4+i3; ub2←ub1+1;
    fcr2←fcr1*(m4≠(m4÷2)*2 | -sm4 | sm4);
    d1 plus (fun(ub1,p,h)-fun(ub1,p,0.0))*fcr2;
    d2 plus (fun(ub2,p,h)-fun(ub2,p,0.0))*fcr2
    c d1 and d2 refer to the two terms with double sums in equations
    (3.3.21) and (3.3.22). c
    end
  end;
  ad←sum2; den←pr2; den times den;
  c1←(1.0+2.0*ad*x[n][j])*d1-2.0*ad*d2; c2←d2-x[n][j]*d1;
  c1 div den; c2 div den;
  c1
end;

for n from 2 to n1 do
begin
  c This do-loop generates n=2,....,n1 point formulae successively. c
    integ[n]←0.0; c the integral c
    h1←(b-a)/round (10/(n-1)); c initially estimated step-size c
    m←entier ((b-a)/h1+0.5); h←(b-a)/(m+1); c step-size c
    hj[n]←loc[1:n] real;
    hbj[n]←loc[1:n]real;
```

```
hjs[n]←loc[1:n]real;
hbjs[n]←loc[1:n]real:
x[n]←loc[1:n]real;
for ik to n do
begin
  x[n][ik]←h*(ik-1)/(n-1)
  c if equally spaced abscissae are used c
end;
ml←n-1;
for j to n do
begin
  hj[n][j]←he(j,n,true,x,bar); hbj[n][j]←bar;
  hjs[n][j]←he(j,n,false,x,bar); hbjs[n][j]←bar
end;
for i2 from 0 to m do
begin
  part1←part2←0.0; ah←a+i2*h; cs1←cos(p*ah); cs2←sin(p*ah);
  for j to n do
  begin
    ax←ah+x[n][j]; fax←f(ax); fdax←fd(ax);
    part1 plus hj[n][j]*fax+hbj[n][j]*fdax;
    part2 plus hjs[n][j]*fax+hbjs[n][j]*fdax
  end;
cs1←cos(p*ah); cs2←sin(p*ah);
integ[n]plus if type then
                    c the weight function is cos px c
                    part1*cs1-part2*cs2
```

```
                else
                c the weight function is sin px c
                part1*cs2+part2*cs1
            fi
        end
    end; c end of n do loop c
    integ
end
```

CHAPTER 4

THE USE OF CHEBYSHEV SERIES FOR THE EVALUATION OF OSCILLATORY INTEGRALS

Clenshaw and Curtis (1960) have given a scheme for the numerical integration of a well-behaved function f(x), with the interval of integration normalized to [-1, 1], which is based on the approximation of f(x) in a series of Chebyshev polynomials, $T_n(x)$. In this context, the function is said to be well-behaved if the coefficients in the Chebyshev expansion fall off rapidly. This method is extended to the consideration of integrals of the form

$$\int_a^b f(x) \begin{matrix} \cos \\ \sin \end{matrix} px \, dx$$

A new algorithm is presented which evaluates the resulting basic integrals by a direct automatic computation (similar to the methods of Chapter 3) which simulates the analytic evaluation. The stability of the method is discussed and critical comparisons, including numerical tests on several practical examples, are carried out with the related earlier work of Bakhvalov and Vasil'eva (1968) and Piessens and Poleunis (1971).

4.1 Introduction

Normalization of the range of integration leads to consideration of integrals of the form

$$\int_{-1}^{1} f(x) \begin{matrix}\cos\\ \sin\end{matrix} \omega x \, dx \qquad (1)$$

The usual methods of evaluating (1) rely on approximating f(x) by a series

$$f(x) \simeq \sum_{i=0}^{n} a_i \, A_i(x) \qquad (2)$$

so that the integrals

$$\int_{-1}^{1} A_i(x) \begin{matrix}\cos\\ \sin\end{matrix} \omega x \, dx \qquad (3)$$

are obtainable analytically. The choice

$$A_i(x) = x^i \qquad (4)$$

yields the existing quadrature formulae of Clendenin (1966), Filon (1928) and Flinn (1960) corresponding to n=1, 2 and 5 respectively. The automatic computer generation of the quadrature formula for general order n has been described in Chapter 3.

The theory of approximation (Davis (1963)) suggests that a better form for $A_i(x)$ would be the Chebyshev polynomial $T_i(x)$. This process has been widely used for non-oscillatory integrands and gives the well-known formulae of Clenshaw and Curtis (1960). However, the evaluation of the integrals (3) in the oscillatory case seems to present a problem when Chebyshev polynomials are employed.

Bakhvalov and Vasil'eva (1968) have briefly considered this problem (although their main theme was the use of Legendre polynomials $P_i(x)$). They suggest that, if the zeros of the Chebyshev polynomials are used as the interpolatory points in a Lagrange interpolation formula, then orthogonality relations can be used to evaluate the required coefficients They state that the resulting quadrature formulae are somewhat more

complicated than the results they quote for the Legendre polynomial procedure and imply that the effect of round-off in the calculations may therefore be more serious.

Piessens and Poleunis (1971) also consider the use of Chebyshev polynomials for $A_i(x)$ but deviate from the Bakhvalov and Vasil'eva approach in that they effectively evaluate the basic integral (3) by a somewhat indirect method involving a truncated infinite series of Bessel functions, instead of utilizing the orthogonality properties of summation over the zeros of the Chebyshev polynomials.

It is therefore considered useful to carry out a critical survey of these earlier methods, starting with the basic Bakhvalov and Vasil'eva prescription involving Legendre polynomials, in order that the underlying structure of the approaches should be investigated and compared.

Thus, following Bakhvalov and Vasil'eva, the integral

$$I = \int_{-1}^{1} f(x)\ e^{i\omega x}\ dx \qquad (5)$$

is treated by introducing the Lagrangian interpolation polynomial of degree n

$$f(x) \simeq \sum_{i=0}^{n} a_i\ P_i(x) \qquad (6)$$

which collocates with f(x) at the (n+1) points x_j (j=0, 1, 2, ... n). If these points are chosen to be the zeros of the Legendre polynomial $P_{n+1}(x)$, that is

$$P_{n+1}(x_j) = 0 \qquad j=0,\ 1,\ 2,\ \ldots\ n\ , \qquad (7)$$

then the coefficients a_i may be found to be

$$a_i = \sum_{j=0}^{n} \alpha_j\ \tfrac{1}{2}\ (2i+1)\ P_i(x_j)\ f(x_j) \qquad (8)$$

which follows on utilizing the orthogonality relation

$$\sum_{j=0}^{n} \alpha_j \, P_i(x_j) \, P_k(x_j) = 2 \, \delta_{ik} / (2i+1) \, . \qquad (9)$$

(See for example, Abramowitz and Stegun (1965), p. 790)

In these formulae α_j (j=0, 1, ... n) denotes the weights of the (n+1)-point Gauss-Legendre quadrature formula for the weight function $\omega(x) = 1$ on the interval [-1, 1], (Davis and Rabinowitz (1965)), (See also Chapter 6).

It is interesting to note the equivalent way of considering equation (6), which utilizes the integral orthogonality result

$$\int_{-1}^{1} P_k(x) \, P_i(x) \, dx = 2 \, \delta_{ik} / (2i+1) \qquad (10)$$

and produces

$$a_i = \tfrac{1}{2} (2i+1) \int_{-1}^{1} f(x) \, P_i(x) \, dx \qquad (11)$$

for the coefficients in a least-squares fit. Recalling that f(x) is to be represented by a polynomial of degree n it is apparent that the integral in equation (11) is obtained <u>exactly</u> on utilizing an (n+1) point Gauss-Legendre quadrature formula which is exact for polynomials of maximum degree (2n+1). Equation (8) follows immediately.

Bakhvalov and Vasil'eva then use expansion (6) in the integral to produce

$$I \simeq \sum_{i=0}^{n} a_i \, M_i \qquad (12)$$

where

$$M_i(\omega) = \int_{-1}^{1} P_i(x) \, e^{i\omega x} \, dx \qquad (13)$$

On adopting formula (8) for a_i and re-arranging, the quadrature formula

$$I \simeq \sum_{j=0}^{n} D_j \, f(x_j) \qquad (14)$$

is obtained, where

$$D_j = \alpha_j \sum_{i=0}^{n} \frac{1}{2} (2i+1) P_i(x_j) M_i(\omega) \qquad (15)$$

Thus, computationally, the coefficients a_i as given by (8) are not evaluated directly, since the summations are performed in the order indicated by equations (14) and (15).

The basic integrals M_k of equation (13) may be obtained analytically since

$$M_k(\omega) = 2\, i^k\, j_k(\omega) \qquad (i=\sqrt{-1}) \qquad (16)$$

where $j_k(\omega)$ denotes the spherical Bessel function of order k defined by Abramowitz and Stegun (1965). The values of $j_k(\omega)$ are obtained from the recurrence relation

$$j_{k+1}(\omega) = (2k+1)\,\omega^{-1}\, j_k(\omega) - j_{k-1}(\omega) \qquad (17)$$

However, since this relation is unstable in the forwards direction, particularly for small ω, it is necessary, when required, to use the relation in the backwards direction in the manner suggested by Miller (See Abramowitz and Stegun (1965), p. 452). Additional details are given by Bakhvalov and Vasil'eva.

If f(x) is now represented by the Chebyshev fit

$$f(x) \simeq {\sum_{i=0}^{n}}' a_i\, T_i(x) \qquad (18)$$

which is analogous to (6) and the collocation points x_j (j=0, 1, ... n) are now taken to be the zeros of $T_{n+1}(x)$, yielding

$$x_j = \cos\left[\frac{2j+1}{n+1} \cdot \frac{\pi}{2}\right] \qquad j=0,\ 1,\ \ldots\ n \qquad (19)$$

the a_i are again obtainable on using orthogonality relations.
(The prime denotes that the first term in the summation is to be multiplied by 1/2)

The required relations are

$$\sum_{j=0}^{n} T_i(x_j)\, T_k(x_j) = 0 \qquad i \neq k$$
$$= \tfrac{1}{2}(n+1) \qquad i = k \neq 0 \qquad (20)$$
$$= (n+1) \qquad i = k = 0$$

and result in the well-known expression

$$a_i = \frac{2}{(n+1)} \sum_{j=0}^{n} f(x_j)\, T_i(x_j)$$
$$= \frac{2}{(n+1)} \sum_{j=0}^{n} f(x_j) \cos\left[i\, \frac{(2j+1)}{(n+1)} \cdot \frac{\pi}{2} \right] \qquad (21)$$

Once again, if the integral orthogonality relationship is used as ar alternative approach, the formula

$$\int_{-1}^{1} T_i(x)\, T_k(x)\, (1-x^2)^{-\frac{1}{2}}\, dx = 0 \qquad i \neq k$$
$$= \frac{\pi}{2} \qquad i = k \neq 0 \qquad (22)$$
$$= \pi \qquad i = k = 0$$

produces the result

$$a_i = \frac{2}{\pi} \int_{-1}^{1} T_i(x)\, f(x)\, (1-x^2)^{-\frac{1}{2}}\, dx \qquad (23)$$

If f(x) is to be represented by the polynomial of degree n given by (18) then this integral is obtained <u>exactly</u> from the Gauss-Chebyshev equal weight quadrature formula of order (n+1) for the weight function $\omega(x) = (1-x^2)^{-\frac{1}{2}}$ on the interval $[-1, 1]$. The required formula is

$$\int_{-1}^{1} F(x)\, (1-x^2)^{-\frac{1}{2}}\, dx = \frac{\pi}{(n+1)} \sum_{j=0}^{n} F(x_j) + \epsilon_{n+1} \qquad (24)$$

where x_j is given by equation (19) and the error ϵ_{n+1} by

$$\epsilon_{n+1} = 2\pi\, F^{(2n+2)}(\vartheta) \,/\, \left[2^{2n+2} (2n+2)! \right] \qquad (-1 < \vartheta < 1) \qquad (25)$$

(Abramowitz and Stegun (1965), p. 889). Formula (21) then follows immediately.

The expansion (18) with a_i given by (21) is now utilized in the integral I and produces

$$I \simeq \sum_{i=0}^{n}{}' a_i N_i \tag{26}$$

where

$$N_i(\omega) = \int_{-1}^{1} T_i(x)\, e^{i\omega x}\, dx \tag{27}$$

This may then be written in the Bakhvalov and Vasil'eva form as

$$I \simeq \sum_{j=0}^{n} D_j f(x_j) \tag{28}$$

where

$$D_j = \frac{2}{(n+1)} \sum_{i=0}^{n}{}' N_i(\omega)\, T_i(x_j) \quad , \tag{29}$$

the order of summation having been changed once again. These results are entirely analogous to the Legendre based prescriptions (14) and (15).

The basic integrals $N_k(\omega)$ may be evaluated in a manner analogous to the Bakhvalov and Vasil'eva approach by means of recurrence relationships. For instance, on writing

$$I_k(\omega) = i^{-k} N_k(\omega) = i^{-k} \int_{-1}^{1} T_k(x)\, e^{i\omega x}\, dx \tag{30}$$

and integrating the appropriate recurrence relations for the Chebyshev polynomials, it is possible to establish the result

$$\omega I_{k+2}(\omega) = (2k+4)\, I_{k+1}(\omega) + (2k-4)\, I_{k-1}(\omega) - 2\omega\, I_k(\omega) - \omega I_{k-2}(\omega) \quad . \tag{31}$$

This relation is again unstable in the forwards direction particularly for small ω, just as in the case of equation (17), and the use of Miller's algorithm is again necessitated.

Alternatively, on considering

$$J_k(\omega) = \int_{-1}^{1} T_k(x) \begin{matrix}\cos\\ \sin\end{matrix} \omega x \, dx \qquad (32)$$

with the cosine being taken in the even case and the sine in the odd case, it is easy to derive the formulae

$$J_k = -\frac{4 \sin\omega}{\omega (k-2)} + \frac{k}{k-2} J_{k-2} - \frac{2k}{\omega} J_{k-1} \qquad \text{(k even)} \qquad (33)$$

$$J_k = \frac{4 \cos\omega}{\omega (k-2)} + \frac{k}{k-2} J_{k-2} + \frac{2k}{\omega} J_{k-1} \qquad \text{(k odd)} \qquad (34)$$

Once again, these relations are unstable in the forwards direction and, although they have the advantage over equation (31) of being 3-term as opposed to 5-term formulae, they have the additional disadvantage of being "inhomogeneous".

The question of the recurrence relation approach to the Bakhvalov ——and Vasil'eva — Chebyshev procedure is being currently investigated by Patterson and his co-workers. Patterson has proposed (Patterson, T.N.L. (1974), Private Communication) that the integral $N_i(\omega)$ in (27) should be evaluated by expanding $T_i(x)$ in a series of Legendre polynomials according to

$$T_i(x) = \sum_{k=0}^{i} \frac{1}{2} (2k+1) R_{ki} P_k(x) \qquad (35)$$

where

$$R_{ki} = \int_{-1}^{1} P_k(x) T_i(x) \, dx \qquad (36)$$

The integral (27) may now be obtained in terms of the <u>analytical</u> result (16) for $M_i(\omega)$ and it follows that

$$N_i(\omega) = \sum_{k=0}^{i} \frac{1}{2} (2k+1) R_{ki} M_k(\omega) \qquad (37)$$

Patterson establishes that R_{ki} satisfies the <u>stable</u> recurrence relationship

$$R_{ki} = \left[(2k-1) / (2k) \right] \left[R_{k-1,i+1} + R_{k-1,i-1} \right] - \left[(k-1) / k \right] R_{k-2,i} \qquad (38)$$

and hence claims that this method may well be no less stable than the Bakhvalov and Vasil'eva - Legendre procedure.

In the present work, however, an alternative approach is proposed for the direct evaluation of (27) by a method which is analogous to the techniques described in the previous Chapter.

However, before going on to describe the alternative procedure, the related work of Piessens and Poleunis (1971) is discussed, since some additional information on the problem is provided. These authors do not use the orthogonality procedure described in equations (18) - (24) in a direct manner, but attempt to avoid the evaluation of $N_i(\omega)$ which would then result by using the infinite expansion

$$(1-x^2)^{\frac{1}{2}} f(x) = \sum_{k=0}^{\infty}{}' c_k T_k(x) \tag{39}$$

The resulting integrals

$$\int_{-1}^{1} T_k(x) (1-x^2)^{-\frac{1}{2}} \begin{matrix}\cos\\ \sin\end{matrix} \omega x \, dx \tag{40}$$

may be evaluated analytically and yield the expressions

$$\int_{-1}^{1} f(x) \cos \omega x \, dx = \sum_{k=0}^{\infty}{}' c_{2k} (-1)^k \pi J_{2k}(\omega) \tag{41}$$

$$\int_{-1}^{1} f(x) \sin \omega x \, dx = \sum_{k=0}^{\infty} c_{2k+1} (-1)^k \pi J_{2k+1}(\omega) \tag{42}$$

involving infinite series of Bessel functions.

The integral orthogonality result (22) is then used in (39) and gives

$$c_k = \frac{2}{\pi} \int_{-1}^{1} f(x) T_k(x) \, dx \tag{43}$$

This integral is evaluated by using the finite expansion (18) for f(x) and leads to the result

$$c_k \simeq \frac{2}{\pi} \sum_{i=0}^{n}{}' a_i \int_{-1}^{1} T_k(x)\, T_i(x)\, dx \qquad (44)$$

The integrals in (44) are easily evaluated analytically and the results for even and odd k as required by equations (41) and (42) are quoted by Piessens and Poleunis.

It is worth pointing out that these final results are also obtainable from the direct Bakhvalov and Vasil'eva approach embodied in equations (26) and (27). The connecting link is the evaluation of

$$N_k(\omega) = \int_{-1}^{1} T_k(x)\, e^{j\omega x}\, dx = \int_0^{\pi} \cos k\theta\, e^{i\omega\cos\theta} \sin\theta\, d\theta \qquad (45)$$

in which the substitution $x = \cos\theta$ has been used, by means of the formulae

$$\cos(\omega\cos\theta) = J_0(\omega) + 2\sum_{k=1}^{\infty} (-1)^k J_{2k}(\omega) \cos(2k\theta) \qquad (46)$$

$$\sin(\omega\cos\theta) = 2\sum_{k=0}^{\infty} (-1)^k J_{2k+1}(\omega) \cos[(2k+1)\theta] \qquad (47)$$

(Abramowitz and Stegun (1965), p. 361). The Piessens and Poleunis final results then follow immediately.

It is the evaluation of the resulting infinite series of Bessel functions which suggests a defect in the Piessens and Poleunis version of the Bakhvalov and Vasil'eva - Chebyshev approach. Piessens and Poleunis demonstrate that the terms in the series in (41) and (42) decrease rapidly for $k > \omega/2$ and suggest that truncation may be effected after M terms where M is "only a little larger than $\omega/2$". Clearly, for large ω the method is unsatisfactory especially since the evaluation of the Bessel functions $J_k(\omega)$ is required in the terms of the series.

Thus, the Piessens and Poleunis approach amounts essentially to the evaluation of the basic integrals $N_i(\omega)$ of equation (27) by means

of an infinite series of Bessel functions and in the present work an alternative method is proposed to avoid this. The integrals $N_i(\omega)$ are obtained directly here by employing an automatic computational technique which simulates the analytic evaluation by a method which is similar to that described in Chapter 3.

4.2 The Quadrature Formula

The basis of the method is the evaluation of $N_i(\omega)$ by picking out the coefficients, $D_{i,r}$, of x^r in $T_i(x)$ and then making use of the results

$$\int_{-1}^{1} x^r \cos \omega x \, dx = \left[\sum_{\ell=0}^{r} \ell! \binom{r}{\ell} \frac{x^{r-\ell}}{\omega^{\ell+1}} \sin \left(\omega x + \tfrac{1}{2} \ell \pi \right) \right]_{-1}^{1} \qquad (1)$$

$$\int_{-1}^{1} x^r \sin \omega x \, dx = \left[-\sum_{\ell=0}^{r} \ell! \binom{r}{\ell} \frac{x^{r-\ell}}{\omega^{\ell+1}} \cos \left(\omega x + \tfrac{1}{2} \ell \pi \right) \right]_{-1}^{1} \qquad (2)$$

(cf. equations (3.3.23) and (3.3.24)).

The Chebyshev polynomials are of the form (Abramowitz and Stegun (1965))

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ &\vdots \end{aligned} \qquad (3)$$

and the coefficients, $D_{i,r}$, of x^r in $T_i(x)$ can be easily calculated by means of the recurrence relation

$$D_{i,r} = 2D_{i-1,r-1} - D_{i-2,r} \qquad i \geq 2, \quad r \leq i \qquad (4)$$

yielding a "Pascal" triangle, which facilitates computation. The basic

integrals $N_i(\omega)$ of (4.1.27) are then easily obtained. In practice these are usually separated into their real and imaginary parts for the separate calculation of the integrals involving cos ω or sin ω using, either (1) or (2) for even or odd r respectively.

The formula (4.1.26) is then used directly here and embodies the Chebyshev fit (4.1.18) at the Gaussian based abscissae of (4.1.19). In their original work on non-oscillatory integrals Clenshaw and Curtis utilized the alternative approximation

$$f(x) \simeq \sum_{i=0}^{n}{}'' a_i T_i(x) \tag{5}$$

with collocation at the points x_j (j=0, 1, 2, ... n) where x_j is now given by

$$x_j = \cos \frac{\pi j}{n} \quad , \quad (j=0, 1, 2, \ldots n) \tag{6}$$

the double primes denoting that the first and the last terms in the summations are to be multiplied by 1/2 . The alternative orthogonality relation

$$\begin{aligned} \sum_{j=0}^{n}{}'' T_i(x_j)\, T_k(x_j) &= 0 \qquad i \neq k \\ &= \frac{n}{2} \qquad i = k \neq 0 \text{ or } n \\ &= n \qquad i = k = 0 \text{ or } n \end{aligned} \tag{7}$$

(cf. equation (4.1.20)) produces the result

$$\begin{aligned} a_i &= \frac{2}{n} \sum_{j=0}^{n}{}'' f(x_j)\, T_i(x_j) \\ &= \frac{2}{n} \sum_{j=0}^{n}{}'' f(x_j) \cos \frac{i\pi j}{n} \end{aligned} \tag{8}$$

which is then used in the quadrature formula (4.1.26) as an alternative to the Gauss-based prescription (4.1.21). It will be noticed that equation (5) is a closed formula in that it involves the end points x = -1 and x = 1, whereas (4.1.18) is open. Elliott (1965) has pointed out

that, in the evaluation of the non-oscillatory integral, the truncation error involved in the use of the classical or open formula (4.1.18) is of the order of $1/n^2$. However, when the practical or closed series (5) is utilized the truncation error is of order $1/n^3$.

In the present work, the use of the Clenshaw and Curtis formula (5) is proposed, although (4.1.18) could be adopted if an open formula is specifically required, such as in the case where the integrand has singularities at its end points. Elliott's analysis gives intuitive backing to our method, that collocation at the practical abscissae is better than collocation at the classical Chebyshev zeros, if we are more interested in the integral of f(x) than in approximating f(x) itself.

Piessens and Poleunis demonstrate, that as for Clenshaw - Curtis quadrature in the non-oscillatory case, the integral of the finite Chebyshev expansion converges more quickly than the expansion itself. It is hoped therefore to retain the advantages of the Clenshaw and Curtis formulation in the oscillatory case. In particular, by choosing the order of the formulae as $n = 2^i$, i=1, 2, ... the adaptive nature of the procedure could be retained.

The errors involved in the Clenshaw and Curtis formulae have been discussed by many authors such as O'Hara and Smith (1968), Gentleman (1972) and Elliott (1965). The errors are, in fact, less than might be expected. O'Hara and Smith show that the error terms are such that the accuracy may even approach that of the corresponding n-point Gauss formula in certain instances. In general however, more functions evaluations are normally required for the Clenshaw - Curtis case than for the corresponding Gaussian quadrature. Nonetheless, it is considered that the present prescription is worth investigating as a practical alternative in the oscillatory case.

It is important to note that, as pointed out by Bakhvalov and Vasil'eva, it may be better not to sub-divide the range of integration but to increase the order of the formula used when Chebyshev (or Legendre) fitting is used for f(x). This is in contrast with the methods of the previous Chapter where equally spaced abscissae were used and gave rise to formulae of the Newton - Cotes type. Due to instabilities in the higher order coefficients, it was not possible to proceed to large n there and the recommendation in practice was to limit the order to n=5 or 6 and sub-divide the interval of integration uniformly. This method was also adopted by Bakhvalov and Vasil'eva for comparison purposes in one of their numerical applications, where they used a formula of order n=4, with a large number of sub-divisions, to consider an integral involving $f(x) = \cos \pi u x^2$, with large u. The use of this technique is not entirely satisfactory in general, and great care must be exercised in the highly oscillatory cases when ω is large. Indeed, preliminary calculations based on the present method have indicated that uniform sub-division may produce similar instabilities to those exhibited in Chapter 3 and that the order of the formulae would have to be similarly curtailed. Hence, uniform sub-division is not adopted here. However, it is worth noting that it may be possible to avoid the cancellation effects produced by uniform sub-division by using special techniques appropriate to the particular function considered. As an example, good results are obtained for $f(x) = \cos \pi u x^2$ by integrating between the "peaks" (which occur at the zeros of $\sin \pi u x^2$) or between the zeros of $\cos \pi u x^2$, using either Newton - Cotes or the present methods. This example is discussed in detail in the following section.

Consequently, to return to the derivation of the quadrature formula, when the integral

$$I_c = \int_a^b f(x) \cos px \, dx \tag{9}$$

is considered, a linear transformation enables the result to be written in the form

$$I_c = \tfrac{1}{2}(b-a) \cos K \int_{-1}^{1} F(t) \cos \omega t \, dt - \tfrac{1}{2}(b-a) \sin K \int_{-1}^{1} F(t) \sin \omega t \, dt \tag{10}$$

where

$$K = \tfrac{1}{2} p (b+a) \tag{11}$$

$$\omega = \tfrac{1}{2} p (b-a) \tag{12}$$

and

$$F(t) \equiv f\left[\tfrac{1}{2}(b+a) + \tfrac{1}{2}(b-a)\, t\right] . \tag{13}$$

Approximating F(t) by the polynomial of degree n

$$F(t) \simeq \sum_{i=0}^{n}{}'' a_i T_i(t) \tag{14}$$

collocating at the (n+1) points t_j ,

$$t_j = \cos \frac{\pi j}{n} \qquad j=0\ (1)\ n \tag{15}$$

yields

$$I_c \simeq \tfrac{1}{2}(b-a) \cos K \sum_{i=0}^{n}{}'' a_i \sum_{r=0}^{i} D_{i,r} \int_{-1}^{1} t^r \cos \omega t \, dt - \tfrac{1}{2}(b-a) \sin K \sum_{i=0}^{n}{}'' a_i \sum_{r=0}^{i} D_{i,r} \int_{-1}^{1} t^r \sin \omega t \, dt \tag{16}$$

where

$$a_i = \frac{2}{n} \sum_{j=0}^{n}{}'' F(t_j) \cos \frac{\pi j i}{n} \tag{17}$$

Similarly,

$$I_s = \int_a^b f(x) \sin px \, dx \tag{18}$$

yields

$$I_s \simeq \tfrac{1}{2}(b-a) \sin K \sum_{i=0}^{n}{}'' a_i \sum_{r=0}^{i} D_{i,r} \int_{-1}^{1} t^r \cos \omega t \, dt$$

$$+ \tfrac{1}{2}(b-a) \cos K \sum_{i=0}^{n}{}'' a_i \sum_{r=0}^{i} D_{i,r} \int_{-1}^{1} t^r \sin \omega t \, dt \tag{19}$$

The basic integrals required in (16) and (19) are supplied by (1) and (2).

It will be noticed that the finite series occuring in equations (1) and (2) converge rapidly when ω is large. This will be emphasized when the function f(x) is sufficiently smooth for accurate fitting to be possible with a formula whose order, n, is reasonably small. On the other hand, if f(x) requires a formula of high order with a large value of n to achieve an accurate fit, the coefficients

$$\frac{n!}{(n-\ell)!} \cdot \frac{1}{\omega^{\ell+1}} \qquad \ell=0 \ (1) \ n \tag{20}$$

which appear in (1) and (2) may become very large. (Note that the largest value of r, namely r=n, has been taken here to accentuate the effect). This will be particularly noticeable when ω is small and serious instabilities may arise in this case of small ω and large n. This is clearly due to the generation of very large numbers, with the resulting cancellation when the terms in the alternating series are summed.

An alternative procedure which avoids this instability is to use series expansions for the trigonometric functions in (1) and (2). The expressions

$$\int_{-1}^{1} x^r \cos \omega x \, dx = 2 \sum_{\ell=0}^{\infty} \frac{(-1)^\ell \omega^{2\ell}}{(2\ell+r+1)(2\ell)!} \tag{21}$$

when r is even, and

$$\int_{-1}^{1} x^r \sin \omega x \, dx = 2 \sum_{\ell=0}^{\infty} \frac{(-1)^\ell \omega^{2\ell+1}}{(2\ell+r+2)(2\ell+1)!} \qquad (22)$$

when r is odd, are readily obtained and are obviously most useful in precisely those circumstances (small ω, large r) under which the finite series (1) and (2) are least stable. In practice, it is easily demonstrated that the maximum value of ℓ required to yield double precision accuracy (about 22 figures) for the basic integrals is given roughly by

$$\ell_o = 2\omega + 10 \, , \qquad (23)$$

round-off to integral values being implied. This estimate for the truncation point of the infinite series is reliable for $\omega \leq 10$. For larger values of ω, it tends to be a gross over-estimate. For instance, at $\omega = 100$, $\ell_o = 210$ whereas the actual maximum value of ℓ required is only about 158. However, for such large values of ω, it is likely that the finite series (1) and (2) would be used instead and, hence, it is suggested that (23) provides a reasonable estimate of the number of terms required in all practical cases.

Indeed, it is clear that formulae (21) and (22) exhibit instabilities for large ω which are "complementary" to those shown by the finite series (1) and (2). It is possible to discuss this effect qualitatively by considering the behaviour of the related simpler series for cos ω whose general term is of the form

$$(-1)^\ell \omega^{2\ell}/(2\ell)! \qquad \ell = 0, 1, 2, \ldots \qquad (24)$$

Thus, the factors such as $(2\ell+r+1)^{-1}$ in (1) and (2) which assist convergence in any case have been omitted. In the case of the ω^{-1} series in (1) and (2) the coefficients of $\pm \sin \omega$ and $\pm \cos \omega$ are given by equation (20) and range from $1/\omega$ when $\ell=0$ to $n!/\omega^{n+1}$ when $\ell=n$.

A measure of the instability of the series is provided by the ratio of these quantities, namely,

$$n!/\omega^n \qquad (25)$$

which are the reciprocals of the terms in (24) or the corresponding terms in the series for sin ω, thus demonstrating the "reciprocal" nature of the instabilities.

An examination of the magnitude of the terms in (24) with $0 \le \ell \le \ell_o$ demonstrates that, for a given ω, the maximum value is attained when $2\ell = [\omega]$ and the required maximum is therefore

$$L = \omega^{[\omega]}/[\omega]! \qquad (26)$$

Consequently, when the alternating series for cos ω is summed, this initial build up in size of the terms before the final convergence, results in severe cancellation if L is large and produces a loss of roughly s significant figures, where s is the exponent of L. For example if ω=10, L is equal to 0.27557 ... $*10^{4}$ and cos 10 is obtained to be -0.8390715112 using 11 figure arithmetic. This is correct to only (11-4)=7 significant figures when compared with the accurate value -0.8390715291 . The value ω=10 is, of course, rather large to use in power series approach and more realistic values produce smaller cancellation effects. Thus, for ω=5 only 2 significant figures are lost and for values less than 4 there is scarcely any diminution in accuracy. (The tables given on pages 818-819 of Abramowitz and Stegun are a useful aid here)

The complementary effect is observed for the original ω^{-1} series (1) and (2) when the inverse ratio (25) is considered. Ultimately this ratio will become very large for a given ω if n is allowed to increase indefinitely and total instability would then arise. However, in practice, the value of n will be restricted by the user and examination of the

ratio (25) shows that, for a given $\omega>1$, no serious build up in magnitude occurs until n reaches values well beyond $[2\omega]$. (It will be recalled from the discussion leading to equation (26) that, as n increases, the ratio (25) actually decreases to a minimum at $n=[\omega]$ before starting to increase.) The situation is clearly best for large ω when it is possible to tolerate large values of n before instability arises.

It appears, therefore, that the main ω^{-1} finite series in (1) and (2) will be stable if n is restricted to values less than a critical value, n_c, which is given by

$$n_c = [2\omega] \tag{27}$$

In practice, this is found to be much too stringent and it is possible to replace it by a relation of the form

$$n_c = [2\omega] + \Upsilon \tag{28}$$

where values of Υ as large as $\Upsilon=10$ are tolerable, particularly for large ω.

For values of ω which are less than 1, the ratio (25) increases monotonically with n and the resulting series are completely unstable. However, the alternative series (21) and (22) are then available and are extremely stable for all n.

In practice, it is recommended that for large ω, say $\omega>4$, the basic ω^{-1} series (1) and (2) should be utilized, bearing in mind the restrictions implied by (28). When ω is smaller, a switch should then be made to the alternative ω series (21) and (22). This point is elaborated in the discussion of the numerical examples presented in the next section.

It will be observed that in the limit as $\omega\to 0$ the value $2/(r+1)$ is obtained from series (4.2.21) and that the corresponding summation in equation (4.2.16) becomes

$$\sum_{r=0}^{i} 2D_{i,r}/(r+1) \qquad (29)$$

where the summation extends over even values of r and i is also even in this, the symmetrical cosine, case. The exact value of this summation is given by integrating $T_i(x)$ (i even) and the result is

$$\sum_{r=0}^{i} 2D_{i,r}/(r+1) = -2/(i^2-1) \qquad \text{(i even)} \qquad (30)$$

When this expression is used in quadrature formula (4.1.26), the result is, of course, the Clenshaw - Curtis prescription for the integral

$$\int_{-1}^{1} f(x)\,dx \qquad (31)$$

This is compared with the Bakhvalov and Vasil'eva approach which in the limit as $\omega \to 0$ reproduces the Gauss-Legendre formula for this integral.

However, if the numerical evaluation of summation (29) or, indeed, the more general series

$$B_i(\omega) = \sum_{r=0}^{i} D_{i,r} \int_{-1}^{1} x^r \begin{matrix}\cos\\ \sin\end{matrix} \omega x\,dx \qquad (32)$$

is attempted directly by the integration routine described here, serious cancellation effects are observed when i is large. The cancellation is due to the alternating signs and varying magnitudes of the Chebyshev coefficients (e.g. $D_{i,o}=\pm 1$; $D_{i,i}=2^{i-1}$.) Since the magnitudes of the integrals fall off with increasing ω, the instability effect is therefore most pronounced for small ω. A rough measure of this instability is given as $\omega \to 0$ by

$$i^2 \cdot 2^i \qquad (33)$$

and the number of figures lost by cancellation in the r series (32) is of the order of the exponent of this quantity. Thus, at i=12, about 5 figures are lost in evaluating $B_i(0)$, whilst at i=20 about 8 figures

are lost. A rough guide to the number of figures lost is provided by the expression

$$0.3i + 2 \qquad (34)$$

For large values of ω, this accuracy loss will be reduced roughly by the exponent of ω.

At first sight, it appears that this is a very serious defect in the method, but it should be recalled that the actual values of the $B_i(\omega)$ are to be used in a quadrature formula of the form

$$\int_{-1}^{1} f(x) \begin{matrix}\cos\\ \sin\end{matrix} \omega x \, dx = \sum_{i=0}^{n}{}'' a_i \, B_i \, (\omega) \qquad (35)$$

in conjunction with the coefficients, a_i, of the Chebyshev series for f(x). Consequently, if f(x) is reasonably smooth, so that accurate fitting is possible for fairly small values of n and the a_i coefficients (i=0, 1, 2, ... n) fall off rapidly with increasing i, then very accurate values of the integral (35) are obtained. Convergence is aided by the fall off of $B_i(\omega)$ with increasing i. This is particularly true in the case of small ω where cancellation in series (32) is at its worst, since, in this instance, B_i falls off most rapidly (approximately as $1/i^2$).

In practice, because of this effect, it has been found possible to proceed to values of n in formula (35) which are much larger than might be suspected from the restriction (34). This will be illustrated by the examples described in the next section. Even in the case of a badly-behaved function, where it was necessary to use n values around 50 to achieve a modest fit, the contributions from the smaller values of i were substantial. These could be calculated accurately and resulted in reasonable values for the integral, in this extreme case.

However, if the function f(x) is such that a large "tail" exists

in its Chebyshev expansion, so that the contribution from the a_n end of the series is still large compared with the a_o end, then errors could occur. An even worse situation would arise for the class of functions which are expressible only in the form

$$f(x) \simeq \sum_{i=N}^{N+n} a_i T_i(x) \tag{36}$$

where N is large. In fact, the integral

$$\int_{-1}^{1} T_N(x) \cos \omega x \, dx \tag{37}$$

itself, corresponding to $a_N=1$ and $a_i=0$ $(i\neq N)$ provides an extreme example. The accuracy loss, according to (34), would be roughly $(0.3N+2)$ figures, less a large ω contribution of about $\log(1+\omega)$ figures. It would be necessary, in such examples, to use double precision (or even, in extreme cases, $N \sim 50$ in (37), multiple precision) arithmetic to carry out the r summation in (32).

In practice, as mentioned above, the functions f(x), such as those arising in the applications of Chapter 1, are sufficiently smooth for the a_i terms for small i to dominate the series and the errors resulting from the large i instability are, therefore, insignificant, in these cases. It is necessary, of course, to take certain practical precautions in using the algorithm and these were adopted in the treatment of the examples in the next section. Thus, in conducting convergence tests on a given integral with increasing n, it is suggested that the stability at large n should be checked by proceeding beyond the point at which convergence of the successive values of the integral has been established. Again, for the reasonably well behaved functions treated, good results were obtained for values of n less than 20 using single precision arithmetic. However, the computations were then repeated in double

precision in order to check the results. This procedure is recommended in practice in selected instances. Double precision arithmetic was utilized in series (32) when values of n in excess of 20 were employed.

4.3 Computational Procedure and the Results

The quadrature rules of (4.2.16) and (4.2.19) are automatically generated on the computer for any order. It is noted that some of the basic integrals of the form (4.2.1) and (4.2.2) are not required, as the i-th order Chebyshev polynomial involves only $\left[(i+2)/2\right]$ non-zero coefficients. Efficiency is achieved by using the following representations for the sums, where the non-zero Chebyshev coefficients are declared by the array $\left[1:(i+2)\div 2\right]$ real d.

$$S1 = \sum_{r=0}^{R-1} d_{i,r+1} \int_{-1}^{1} t^{2r} \cos \omega t \, dt \qquad (i \text{ even}) \qquad (1)$$

$$S2 = \sum_{r=1}^{R} d_{i,r} \int_{-1}^{1} t^{2r-1} \sin \omega t \, dt \qquad (i \text{ odd}) \qquad (2)$$

where $R=\left[(i+2)/2\right]$, the non-zero coefficients of the $\{D\}$ being denoted by d. Also, the values of the basic integrals are stored for all the i's considered and then used in the procedure which evaluates I_c or I_s, thus resulting in computational economy.

Furthermore, the number of cosines required in formula (4.2.17) has been minimized by taking symmetry into account.

Again, it will be noticed that equations (4.1.1) and (4.1.2) involve only two independent trigonometric functions, namely cos ω and sin ω.

An Algol68 version of the algorithm is presented in the Appendix. The algorithm is applied first of all to the integral considered in Chapter 3, namely,

$$\int_0^1 e^x \cos px \, dx = \left[e(\cos p + p \sin p) - 1 \right] (p^2 + 1)^{-1} \qquad (3)$$

(cf. equation (3.4.5))

The absolute errors (defined by $\left|$ exact value - computed value $\right|$) in the numerical evaluation of the above integral are presented in Table 1 for $p=10^i$, i=0 (1) 4. The notation $x(-m)$ is used to denote $x*10^{-m}$. The calculations were carried out in single precision arithmetic (about 11 figures) to start with and it is seen that machine accuracy is rapidly approached as the order, n, of the formula is increased, particularly for the larger values of p. The function f(x) = exp (x) is so smooth on $[0, 1]$ that accurate fitting is possible for relatively small values of n (say 8 or 9) and excellent results are obtained as a consequence of the good behaviour of equations (4.2.1) and (4.2.2). The stability of the ω^{-1} series was tested by extending the order well beyond the limits where the successive values of the integral had converged. Stability was observed for $p \geq 10$ for values of n up to at least 25, thus providing a test of the robustness of the algorithm.

In the case p=1, corresponding to $\omega = \frac{1}{2}$, the basic ω^{-1} series (4.2.1) and (4.2.2) exhibited instability for values of n beyond n=12. Thus, although convergence to the exact result was observed at about n=8, the calculated values of the integral began to diverge from the exact at about n=12. Clearly, this was a case for a switch to be made to the alternative ω series (4.2.21) and (4.2.22) and it was confirmed that stable results were then obtained for values up to at least 27.

The results presented show considerable improvement over the Filon - type quadrature prescriptions as depicted in Table 2 of Chapter 3. It is noticed that in the lowest order cases n=1 and n=2, the two algorithms become, in fact, identical. The reason is that the Clenshaw and Curtis abscissae

Table 1. Absolute errors in the numerical evaluation of

$$\int_0^1 e^x \cos px \, dx$$

Order n	p				
	1	10	100	1000	10000
1	1.2(- 1)	5.8(- 4)	1.6(- 4)	1.3(- 6)	2.3(- 9)
2	6.1(- 4)	1.7(- 3)	1.9(- 6)	3.5(- 8)	2.7(- 9)
3	1.4(- 4)	1.0(- 4)	2.4(- 6)	2.0(- 8)	2 (-11)
4	6.9(- 7)	1.9(- 5)	1.2(- 8)	2.8(-10)	2 (-11)
5	2.6(- 8)	2.2(- 7)	8.8(- 9)	7 (-11)	1 (-11)
6	1.5(-10)	2.1(- 8)	4 (-11)	1 (-11)	1 (-11)
7	5 (-11)	1.1(-10)	1 (-11)	1 (-11)	exact
8	exact	1 (-11)	1 (-11)	1 (-11)	exact
9	exact	2 (-11)	1 (-11)	1 (-11)	exact
10	exact	1 (-11)	1 (-11)	1 (-11)	exact
11	exact	1 (-11)	1 (-11)	1 (-11)	exact
12	exact	1 (-11)	1 (-11)	exact	exact
exact	1.37802461354	-.17889960288	-.01362867977	.00224821809	-.00008311049

Table 2. Comparison of the absolute errors in the numerical evaluation of

$$\int_0^{2\pi} x \cos x \sin px \, dx \, .$$

p	Exact	Piessens and Poleunis		Present method	
		Absolute error	No. of function evaluations	Absolute error	No. of function evaluations
1	-1.5707963267948966	5(-15)	30	4(-16)	19
2	-4.1887902047863910	9(-15)	30	6(-16)	19
4	-1.6755160819145564	3(-15)	30	1(-15)	20
16	-0.3942390780975427	1(-14)	30	5(-15)	20
64	-0.0981987447275930	3(-15)	30	2(-16)	20
256	-0.0245440671189132	1(-15)	30	2(-16)	19

$$t_j = \cos \frac{\pi j}{n} \qquad j=0\ (1)\ n \tag{4}$$

which were used in equation (4.2.14) degenerate into the Newton-Cotes equally spaced abscissae

$$t_j = \frac{2j}{n} - 1 \tag{5}$$

in the cases n=1 and n=2. This degeneracy does not, of course, occur for $n \geq 3$ and considerable improvement in accuracy is obtained in these cases over the earlier calculations.

The present calculations were repeated using double precision arithmetic for checking purposes, one of the main objects being the removal of the inaccuracies associated with the evaluation of cos p and sin p when p is very large. For instance, subtraction of large multiples of π from the argument may result in the loss of about 4 figures in accuracy when $p=10^4$, when the standard subroutines are employed. The double precision calculations yielded greater accuracy and confirmed the validity of the single precision results. These single precision values are presented here, so that comparison with the earlier calculations may be carried out.

It is also of interest to use quadrature formula (4.2.19) for the test integrals of Piessens and Poleunis. For the purpose of illustration the integral

$$\int_0^{2\pi} x \cos x \sin px \, dx = \begin{cases} -2\pi p\ (p^2-1)^{-1} & (p=2,\ 3,\ 4\ \ldots\ ) \\ -\pi/2 & (p=1) \end{cases} \tag{6}$$

is considered. Numerical results are depicted in Table 2 for p=1, 2, 4, 16, 64 and 256. To facilitate direct comparison, double precision calculations were carried out. The order of the formula used was increased until the errors were less than those obtained by Piessens and Poleunis for 30 function evaluations. This accuracy was achieved

for all values of p from about 19 or 20 function evaluations, thus representing an improvement over the earlier calculations. This is due, presumably, to a decrease in round-off errors generated by the present algorithm compared with the earlier Bessel - function series prescription for the evaluation of the basic integrals. It will also be noticed that the values of ω are $\omega=p\pi$ here, and that, these are large enough for the stability criterion (4.2.28) to be applicable for the values of n used to fit $f(x) = x \cos x$ on $[0, 2\pi]$. The stability of the algorithm was again tested by proceeding to larger values of n and it was possible to go to n=25 even for p=1 and still use the ω^{-1} series.

Numerical tests were also carried out for this integral using the alternative "open" or Gaussian based Chebyshev zeros of expression (4.1.19) as utilized by Piessens and Poleunis. The results are compared with those of the present algorithm which use the Clenshaw - Curtis or "closed" abscissae (4.2.6) and are shown in Table 3 for increasing n. It will be observed that the closed formula is converging more rapidly particularly for large ω, although for large values of n the accuracy obtained by both methods is substantially the same. In this case, it will be noticed that the adaptive nature of the closed formulae could be taken into account here with advantage, to reduce the number of function evaluations required. This is not true for the case of the open formula.

Finally, a much more stringent test of the present algorithm is carried out by considering the test integral of Bakhvalov and Vasil'eva which involves the badly-behaved function $f(x) = \cos \pi u x^2$. The integral is denoted by

$$I(u,q) = \int_{-1}^{1} \cos \pi u x^2 \cos \pi q x \, dx \qquad (7)$$

and has the exact value

Table 3. Comparison of the absolute errors in the numerical evaluation of $\int_0^{2\pi} x \cos x \sin px \, dx$ using formulae (4.1.18) (open) and (4.2.5) (closed).

p	formula	n (order)				
		2	10	15	17	22
1	closed	> 1	6.1(- 7)	2.4(-13)	1.0(-15)	2.2(-18)
	open	> 1	6.1(-7)	3.3(-13)	1.7(-15)	2.2(-18)
2	closed	1.0	3.0(- 5)	4.4(-13)	2.4(-15)	2.0(-19)
	open	1.3	4.3(-5)	7.1(-13)	3.4(-15)	2.0(-19)
16	closed	1.5(- 3)	2.8(- 7)	1.9(-11)	2.9(-13)	1.4(-17)
	open	3.6(- 2)	3.5(- 7)	1.2(-10)	6.5(-13)	2.7(-17)
64	closed	2.4(- 5)	1.2(-7)	2.3(-12)	2.5(-14)	8.0(-19)
	open	8.6(- 3)	1.6(- 6)	5.3(-12)	1.7(-14)	1.4(-18)
256	closed	3.7(- 7)	1.9(- 9)	3.8(-14)	4.0(-16)	1.9(-20)
	open	2.1(- 3)	4.6(- 7)	2.7(-12)	1.1(-14)	3.5(-19)

$$I(u,q) = \frac{1}{\sqrt{2u}} \left\{ \cos A \left[C(B_1) + C(B_2) \right] + \sin A \left[S(B_1) + S(B_2) \right] \right\} \quad (8)$$

where

$$A = \pi q^2/(4u)$$

$$B_1 = (2/u)^{\frac{1}{2}} (u+q/2)$$

and (9)

$$B_2 = (2/u)^{\frac{1}{2}} (u-q/2)$$

and $C(z)$ and $S(z)$ are the Fresnel integrals. (Abramowitz and Stegun, p. 304)

Bakhvalov and Vasil'eva have considered a set of 11 values of q ranging from 5/4 to 451/4, coupled with a set of 14 values of u ranging from 1/4 to 47/4 and have tabulated the relative and absolute errors, quoting the maximum errors obtained over the set $\{q\}$. They have also repeated the exercise using a Newton - Cotes type formula with n=4 with up to 90 sub-divisions for comparison purposes. Here, attention is confined to the extreme values of u and q together with one intermediate value in each case to give a smaller, though representative, set of calculations. Thus, the u and q values are taken to be

$$u = \frac{1}{4}, \frac{23}{4}, \frac{47}{4} \quad (10)$$

and

$$q = \frac{5}{4}, \frac{41}{4}, \frac{451}{4} \quad (11)$$

the corresponding ω values being given by $q\pi$.

The results obtained for various values of n are shown in Table 4 in which the absolute errors are presented. It will be seen that when u=1/4 the function $f(x) = \cos \pi u x^2$ is very well-behaved and hence the order of the formula required is small, reasonable results being obtained even for n=9. Consequently, since the lowest ω value is $5\pi/4$ which is nearly 4, the ω^{-1} series may be used confidently here.

However, in the cases u=23/4 and u=47/4 the function f(x) possesses

Table 4. Absolute errors in the numerical evaluation of

$$\int_{-1}^{1} \cos \pi u x^2 \cos \pi q x \, dx$$

u	q	exact	n		
			9	15	22
$\frac{1}{4}$	$\frac{5}{4}$	-0.25816237030406	2.3(- 8)	1.5(-13)	exact*
	$\frac{41}{4}$	0.02966470953267	1.4(- 7)	5.7(-12)	exact
	$\frac{451}{4}$	0.00283575769375	1.1(- 9)	8.9(-14)	exact
u	q	exact	n		
			34	40	47
$\frac{23}{4}$	$\frac{5}{4}$	-0.38215576878521	1.7(- 8)	4.5(-12)	9.1(-13)
	$\frac{41}{4}$	0.09736925629823	1.6(- 5)	2.0(- 8)	6.0(-12)
	$\frac{451}{4}$	0.00256072719178	8.4(- 8)	1.6(-10)	2.9(-12)
u	q	exact	n		
			34	40	47
$\frac{47}{4}$	$\frac{5}{4}$	0.24111868127101	4.3(- 5)	8.0(- 6)	1.0(- 7)
	$\frac{41}{4}$	0.26746038313496	2.5(- 2)	6.2(- 4)	1.4(- 8)
	$\frac{451}{4}$	0.00233286903630	2.8(- 4)	1.4(- 4)	1.6(- 5)

* indicates accuracy in excess of 16 digits.

12 and 24 zeros respectively on the range $[-1, 1]$. Hence it will be necessary to use a high order formula particularly in the latter case. The ω^{-1} series should be stable here for the values q=41/4 and q=451/4 for which the corresponding values of $[2\omega]$ as required by the stability criterion (4.2.27) are 64 and 708 respectively. This is confirmed by the entries in Table 4 where accurate results are obtainable in most instances although values of n as large as n=47 are required to fit f(x). In the worst case, u=47/4 and q=451/4, convergence is very slow, due mainly to the badly-behaved nature of the function. Some improvement was observed on extending the calculation to order n=55 where the absolute error was found to be 5.5(-7). However, such values of n are extreme both from the point of view of the stability criterion (4.2.34) and also economically, since it is desirable to produce an accurate answer with a minimal number of function evaluations. This particular case was therefore also treated by special techniques as described below.

To return to Table 4, for q=5/4 the value of $[2\omega]$ is only about 8 and, clearly, the ω^{-1} series will be completely unstable long before a large enough n value is attained to fit the function accurately. It follows, therefore, that the alternative ω series (4.2.21) and (4.2.22) must be used here. This is again borne out by the results obtained. The accuracy attained over the u and q ranges is comparable with the results quoted by Bakhvalov and Vasil'eva. Greater accuracy is apparently obtained by the present algorithm in some instances, but it should be pointed out that the present calculation has employed larger orders than the maximum (n=36) used in the earlier work and that double precision arithmetic was necessary in the evaluation of the series (1) and (2) for n>20. In fact it is noted that values of n around 36 are necessary even to begin to fit the function $\cos \pi u x^2$, when u is large.

Note also, that good results are obtainable for various u values for both the large q and the small q values.

As a further illustration of the behaviour of the present algorithm for small values of ω, the limiting case $\omega \to 0$ is treated by taking q=0 in the worst behaved instance of the function f(x), namely u=47/4. This is the case where the cancellation effects in series (4.2.32) are at their worst and should be a stringent practical test of the stability of the algorithm, since very high orders are required to fit f(x). The results are given in Table 5 and show that, even though instabilities of this type are present, the contribution from the smaller values of i are relatively large and produce reasonable values of the integral.

The badly-behaved nature of the function $f(x) = \cos \pi u x^2$ when u is large (say u=47/4) prompted also an investigation into the special techniques suggested in section 4.1. Thus, the range was sub-divided between the complete cycles of $\cos \pi u x^2$, starting at x=0 and integrating over each cycle separately. It was hoped to reduce the cancellation effects arising on sub-division by this device. Thus, integration is carried out between the points $x=(2m/u)^{\frac{1}{2}}$ where m=0, 1, 2, ... m_o and, finally, between $(2m_o/u)^{\frac{1}{2}}$ and x=1. The maximum value of m_o is given by $[u/2]$. The results are shown in Table 6 and demonstrate high accuracy for an economical number of function evaluations in this, the most badly-behaved case of f(x). The number of sub-divisions used here is 5+1=6 and the order of formula employed in each cycle varied from n=13 to n=24. Accurate single precision results are therefore obtainable by this method for an economical number (around 100) function evaluations using maximum order of around 20.

Table 5. Absolute errors in the numerical evaluation of

$$\int_{-1}^{1} \cos \pi u x^2 \, dx \, .$$

u	exact	n		
		34	40	47
$\frac{47}{4}$	0.186880300	6.6(- 5)	1.2(- 5)	4.4(- 7)

Table 6. Absolute errors in the numerical evaluation of

$$\int_{-1}^{1} \cos \pi u x^2 \cos \pi q x \, dx$$ by integrating over separate cycles of $\cos \pi u x^2$.

u	q	no. of function evaluations			
		79	97	121	145
$\frac{47}{4}$	$\frac{5}{4}$	2.1(- 9)	3.4(-11)	5.6(-14)	1.8(-16)
	$\frac{41}{4}$	7.9(- 9)	1.1(-10)	5.9(-14)	8.2(-16)
	$\frac{451}{4}$	9.6(- 8)	1.3(- 9)	4.9(-12)	2.9(-14)

Appendix

An Algol 68 version of the algorithm is presented. Although this program does not use any advanced features of Algol 68 (and so could have been programmed in other languages such as Fortran or Algol 60), the language chosen enables the algorithm to be presented in a neat and efficient form.

```
'PROC' QUADRULE=('PROC'('REAL')'REAL'F,'REAL'A,B,P,'INT'N,
                 'BOOL'TYPE)'REAL':
'BEGIN'
'C' THIS PROCEDURE EVALUATES THE INTEGRAL OF F(X) SIN OR COS P*X
ON [A,B] USING N-TH ORDER QUADRATURE FORMULA OF CLENSHAW-CURTIS
TYPE, NOTATION AS IN THE TEXT, THE BOOLEAN TYPE IS TRUE IF
SIN PX IS THE WEIGHT FUNCTION AND FALSE IF COS PX
IS THE WEIGHT. 'C'

   [0:N]'REF'[]'REAL'D; [0:N]'REAL'CAPF,TCWT,TSWT;
   'REAL'INTEGRAL,S1,S2,W1,WB,CS1,CS2,AI,CAPK,PART1,
         PART2,OMEGA,X1,X2,SINE,COSE;
   'INT'R1,L,K,UPB,N2,J1,NI1;
   X1←(B+A)/2.0; X2←(B-A)/2.0; CAPK←P*X1; OMEGA←P*X2;
   SINE←SIN(OMEGA); COSE←COS(OMEGA);'BOOL'ODD;

   'PROC'INTEG=('INT'N,'REAL'OMEGA,'REF''REAL'SI)'REAL':
   'BEGIN'
   'C' THIS IS A PROCEDURE TO EVALUATE THE INTEGRAL OF
   X↑N SIN OR COS OMEGA*X ON [-1,1] USING EQUATIONS (4.2.1) AND
   (4.2.2). THE INTEGRAL FOR COSINE WEIGHT FUNCTION IS
   DELIVERED WHILE THE INTEGRAL FOR SINE WEIGHT FUNCTION IS
   ASSIGNED TO SI. 'C'
     'REAL'S1,S2,P1,P2,W1,W2,T1,T2,P3;P1←P2←1.0;S1←S2←0.0;P3←1.0;
     'INT'IB,NN; [0:N]'REAL'Y;
     'BOOL'SW←'TRUE',SV←'ODD'N;
     'REAL' EPS←1.0&-60;
      'IF'N=0'THEN'SI←0.0; 2.0*SINE/OMEGA
            'ELSE'
               P3←1.0/OMEGA; Y[0]←P3;
               'FOR'I1'TO'N'WHILE'P3>EPS'DO'
               (P3'TIMES'(N-I1+1)/OMEGA;Y[I1]←P3;NN←I1);
               IB←NN+1-(NN'/'4)*4; W1←SINE; W2←COSE; T1←P3;
               ('ODD'(N-NN)|SW←'FALSE'; T2←-T1|SW←'TRUE';T2←T1);
                     'FOR'I1'FROM'NN'BY'-1'TO'0'DO'
                        'BEGIN'
                           (I1#NN|T1←Y[I1];
                                 (SW|T2←-T1;SW←'FALSE'
                                 |T2←T1 ; SW←'TRUE'));
```

```
                    'IF'SV'THEN'
                        'CASE'IB'IN'(P1←W2;P2←W2;IB←4),
                                    (P1←-W1;P2←W1;IB←1),
                                    (P1←-W2;P2←-W2;IB←2),
                                    (P1←W1;P2←-W1;IB←3)
                        'ESAC'
                          'ELSE'
                        'CASE'IB'IN'
                                    (P1←W1;P2←-W1;IB←4),
                                    (P1←W2;P2←W2; IB←1),
                                    (P1←-W1;P2←W1;IB←2),
                                    (P1←-W2;P2←-W2;IB←3)
                          'ESAC'
                      'FI';
                      S1'PLUS'T1*P1;S2'PLUS'T2*P2
                  'END';

                  SI←(SV|S2-S1|0.0);
                   (SV|0.0|S1-S2)
   'FI'
'END';


 'PROC'INTEG2=('INT'N,'REAL'OMEGA,'REF''REAL'SI)'REAL':
 'BEGIN'
 'C' THE VALUE OF THE INTEGRAL (4.2.21) IS DELIVERED AND
 THAT OF (4.2.22) IS ASSIGNED TO SI . 'C'
    'REAL'S, T1, SK,T1,T2,W2←OMEGA*OMEGA;
 'INT'M←18; 'BOOL'BOOL←'ODD'N;
    SI←(BOOL|T1←OMEGA/(N+2); S←T1;
      'FOR'I'TO'M'WHILE'('ABS'(T1/S)>1.0&-22)'DO'
    'BEGIN'
       T1'TIMES'-W2*(2*I+N)/(2*I+N+2)/(2*I)/(2*I+1);S'PLUS'T1
    'END';
      S'TIMES'2.0|0.0);
    SK←(BOOL|0.0|T2←1.0/(N+1);S←T2;
    'FOR'I'TO'M'WHILE'('ABS'(T2/S)>1.0&-22)'DO'
    'BEGIN'
      T2'TIMES'-W2*(2*I+N-1)/(2*I+N+1)/(2*I)/(2*I-1);S'PLUS'T2
    'END';
    S'TIMES'2.0);
    SK
 'END';


'PROC'CHEBCOEF=('INT'R)'REF'[ ] 'REAL':
'BEGIN'
'C' THIS IS A PROCEDURE TO CALCULATE THE NON-ZERO COEFFICIENTS
OF THE R-TH ORDER CHEBYSHEV POLYNOMIAL USING THE
RECURRENCE RELATIONSHIP (4.2.4) . 'C'
  'INT'R1←(R+2)'/'2;
  [1:R1]'REAL'DB,DD;
   DD[1]←DB[1]←1;
```

```
    'IF'R>1'THEN'
        'FOR'I'FROM'2'TO'R1'DO'
        'FOR'J'FROM'I'BY'-1'TO'1'DO'
        'BEGIN'
          DD[J]←(J=I|2*DB[J-1]|;J=1|-DD[1]
                                |2*DB[J-1]-DD[J]);
          DB[J]←(J=I|2*DD[I]|2*DD[J]-DB[J])
        'END';
    (R#(R'/'2)*2|'FOR'I'TO'R1'DO'DD[I]←DB[I])
    'FI'; DD
'END';
```

```
'PROC'CAPI=([ ]'REAL'CAPF)'REAL';
'BEGIN'
  'INT'N2←N'/'2; [0:N]'REAL'CJI;
  INTEGRAL←0.0;
  'C' THE INTEGRAL DEFINED BY (4.2.16) AND (4.2.19) 'C'
  PART1←PART2←0.0;
  'FOR'I'FROM'0'TO'N'DO'
    'BEGIN'
      S1←S2←0.0; R1←(I+2)'/'2; ODD←'ODD'I; AI←0.0;
      'C' S1 AND S2 REPRESENT THE WEIGHTED INTEGRALS OF THE
      CHEBYSHEV POLYNOMIALS DEFINED BY (4.3.1),  (4.3.2)
      FOR USE IN FORMULAE (4.2.16) AND (4.2.19) . 'C'
      'FOR' R 'TO' R1 'DO'
        'BEGIN'
          L←(ODD|2*R-1|2*(R-1)); W1←D[I][R];
          S1'PLUS'W1*TCWT[L]; S2'PLUS'W1*TSWT[L]
        'END';
      CJI[0]←0.5;
      'FOR'J'TO'N2'DO' CJI[J]←(I=0|1.0|COS(PI*J*I/N));

      'C' FOR USE OF THE OPEN FORMULA THE PRECEDING TWO
       LINES ARE REPLACED BY
      'FOR' J'FROM'0'TO'N2'DO'
      CJI[J]←(I=0|1.0|COS(PI*I*(2.0*J+1.0)/(2.0*N+2.0))); 'C'

      'FOR'J'FROM'0'TO'N2'DO' AI'PLUS'CAPF[J]*CJI[J];
      (I=(I'/'2)*2| 'FOR'J'FROM'N2+1'TO'N'DO'
                    AI 'PLUS'CAPF[J]*CJI[N-J]
                  | 'FOR'J'FROM'N2+1'TO'N'DO'
                    AI'PLUS'CAPF[J]*(-CJI[N-J]));
      AI 'DIV'(I=0'OR'I=N|N!N/2.0);

      'C' FOR USE OF THE OPEN FORMULA (4.1.21) THE PRECEDING
      LINE IS REPLACED BY
      AI'DIV'(I=0|N+1|(N+1)/2.0); 'C'

      PART1'PLUS'AI*S1; PART2'PLUS'AI*S2
    'END';
```

```
      INTEGRAL 'PLUS' 'IF' TYPE 'THEN'
                      'C' THE WEIGHT FUNCTION IS SIN PX 'C'
                      PART1*CS2+PART2*CS1
                      'ELSE'
                      'C' THE WEIGHT FUNCTION IS COS PX 'C'
                      PART1*CS1-PART2*CS2
                      'FI';
      INTEGRAL 'TIMES'X2
   'END';

   N2←N'/'2; [0:N2]'REAL'XS;
   K←0;
   L2: UPB←(K+2)'/'2;
   D[K]←'LOC'[1:UPB]'REAL';
   D[K][1:UPB]←CHEBCOEF(K);
   ((K'PLUS'1)<=N! 'GOTO' L2);
   'C' THE PRECEDING PART GENERATES A TRIANGULAR ARRAY
   TO RETAIN ALL THE NON-ZERO COEFFICIENTS OF CHEBYSHEV
   POLYNOMIALS OF ORDER 0,1,2,...N . 'C'

   'C' XS IS (B-A)T/2 AS IN (4.2.13), ONLY HALF OF THE
   NUMBER OF XS ARE CALCULATED .'C'
   XS[0]←X2; 'FOR'S'TO'N2'DO' XS[S]←X2*COS(PI*S/N);

   'C' FOR USE OF THE OPEN FORMULA (4.1.21) THE PRECEDING LINE
   IS REPLACED BY
   'FOR'S'FROM'0'TO'N2'DO'
   XS[S]←XI*COS(PI*(2.0*S+1.0)/(2.0*N+2.0)); 'C'

   (OMEGA>4.0!
   'FOR'I'FROM'0'TO'N'DO'(TCWT[I]←INTEG(I,OMEGA,WB); TSWT[I]←WB)
   !'FOR'I'FROM'0'TO'N'DO'(TCWT[I]←INTEG2(I,OMEGA,WB); TSWT[I]←WB) );
   'C' TCWT AND TSWT STORE THE VALUES OF INTEGRAL OF T↑I COS OR
   SIN OMEGA*T ON [-1,1] .  FOR OMEGA>4 PROCEDURE INTEG,
   OMEGA<4 INTEG2 IS CALLED . 'C'

   CS1←COS(CAPK); CS2←SIN(CAPK);
   'FOR'J'FROM'0'TO'N2'DO' CAPF[J]←F(X1+XS[J]);
   'FOR'J'FROM'N2+1'TO'N'DO' CAPF[J]←F(X1-XS[N-J]);

   INTEGRAL←CAPI(CAPF);
   INTEGRAL
'END';
```

CHAPTER 5

THE EVALUATION OF OSCILLATORY INTEGRALS WITH INFINITE LIMITS

A numerical method for the evaluation of highly oscillatory integrals with semi-infinite ranges is considered. A sequence of integrals is formed by sub-division of the range and the convergence of this sequence is accelerated by using the non-linear transformation proposed by Shanks (1955). The method is applied to a number of integrals including those convergent in the mean only. The ability of Shanks' transformation to accelerate and even to induce convergence is demonstrated. Numerical comparisons are made by using the well-known transformation due to Euler and examples are given to indicate the efficiency of Shanks' technique.

5.1 Introduction

In the previous chapters integrals over a finite range have been studied. It is now proposed to present a practical method of evaluating integrals of the type

$$\int_0^\infty f(x) \begin{matrix}\sin \\ \cos\end{matrix} px \, dx \qquad (1)$$

In certain circumstances, the constant p may take large values and, as with the finite range integrals discussed in the earlier chapters, considerable difficulty is experienced in computing these integrals by conventional methods, owing to the extremely strong cancellation of the positive and negative contributions from the rapidly oscillatory integrand. The methods due to Newton-Cotes, Euler-Maclaurin and Gauss treat the entire integrand and hence require a large number of points when p is large. The quadrature rules discussed in Chapters 3 and 4, such as the Filon and the Clenshaw and Curtis rules, have the advantage that they apply an interpolation formula to f(x) only. However, as is well-known, all these formulae apply to the finite range. In principle, it is possible to deal with (1), by applying the conventional quadrature formulae over a finite interval (0, N) such that the remaining part of the integral referring to the interval (N, ∞) can be calculated by an asymptotic expansion. (Stiefel, 1961). In practice, however, there are the usual difficulties associated with this method.

An alternative approach introduced by Hurwitz and Zweifel (1956) and further developed by Hurwitz, Pfeifer and Zweifel (1959) is to subdivide the range and to integrate between the successive zeros of the trigonometric function thus converting the infinite integral to a

summation. With the respective changes of the variable

$$x = (\pi/p)(y + \tfrac{1}{2}) \quad , \qquad x = (\pi/p)y \tag{2}$$

the integrals I_s and I_c are written as

$$I_s = \int_0^\infty f(x) \sin px \, dx = \frac{\pi}{2p} \int_{-\infty}^{\infty} f\left(\frac{\pi}{p} \left[y + \frac{1}{2} \right] \right) \cos \pi y \, dy \tag{3}$$

$$I_c = \int_0^\infty f(x) \cos px \, dx = \frac{\pi}{2p} \int_{-\infty}^{\infty} f\left(\frac{\pi}{p} y \right) \cos \pi y \, dy \tag{4}$$

Using the transformation

$$\int_{-\infty}^{\infty} F(y) \, dy = \int_{-1/2}^{1/2} \sum_{n=-\infty}^{\infty} F(y + n) \, dy \tag{5}$$

(3) and (4) take the forms

$$I_s = \frac{\pi}{2p} \int_{-1/2}^{1/2} \sigma(y, p) \cos \pi y \, dy \tag{6}$$

$$I_c = \frac{\pi}{2p} \int_{-1/2}^{1/2} \gamma(y,p) \cos \pi y \, dy \tag{7}$$

where

$$\sigma(y,p) = \sum_{n=-\infty}^{\infty} (-1)^n f\left(\frac{\pi}{p} \left[y + n + \frac{1}{2} \right] \right) \tag{8}$$

$$\gamma(y,p) = \sum_{n=-\infty}^{\infty} (-1)^n f\left(\frac{\pi}{p} \left[y + n \right] \right) \tag{9}$$

In these papers a variation of Gaussian quadrature is suggested to evaluate the integrals (6) and (7). Later it was shown by Saenger (1964) that this method is nothing more than the use of an infinite

trapezoidal sum to approximate the integral. The main objection to the method of Hurwitz and Zweifel is that the resulting series may converge slowly. An attempt to remedy such a defect was made by Longman (1960), who employed a variation of Euler's transformation to accelerate convergence. Further, Balbine and Franklin (1966) showed that the Euler transformation approximates the Fourier integral by infinite series, and gave a detailed explanation of this approach. Similar methods have been proposed recently by Piessens and Haegemanns (1973) and by Squire (1973).

The present method deals with the possibility of using the more general non-linear transformation of Shanks.

5.2 Shanks' Non-Linear Transformation

Shanks regards a given sequence $\{A_n\}$ $n = 0,1,2,\ldots$ as a function of n, evaluated for integer values of n, and approximates this function by a system of equations by likening it to a "mathematical transient of order j"

$$A_n = B + \sum_{i=1}^{j} a_i (q_i)^n \qquad (q_i \neq 0,1) \qquad (1)$$

and obtains information about the behaviour of the sequence as $n \to \infty$. This involves the use of the operators e_j to transform $\{A_n\}$ into another sequence $\{B_{j,n}\}$ according to

$$\{B_{j,n}\} = e_j \{A_n\} \qquad (n \geq j) \qquad (2)$$

Shanks obtains the general term of the transformed sequence, by solving the system of equations mentioned above, in the form of the

ratio of two determinants of order (j + 1), namely

$$B_{j,n} = \frac{\begin{vmatrix} A_{n-j} & A_{n-j+1} & \cdots & A_{n-1} & A_n \\ \Delta A_{n-j} & \Delta A_{n-j+1} & \cdots & \Delta A_{n-1} & \Delta A_n \\ \Delta A_{n-j+1} & \Delta A_{n-j+2} & \cdots & \Delta A_n & \Delta A_{n+1} \\ \vdots & \vdots & & \vdots & \vdots \\ \Delta A_{n-1} & \Delta A_n & & \Delta A_{n+j-2} & \Delta A_{n+j-1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 & 1 \\ \Delta A_{n-j} & \Delta A_{n-j+1} & \cdots & \Delta A_{n-1} & \Delta A_n \\ \Delta A_{n-j+1} & \Delta A_{n-j+2} & \cdots & \Delta A_n & \Delta A_{n+1} \\ \vdots & \vdots & & \vdots & \vdots \\ \Delta A_{n-1} & \Delta A_n & & \Delta A_{n+j-2} & \Delta A_{n+j-1} \end{vmatrix}} \qquad (3)$$

where

$$\Delta A_n = A_{n+1} - A_n \qquad (4)$$

Details of restrictions imposed and conditions to be satisfied in the use of these operators are presented in Shanks' paper. In very general terms, if the convergence of the parent sequence is approaching the geometric state, the transformations are extremely effective. The theoretical basis of the Shanks' paper has been discussed recently in some considerable detail by Levin (1973) who also proposed some possible generalizations.

The most frequently used transformation, e_1, produces the particularly simple result

$$B_{1,n} = (A_{n+1} A_{n-1} - A_n^2)(A_{n+1} + A_{n-1} - 2A_n)^{-1} \tag{5}$$

which is, of course, the well-known Aitken's δ^2 extrapolation procedure (Hildebrand (1956)). The transformation e_j may also be used iteratively to produce a triangular array of sequences

$$\begin{array}{llll} A_0 & & & \\ A_1 & B_{1,n} & & \\ A_2 & B_{2,n} & C_{1,n} & \\ A_3 & B_{3,n} & C_{2,n} & D_{1,n} \\ \vdots & \vdots & \vdots & \vdots \end{array} \tag{6}$$

with the following relations

$$\begin{aligned} \{B_{j,n}\} &= e_j \{A_n\} && n \geq j \\ \{C_{j,n}\} &= e_j \{B_{j,n}\} = e_j^2 \{A_n\} && n \geq 2j \\ \{D_{j,n}\} &= e_j \{C_{j,n}\} = e_j^3 \{A_n\} && n \geq 3j \end{aligned} \tag{7}$$

A practical algorithm for the repeated application of this transformation was suggested by Wynn (1956), who defined the sequence e_j by the non-linear recurrence relation

$$e_{j+1}\{A_n\} = e_{j-1}\{A_n\} + \left[e_j\{A_{n+1}\} - e_j\{A_n\}\right]^{-1} \qquad (j = 1,2,\ldots) \tag{8}$$

with

$$e_o\{A_n\} = \{A_n\} \tag{9}$$

The $e_{2j}\{A_n\}$ are equivalent to the results of applying the j-th Shanks' transformation to the sequence $\{A_n\}$ and yield a particular form of the Padé method, which has been applied to numerical integration by Chisholm, Genz and Rowlands (1972).

5.3 Applications and Results

The interval of integration is divided in accordance with the half-cycles of the integrand into the sub-intervals $[a_n, a_{n+1}]$, $n = 0,1,2,\ldots$, where in the case of the integrals (5.1.1) with weight function $\sin px$, a_n is given by

$$a_n = n\pi/p \tag{1}$$

The rather more general oscillatory integrals

$$\int_o^\infty f(x) \begin{matrix}\sin \\ \cos\end{matrix} px^2 \, dx \tag{2}$$

which arise in many applications are also treated here. For such integrals a_n is given by

$$a_n = (n\pi/p)^{1/2} \tag{3}$$

in the case of the $\sin px^2$ weight function. Initially, a low order Gauss-Legendre quadrature formula (Davis and Rabinowitz (1967)) was employed to carry out the integrations over each half-cycle $[a_n, a_{n+1}]$ according to the prescription

$$\int_a^b g(x)\,dx = \frac{1}{2}(b-a)\sum_{i=1}^{r} w_i\, g\left[\frac{1}{2}(b+a) + \frac{1}{2}(b-a)\,x_i\right] + \varepsilon_r \qquad (4)$$

This result represents the basic r-point Gauss-Legendre quadrature formula, ε_r being the associated error, and the weights w_i and the abscissae x_i are extensively tabulated by Stroud and Secrest (1966). To minimize the contributions from truncation and round-off errors the intervals $[a_n, a_{n+1}]$ were again sub-divided uniformly and formula (4) was applied successively in each of the sub-intervals to yield the values

$$T_n = \int_{a_n}^{a_{n+1}} g(x)\,dx \qquad (5)$$

where g(x) represents the appropriate integrand from integrals (2) or (5.1.1).

The Tables presented in this Chapter are therefore based on the evaluation of the half-cycle contributions T_n using a low order Gauss-Legendre formula. Since the completion of this work a powerful quadrature formula was developed as described in Chapter 4, and considerable economy in the number of function evaluations is realized over the earlier calculations. However, the primary object of this Chapter is to assess Shanks' transformation as a method to accelerate or induce convergence of the sequence T_n and so the earlier Tables are retained. The most widely used combination in practice is to use six 2-point rules in the interval $[a_n, a_{n+1}]$, although on some occasions twelve 2-point rules are employed for greater accuracy. The actual integrals required are given by

$$A = \sum_{i=0}^{\infty} T_i \qquad (6)$$

and the terms in the sequence $\{A_n\}$ are specified by

$$A_n = \sum_{i=0}^{n} T_i \tag{7}$$

In cases in which the integral has weight function cos px, the first term of the sequence $\{T_n\}$ needs to be evaluated separately while the half-cycle contributions are due to the sub-intervals $[a_n, a_{n+1}]$, $n = 1,2,\ldots$ with

$$a_n = n\pi/2p \tag{8}$$

This would also be the case if it were desired to extend the method to integrals over the interval $(-\infty, \infty)$, which would be treated using the present techniques on the intervals $[0, \infty)$ and $[0, -\infty)$. Again the isolation of T_0 would be necessary to avoid, for example, a cusp at $x = 0$ of the form $\exp(-|x|)$. This point is also discussed by Balbine and Franklin.

The first application concerns the evaluation of the integral

$$I_1(\alpha, p) = \int_0^{\infty} e^{-\alpha x} \sin px \, dx = p(\alpha^2 + p^2)^{-1} \tag{9}$$

In this case T_n is given by

$$T_n = (-1)^n \, p(\alpha^2+p^2)^{-1} \exp(-\gamma n) \left[1 + \exp(-\gamma)\right] \tag{10}$$

where

$$\gamma = \alpha\pi/p \tag{11}$$

The terms of the sequence of partial sums, $\{A_n\}$, are easily evaluated analytically, yielding

$$A_n = p(\alpha^2+p^2)^{-1} \left\{1 - (-1)^{n+1} \exp\left[-(n+1)\alpha\pi/p\right]\right\} \tag{12}$$

The geometric convergence of this sequence, as indicated by the relation,

$$\Delta A_{n+1} \; / \; \Delta A_n \;\; = \;\; - \exp(-\gamma) \tag{13}$$

implies <u>immediate</u> convergence of the sequence $e_1\{A_n\}$ to the exact limiting value, that is

$$B_{1,n} \;\; = \;\; p(\alpha^2+p^2)^{-1} \tag{14}$$

for <u>all</u> $n \geq 1$.

On the other hand, the well-known Euler transformation sometimes makes the series converge faster and sometimes it does not. The Euler sum is

$$E_n = \frac{1}{2}\, T_o - \frac{1}{4}\, \Delta T_o + \frac{1}{8}\, \Delta^2 T_o - \ldots + \frac{(-1)^n}{2^{n+1}}\, \Delta^n T_o \tag{15}$$

$$= T_o \left\{ 1 - \left[1 - \exp(-\gamma)/2 \right]^{n+1} \right\} \left[1 + \exp(-\gamma) \right]^{-1}$$

$$= p(\alpha^2+p^2)^{-1} \left\{ 1 - \left(\left[1 - \exp(-\gamma) \right]/2 \right)^{n+1} \right\} \tag{16}$$

and the remainder after n terms of the Euler series takes the form

$$\varepsilon_E \;\; = \;\; p(\alpha^2+p^2)^{-1} \left(\left[1 - \exp(-\gamma) \right]/2 \right)^{n+1} \tag{17}$$

A comparison of (16) and the truncation error associated with the approximate sum of the series $\{A_n\}$, namely

$$\varepsilon_A \;\; = \;\; p(\alpha^2+p^2)^{-1} \left\{ (-1)^{n+1}\, e^{-(n+1)\gamma} \right\} \tag{18}$$

indicates that the Euler transformation produces a sequence which

converges less quickly than even the original sequence $\{A_n\}$ for values of α which are greater than the critical value $(p \ln 3)/\pi$. The Euler transformation does in fact produce accelerated convergence for α less than the critical value, but is, of course, always inferior to the immediately converging Shanks' transformation. The implication of this result is that the Shanks' transformation is extremely powerful in dealing with integrals where T_n exhibits predominantly exponential decay.

As a practical example of this class of integrals consideration is given to

$$I_2 = \int_0^\infty x^{-1} \exp(-x/2) \sin x \, dx = \tan^{-1} 2 = 1.107149... \qquad (19)$$

The transformation e_1 was applied iteratively to the sequences $\{A_n\}$ to give the sequences $e_1\{A_n\}, e_1^2\{A_n\}, \ldots$ the first few members of which are shown in Table 1. For comparison purposes, Euler's transformation was applied to the terms T_n to produce the sequence of partial sums $\{E_n\}$. The half-cycle contributions, T_n, were evaluated over $[n\pi,(n+1)\pi]$ using twelve 2-point Gauss-Legendre formulae. It will be noticed that the Euler sequence $\{E_n\}$ converges less well than the original sequence $\{A_n\}$. In contrast $e_1\{A_n\}$ converges extremely rapidly and $e_1^2\{A_n\}$ converges immediately to the limiting value. Moreover, the transformation e_2 could also be applied to $\{A_n\}$ and again yields the limiting value immediately, namely

$$B_{2,2} = 1.107149... \qquad (20)$$

This is important in a wide number of applications in molecular quantum mechanics where integrands exhibit such behaviour.

Table 1. Successive Sequences for evaluating $I_2 = \int_0^\infty x^{-1} \exp(-x/2) \sin x \, dx$

n	T_n	A_n	E_n	$e_1\{A_n\}$	$e_1^2\{A_n\}$
0	1.148148	1.148148	0.574074		
1	-0.045820	1.102328	0.849656	1.107254	
2	0.005519	1.107847	0.982409	1.107141	1.107149
3	-0.000809	1.107038	1.046562	1.107150	
4	0.000130	1.107168	1.077652		

A further example involving a different family of integrals is

$$I_3 = \int_{\pi}^{\infty} x^{-2} \sin x \, dx = - \, ci(\pi) = - \, 0.073668... \qquad (21)$$

where ci is the cosine integral as defined by Gradshteyn and Ryzhik (1965). The results are demonstrated in Table 2. The half-cycle contributions, T_n, were evaluated over $[(n+1)\pi, (n+2)\pi]$ using six 2-point Gauss-Legendre formulae. It will be observed once more that the Shanks' transformations give rise to sequences which converge more rapidly than that produced by the Euler method.

As another example, the integral

$$I_4 = \int_{o}^{\infty} x^2 \sin(100 \, x^2) dx \qquad (22)$$

which converges in the mean only (in the Abel sense) is considered. The exact result is

$$I_4 = (\pi/2)^{1/2} \, / \, 4000 = 3.133285... \times 10^{-4} \qquad (23)$$

which is readily obtained by standard integration, using the integrating factor $\exp(-\beta x^2)$ as β tends to zero. The numerical results are shown in Table 3. The half-cycle contributions, T_n, were evaluated with $a_n = (n\pi/100)^{1/2}$ using twelve 2-point Gauss-Legendre formulae. It is noticed in this case that the original sequence $\{A_n\}$ is divergent. The Euler transformation produces the sequence $\{E_n\}$ which is slowly convergent. The sequence $e_1\{A_n\}$ is also seen to be slowly convergent, but when the operator e_1 is used iteratively, the successive sequences $e_1^2\{A_n\}$, $e_1^3\{A_n\}$... are seen to be converging rapidly. The application of the operator e_2 to the original sequence $\{A_n\}$ produces the sequence $\{ 3.1268, 3.1354, 3.1322, 3.1337, 3.1329,... \times 10^{-4}\}$ and e_2^2 produces 3.1332×10^{-4} as its first term.

Table 2. Successive Sequencies for evaluating $I_3 = \int_{\pi}^{\infty} x^{-2} \sin x \, dx$

n	T_n	A_n	E_n	$e_1\{A_n\}$	$e_1^2\{A_n\}$	$e_1^3\{A_n\}$
0	-9.6230	-9.6230	-4.8115			
1	3.3180	-6.3049	-6.3877	-7.3496		
2	-1.6737	-7.9786	-7.5529	-7.3744	-7.3677	
3	1.0078	-6.9708	-7.4305	-7.3633	-7.3667	-7.3669
4	-0.6730	-7.6439	-7.3901	-7.3689	-7.3670	
5	0.4812	-7.1626	-7.3614	-7.3657		
6	-0.3612	-7.5238	-7.3651			

The entries have a multiplying factor of 10^{-2}.

Table 3. Successive Sequencies for evaluating $I_4 = \int_0^\infty x^2 \sin(100\, x^2)dx$

n	T_n	A_n	E_n	$e_1\{A_n\}$	$e_1^2\{A_n\}$	$e_1^3\{A_n\}$	$e_1^4\{A_n\}$
0	1.2177	1.2177	6.0883				
1	-2.1650	-0.9474	3.7199	2.7356			
2	2.7998	1.8525	3.3292	3.3475	3.1236		
3	-3.3143	-1.4619	3.2090	2.9944	3.1366	3.1330	
4	3.7588	2.2970	3.1647	3.2324	3.1317	3.1332	3.1332
5	-4.1560	-1.8590	3.1468	3.0577	3.1340	3.1332	
6	4.5182	2.6593	3.1392	3.1931	3.1327		
7	-4.8535	-2.1943	3.1359	3.0842			
8	5.1671	2.9728	3.1344				
	$\times 10^{-3}$	$\times 10^{-3}$	$\times 10^{-4}$	$\times 10^{-4}$	$\times 10^{-4}$	$\times 10^{-4}$	$\times 10^{-4}$

The final application concerns a practical problem arising in the study of particle interaction in a slow viscous flow (Evans, (1973)). The force on one particle due to another requires the evaluation of the integral

$$I_5 = \int_o^\infty x \exp(-y \sin\alpha) c^{-1} y^{-1} \left[2(c+x^2) J_o(\ell x) - 2x J_1(\ell x)/\ell \right] dx \quad (24)$$

where

$$y = (x^2 + R^2/4)^{1/2} \quad (25)$$

$$c = Ry + R^2/2 \quad (26)$$

with

$$\ell = \cos\alpha , \quad \sin\alpha > 0 \quad (27)$$

and α being an angle relating the particle position to the flow direction and R the Reynold's number. At $\alpha = 0$ this integral exists in the mean only. For small R the integral is asymptotically

$$2R^{-1} \left[\ell - (1-\ell)/\cos\alpha \right] + O(R) \quad (28)$$

and this agrees with numerical values obtained for the complete integral using Shanks' technique with the operator e_2 near and at $\alpha = 0$. The sequences were generated by integrating between the zeros of the respective Bessel functions J_o and J_1. Agreement also occurs between Shanks' technique and the conventional integration method for α around 10^o though the conventional techniques are more difficult to apply for smaller α.

In conclusion, it appears that the Shanks' acceleration technique is a powerful tool for the evaluation of oscillatory integrals with an

infinite range. This is particularly true when the oscillations are damped in a predominantly exponential manner or fall off, for example, as $1/x^k$ where k is sufficiently large (say $k \geq 1$). In these cases Shanks' technique proves more economical in terms of function evaluations than the well-known Euler transformation. However, when the higher order difference terms in the Euler formula are small (for example, when the convergence is very slow or when polynomial behaviour is exhibited) the Euler method converges extremely rapidly. Also for integrals which converge in the mean only Shanks' method proves successful, and may again converge more rapidly than Euler's transformation, depending on the behaviour of the original sequence.

CHAPTER 6

GENERATION OF GAUSSIAN QUADRATURE RULES

A general investigation into the structure of Gaussian quadrature formulae is carried out. The notorious instability associated with the algebraic approach to the generation of Gaussian quadrature coefficients for certain weight functions is discussed. A non-linear approach in which the advent of instability is postponed to formulae of higher order is introduced.

In the case of weight functions whose associated monomials are expressible as rational numbers (apart from multiplying constants), an accurate algorithm is developed by extending the standard prelude of the Algol 68 language. The algorithm is based on the use of multilength rational arithmetic to solve the resulting system of linear equations. Applications and numerical results are included and Algol 68 programs are appended.

6.1 Introduction and Preliminaries

In the preceding chapters various methods have been developed for dealing with the evaluation of integrals containing oscillatory weight functions. It is suggested that the Chebyshev based methods of Chapter 4 are to be preferred in practice to the Newton-Cotes based prescriptions of Chapters 2 and 3. It will also be observed that Gaussian based quadrature methods have been used in the acceleration algorithms of Chapter 5. However at that stage attention was confined to low order quadrature procedures which were applied to the complete integrand, whereas, as pointed out in Section 1.1., it would probably be preferable to consider the "factored form", and to incorporate the trigonometric factor as a weight function.

It is therefore of interest to consider the development of Gaussian formulae with trigonometric weight functions and in the present chapter a general investigation is instigated into the structure of such formulae for general weight functions.

The method of undetermined coefficients provides a basis for generating a sequence of Gaussian quadrature rules of the form:

$$\int_a^b w(x)\ f(x)\ dx \simeq \sum_{k=1}^{n} \alpha_k\ f(x_k) \qquad n = 1,2,3,.... \tag{1}$$

where α_k and x_k are regarded as parameters. The weight function $w(x)$ is nonnegative and such that all its moments

$$g_j = \int_a^b x^j\ w(x)\ dx \qquad j = 0,1,2,... \tag{2}$$

exist. The $2n$ free parameters are determined by the same number of defining equations by making eq. (1) an exact equality for the sequence

of functions $f(x) = 1, x, x^2, \ldots x^{2n-1}$, resulting in

$$g_j = \sum_{k=1}^{n} \alpha_k x_k^j \qquad j = 0,1,2,\ldots(2n-1) \tag{3}$$

The problem of producing numerical tables for Gaussian quadrature basically involves the solution of the above algebraic system of equations. This problem had received considerable attention after the emergence of high speed computers and several algorithms were given by different authors. The majority of these methods consist of constructing the system of orthogonal polynomials associated with the weight function and obtaining zeros of these orthogonal polynomials. However, for higher order formulae severe numerical instability occured and numerous possible ways were tried to improve the solution. For example, Anderson (1965) found it necessary to use double precision arithmetic but even so obtained only 4 figure accuracy for the 10-point quadrature rule with weight function $\log_e x$ and interval $[0, 1]$. Alternatively, Rutishauser (1962) suggested the use of the quotient-difference algorithm, whilst Golub and Welsch (1969) employed Francis' QR-transformation to compute x_k as the eigenvalues of a Jacobi matrix and α_k as the corresponding eigenvectors. In the recent years, Gautschi (1968), and Sack and Donovan (1972) have diverted attention from the use of ordinary moments as a starting point. The latter argued that with increasing j the powers of x^j tend to reach a strong maximum near one or both ends of the range and the moments (2) do not give a good description of w(x) over the whole interval. Their method consisted of the use of orthogonal polynomials $\Pi_j(x)$, satisfying the usual recurrence relation, such that the modified moments

$$m_j = \int_a^b \Pi_j(x)\, w(x)\, dx \tag{4}$$

have the values of the same order of magnitude throughout the interval. Unfortunately the modified moments are not known in advance and must be generated along with the polynomial $\Pi_j(x)$. Gautschi (1968) had shown that the system

$$m_j = \sum_{k=1}^{n} \alpha_k \Pi_j(x_k) \qquad j = 0,1,\ldots 2n-1 \tag{5}$$

is well conditioned if the Jacobian matrix

$$J(\underline{\alpha}, \underline{x}) = \begin{pmatrix} \Pi_0(x_1) & \ldots & \Pi_0(x_n) & \alpha_1 \Pi_0'(x_1) & \ldots & \alpha_n \Pi_0'(x_n) \\ \Pi_1(x_1) & \ldots & \Pi_1(x_n) & \alpha_1 \Pi_1'(x_1) & \ldots & \alpha_n \Pi_1'(x_n) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \Pi_{2n-1}(x_1) & \ldots & \Pi_{2n-1}(x_n) & \alpha_1 \Pi_{2n-1}'(x_1) & \ldots & \alpha_n \Pi_{2n-1}'(x_n) \end{pmatrix} \tag{6}$$

evaluated at the exact solution x_k, α_k is orthogonal. However in his iterative process he could only satisfy the exact orthogonality of the first approximation

$$J(\underline{\alpha}^{(o)}, \underline{x}^{(o)}) \tag{7}$$

while the remaining iterations result in nearly orthogonal matrices. Nevertheless, the test results indicated that the ill-conditioned character of the problem could often be avoided by essentially the same methods as Gautschi (1968) and Sack and Donovan (1972).

The present work is concerned with obtaining accurate results to the ordinary moment problem. For the solution of eq. (3) the generally recommended procedure consists of constructing the orthogonal polynomial

$$\begin{aligned} \Pi_n(x) &= (x-x_1)(x-x_2)\ldots(x-x_n) \\ &= x^n + c_{n-1}x^{n-1} + \ldots + c_0 \quad ; \quad c_n = 1 \end{aligned} \tag{8}$$

whose zeros are the required abscissae, (Hamming (1973), p.322).

Multiplying the j-th equation of (3) by c_j and summing we obtain

$$\sum_{j=0}^{n-1} g_j c_j + g_n = \sum_{k=1}^{n} \alpha_k \Pi_n(x_k) = 0 \tag{9}$$

Repeating this process for all equations in the system (3) shifting the multiplier c_j down by one line each time, the following n equations are obtained

$$\sum_{j=0}^{n-1} g_{j+k} c_j + g_{n+k} = 0 \qquad k = 0,1,\ldots,n-1 \tag{10}$$

This nonhomogenous system can be solved for c_j since the persymmetric determinant (named by Sylvester in 1853),

$$G = \begin{vmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-1} \\ g_1 & g_2 & g_3 & \cdots & g_n \\ g_2 & g_3 & g_4 & \cdots & g_{n+1} \\ \vdots & & & & \\ g_{n-1} & g_n & g_{n+1} & \cdots & g_{2n-2} \end{vmatrix} \tag{11}$$

is nonzero. This may be shown using the set of n homogeneous equations, and multiplying the k-th equation by c_k and summing. Thus

$$\begin{aligned} \sum_{k=0}^{n-1} \sum_{j=0}^{n-1} g_{j+k} c_j c_k &= \int_a^b w(x) \left\{ \sum_{k=0}^{n-1} c_k x^k \sum_{j=0}^{n-1} c_j x^j \right\} dx \\ &= \int_a^b w(x) \left[\sum_{k=0}^{n-1} c_k x^k \right]^2 dx = 0 \end{aligned} \tag{12}$$

Since $w(x) \geq 0$, eq. (12) holds only if

$$\sum_{k=0}^{n-1} c_k x^k \equiv 0 \tag{13}$$

that is when all $c_j = 0$, (Hamming (1973), p.324).

The coefficients c_j of the polynomial are the sums of the products of n Gaussian abscissae taken j at a time. Once these coefficients are calculated exactly, the problem will reduce to employing a root finding procedure. Since the weight function is nonzero on the interval of integration the zeros of the polynomial are known to be real distinct and to lie in the interior of the interval (Hamming (1973), p.325).

As the linear approach outlined above is so unstable in practice an attempt to by-pass the difficulty by solving an equivalent set of non-linear equations was made and is described in the following section.

6.2 The Use of Non-linear Equations

The method is based on the rearrangement of equation (6.1.10) in the form

$$\sum_{j=0}^{n-1} (-1)^{n-j} g_{j+k} \sigma_{n-j}(X) + g_{n+k} = 0 \qquad k = 0,1,\ldots,n-1 \qquad (1)$$

where the c_j term is replaced by $(-1)^{n-j} \sigma_{n-j}(X)$, where

$$X = \{ x_1, x_2, \ldots, x_n \} \qquad (2)$$

as is obvious from equation (6.1.8). This represents a set of non-linear equations in the variables $x_1, x_2, \ldots x_n$. The solution to equation (1) can be obtained by Newton's method for finding the roots of simultaneous non-linear equations, as described in Conte (1965, p.43). Defining F_k as

$$F_k(x_1, x_2, \ldots, x_n) = \sum_{j=0}^{n-1} (-1)^{n-j} g_{j+k} \sigma_{n-j}(X) + g_{n+k} = 0 \qquad k = 0,1,\ldots,n-1 \qquad (3)$$

and assuming that $F_k(\underline{x})$ is sufficiently differentiable and $\underline{x}^{(o)}$ is an approximation to the solution of system (3), Taylor's expansion about $\underline{x}^{(o)}$ gives

$$F_k(\underline{x}) = F_k(\underline{x}^{(o)}) + (\underline{x} - \underline{x}^{(o)}) \cdot \text{grad } F_k(\underline{x}^{(o)}) + \ldots \tag{4}$$

This yields the iteration

$$(\underline{x}^{(n+1)} - \underline{x}^{(n)}) \cdot \text{grad } F_k(\underline{x}^{(n)}) = - F_k(\underline{x}^{(n)}) \tag{5}$$

which converges to the required solution $\underline{x}$. The linear equations involved can be solved using the Gauss-Jordan method with pivoting, (Fox (1964), p.65 and 179).

Having found the abscissae, the first n equations of (6.1.3) will determine the weights α_k. In matrix notation

$$\underline{g} = V \underline{\alpha} \tag{6}$$

where V is the Vandermonde matrix of the abscissae, namely,

$$V = \begin{pmatrix} 1 & 1 & \ldots\ldots\ldots & 1 \\ x_1 & x_2 & \ldots\ldots\ldots & x_n \\ x_1^2 & x_2^2 & \ldots\ldots\ldots & x_n^2 \\ \vdots & & & \\ x_1^n & x_2^n & \ldots\ldots\ldots & x_n^n \end{pmatrix} \tag{7}$$

On inversion, (Faddeev and Sominskii (1965), p.70), the weights α_k are obtained in the form

$$\alpha_k = \frac{1}{\Pi_n'(x_k)} \sum_{j=0}^{n-1} (-1)^{n-j-1} g_j \, \sigma_{n-j-1}(X_{j+1}') \tag{8}$$

where

$$X'_{j+1} = \left\{x_1, x_2, \ldots, x_j, x_{j+2}, \ldots, x_n\right\} \tag{9}$$

and

$$\Pi'_n(x_k) = (x_k - x_1) \ldots (x_k - x_{k-1})(x_k - x_{k+1}) \ldots (x_k - x_n) \tag{10}$$

It will be noticed that the computation of the partial derivatives $\partial F_k/\partial x_i$, required in equation (4), is greatly facilitated by the relation

$$\frac{\partial}{\partial x_i}\ \sigma_{n-j}(X) = \sigma_{n-j-1}(X'_i) \tag{11}$$

where

$$\left\{X'_i\right\} = \left\{x_1, x_2, \ldots x_{i-1}, x_{i+1}, \ldots, x_n\right\} \tag{12}$$

Before starting the numerical applications, it is perhaps worthwhile pointing out that the coefficients of the two-point Gaussian quadrature formula can be written down in closed form, which is useful for checking the numerical procedure outlined above. In addition, it is instructive to note that the basic formulae (6.1.3) may also be derived by means of Taylor expansions based on the result

$$\int_a^b f(x)\ w(x)\ dx = \alpha_1\ f(x_1) + \alpha_2\ f(x_2) \tag{13}$$

Thus, utilizing the expansion

$$f(x) = f(0) + xf'(0) + \frac{1}{2!}\ x^2\ f''(0) + \ldots \tag{14}$$

on both sides of (13) and equating coefficients of $f^{(j)}(0)/j!$ for $j = 0,1,2$ and 3, the four equations

$$
\begin{aligned}
g_0 &= \alpha_1 + \alpha_2 \\
g_1 &= \alpha_1 x_1 + \alpha_2 x_2 \\
g_2 &= \alpha_1 x_1^2 + \alpha_2 x_2^2 \\
g_3 &= \alpha_1 x_1^3 + \alpha_2 x_2^3
\end{aligned}
\tag{15}
$$

for the unknown quantities $x_1 x_2, \alpha_1$ and α_2 are obtained immediately, exactly as in equation (6.1.3). This procedure is, of course, entirely equivalent to the undetermined coefficient approach adopted in the previous section.

Algebraic manipulation yields

$$
\begin{aligned}
x_2 &= \left[z g_0 + (z^2 g_0^2 - 4 g_0 (z g_1 - g_2))^{\frac{1}{2}} \right] / 2 g_0 \\
x_1 &= - x_2 + z \\
\alpha_1 &= (g_1 - g_0 x_2) / (z - 2 x_2) \\
\alpha_2 &= g_0 - \alpha_1
\end{aligned}
\tag{16}
$$

where

$$
z = (g_0 g_3 - g_1 g_2) / (g_0 g_2 - g_1^2) \tag{17}
$$

6.3 Numerical Tests

The following quadrature rules are generated.

	Name of the Quadrature Rule	Weight function	Interval
i)	Gauss-Legendre	1	$[-1, 1]$
ii)	Gauss-Hermite	e^{-x^2}	$[-\infty, \infty]$
iii)	Gauss-Laguerre	e^{-x}	$[0, \infty]$
iv)	Gauss-logarithm	$-\log_e x$	$[0, 1]$
v)	Gauss-sine	$\sin x$	$[0, \pi/2]$

Gauss-Legendre and Gauss-Hermite coefficients are calculated for $n \leq 10$ and the results are presented in Tables 1 and 2.

Comparisons with abscissae and weights quoted by Stroud and Secrest (1966, p.100 and 218) show agreement to near machine accuracy, using single precision arithmetic (37 binary digits, or roughly 10 decimal digits). There are small discrepancies in the quoted values due mainly to round-off in the Newton procedure and these are indicated by underlining the appropriate digits.

In Table 3, Gauss-Laguerre coefficients are presented for $n \leq 7$ and it is found that accuracy starts to fall rapidly at this order due to instability of basic equations. Nevertheless, this is an improvement on the algebraic approach where instability begins to appear even at $n = 4$.

Coefficients for case (iv) are quoted to only six significant figures up to order $n = 7$, as it is impossible to obtain more than a few significant digits for the higher order formulae. This difficulty

Table 1. Coefficients for the Gauss-Legendre quadrature rule.

n	x_i	α_i
2	± 0.57735027	1.00000000
3	± 0.77459667	0.55555555
	0.00000000	0.88888889
4	± 0.86113631	0.34785485
	± 0.33998104	0.65214515
5	± 0.90617985	0.23692689
	± 0.53846931	0.47862867
	0.00000000	0.56888889
6	± 0.93246951	0.17132449
	± 0.66120939	0.36076157
	± 0.23861919	0.46791393
7	± 0.94910791	0.12948497
	± 0.74153119	0.27970539
	± 0.40584515	0.38183005
	0.00000000	0.41795919
8	± 0.96028986	0.10122854
	± 0.79666648	0.22238104
	± 0.52553241	0.31370665
	± 0.18343464	0.36268378
9	± 0.96816024	(-1)0.81274389
	± 0.83603111	0.18064816
	± 0.61337143	0.26061070
	± 0.32425342	0.31234708
	0.00000000	0.33023935
10	± 0.97390653	(-1)0.66671344
	± 0.86506337	0.14945135
	± 0.67940957	0.21908637
	± 0.43339539	0.26926671
	± 0.14887433	0.29552423

Table 2. Coefficients for the Gauss-Hermite quadrature rule.

n	x_i	α_i
2	± 0.70710678	0.88622693
3	± 1.22474487	0.29540898
	0.00000000	1.18163590
4	± 1.65068012	(-1)0.81312835
	0.52464762	0.80491409
5	± 2.02018287	(-1)0.19953242
	± 0.95857246	0.39361932
	0.00000000	0.94530872
6	± 2.35060497	(-2)0.45300099
	± 1.33584907	0.15706732
	± 0.43607741	0.72462960
7	± 2.65196136	(-3)0.97178125
	± 1.67355163	(-1)0.54515583
	± 0.81628788	0.42560725
	0.00000000	0.81026462

n	x_i	α_i
8	± 2.93063742	(-3)0.19960407
	± 1.98165676	(-1)0.17077983
	± 1.15719371	0.20780233
	± 0.38118699	0.66114701
9	± 3.19099320	(-4)0.39606977
	± 2.26658058	(-2)0.49436243
	± 1.46855329	(-1)0.88474527
	± 0.72355102	0.43265156
	0.00000000	0.72023522
10	± 3.43615912	(-5)0.7640433<u>0</u>
	± 2.53273167	(-2)0.13436457
	± 1.75668365	(-1)0.33874394
	± 1.03661083	0.24013861
	± 0.34290133	0.61086263

Table 3. Coefficients for the Gauss-Laguerre quadrature rule.

n	x_i	α_i
2	0.58578644	0.85355339
	3.41421356	0.14644661
3	0.41577456	0.71109301
	2.29428036	0.27861773
	6.28994508	(-1)0.10389256
4	0.32254769	0.60315410
	1.74576110	0.35741869
	4.53662030	(-1)0.38887909
	9.39507091	(-3)0.53929470
5	0.26356032	0.52175561
	1.41340306	0.39866681
	3.59642577	(-1)0.75942449
	7.08581000	(-2)0.36117587
	(1)1.26408008	(-4)0.23369973

n	x_i	α_i
6	0.22284660	0.45896467
	1.18893210	0.41700083
	2.99273632	0.11337338
	5.77514355	(-1)0.10399198
	9.83746738	(-3)0.26101721
	(1)1.59828739	(-6)0.89854824
7	0.19304366	0.40931893
	1.02666482	0.42183128
	2.56787661	0.14712636
	4.90035291	(-1)0.20633518
	8.18215328	(-2)0.10740104
	(1)1.27341802	(-4)0.15865465
	(1)1.93957278	(-7)0.31703147

Table 4. Coefficients for $-\int_0^1 \log_e x \, f(x) \, dx$.

n	x_i	α_i
2	0.112009	0.718539
	0.602277	0.281461
3	(-1)0.638908	0.513405
	0.368997	0.391980
	0.766880	(-1)0.946154
4	(-1)0.414485	0.383464
	0.245275	0.386875
	0.556165	0.190435
	0.848982	(-1)0.392255
5	(-1)0.291345	0.297893
	0.173977	0.349776
	0.411702	0.234488
	0.677314	(-1)0.989305
	0.894771	(-1)0.189116

n	x_i	α_i
6	(-1)0.216340	0.238764
	0.129583	0.308287
	0.314020	0.245317
	0.538657	0.142009
	0.756915	(-1)0.554547
	0.922669	(-1)0.101690
7	(-1)0.167176	0.196154
	0.100176	0.270293
	0.246277	0.239686
	0.433444	0.165785
	0.632335	(-1)0.889508
	0.811109	(-1)0.331977
	0.940845	(-2)0.593344

associated with the logarithmic formula is well known (Anderson (1965)) and, indeed, makes the algebraic approach virtually unusable in the present form due to the near singular nature of the matrices involved. The non-linear approach postpones the advent of instability but is still far from satisfactory.

Similar remarks apply to the important case (v) for the trigonometric weight functions. Note that the present quadrature coefficients could be applied directly to the integrals of Chapter 5 between the zeros of the weight functions cos px or sin px. Linear transformations normalize the integration interval to $[0, \pi/2]$ and the weight function to sin x in both cases. Instability begins to appear in both the linear and the non-linear algorithms when n = 5 and for this reason the coefficients are quoted to only 7 figures here. Although these results are regarded as being only of a preliminary nature, comparisons with the related coefficients of Piessens (1970) have been carried out and the results given in Table 5 are expected to be accurate to the number of figures quoted, the questionable figures being underlined.

It is worthwhile pointing out that the results obtained on applying these quadrature rules for cases (iv) to (v) do not appear to be particularly sensitive to errors in the coefficients. Accurate numerical values of the monomial integrals are reproduced easily, even when fluctuations exist in the values of the weights and abscissae and care is needed to apply a sufficiently sensitive test. This point is raised by Stroud and Secrest (1966, p.27).

It is concluded that, although the introduction of the non-linear system into the computational scheme, in an attempt to increase accuracy,

Table 5. Coefficients for $\int_0^{\pi/2} \sin x\, f(x)\, dx$

n	x_i	α_i
2	0.5356437	0.3963745
	(1)0.1304922	0.6036255
3	0.3235279	0.1591450
	0.9046139	0.4768423
	(1)0.1420703	0.3640127
4	0.2143501	0.0726936
	0.6389416	0.2868153
	(1)0.1118384	0.4026582
	(1)0.1475126	0.2378328

n	x_i	α_i
5	0.1517149	0.0371544
	0.4685867	0.1688287
	0.8674892	0.3102768
	(1)0.1247020	0.3173873
	(1)0.1504647	0.1663528

delays the instability to higher order quadrature rules, it does not eliminate the instability altogether. The use of double precision arithmetic produces a small improvement in accuracy, but, in general, multiple precision procedures would be required. Currently, work is being carried out on improving the linear algorithm, so that accurate quadrature coefficients for general weight functions, including the particularly important sin x case, are expected to be available in the near future. However, in the remainder of the present work attention is confined to the commonly occuring case where the monomials are expressible in rational form (apart from multiplying constants) and the introduction of rational arithmetic produces essentially exact results. This method does not apply to the sin x case where the monomials may not be expressed in rational form.

6.4 Multilength Rational Arithmetic

The proposed method consists of the exact solution of the linear system (6.1.10) using multilength rational arithmetic. As this work involves the re-definition of the operators +, -, /, *, a convenient language to use is Algol 68. Use can also be made of 'structures' to define the relevant modes as follows:

```
mode  ℓint = struct (boo ℓ si, [1:N] int X) ;
mode  ℓrat = struct (ℓint num, den)
```

A ℓint is a N-tuple length integer with sign si (faℓse for + ve) and digits [1:N] int X. The implementation which was used limited integers to $2^{23} - 1$, and therefore each element of the array X was allowed to

take 6 digits only, X[1] being the most significant part and X[N] the least. A long rational ℓrat consists simply of a numerator num and denominator den.

For these modes a set of monadic and dyadic operators are defined using the built-in symbols such as <, >, =, −, +, *, /, ÷, and some indicants, for instance sign. All the arithmetic operators were designed to keep the resulting rationals in their simplest forms by immediate cancellation of factors. In this way artificial overflow, and the subsequent increase in the amount of arithmetic due to an enlarged N, was avoided. To this end Euclid's algorithm (Birkoff and Maclane (1965), p.16) was used to find hcf's in a procedure defined by

proc hcf = (ℓint I, J) ℓint .

For use with addition and subtraction routines, ℓcm's were found using

proc ℓcm = (ℓint I, J) ℓint .

Further special operators and procedures were:

op eq0 = (ℓint A) boolℓ : (delivers true if A is zero, else false)

op % = (ℓrat A) ℓrat : (brings A to its simplest form)

op £ = (ℓint A) ℓrat : (converts ℓint A to the corresponding ℓrat)

op sign = (ℓrat A) boolℓ : (delivers true if A is -ve else false)

op @ = (ℓint A) realℓ : (converts ℓint A to the corresponding real number)

op @ = (ℓrat A) realℓ : (converts ℓrat A to the corresponding real number)

proc printℓi = (ℓint A): (prints out ℓint A)

proc prinℓr = (ℓrat A): (prints out two ℓint's that form ℓrat A)

The set of operators and procedures was compiled and put into an album, from which it could be obtained for use in the problem.

a) Basic Operations on ℓint's

Addition and subtraction was performed in 6 digit blocks (each X [I]) with a carry mechanism arranged to connect successive blocks.

Multiplication and division operations are expressed by simple algorithms in which addition and subtractions are used, respectively.

For multiplication, two registers reg1 and reg2 defined by [1: 2* N] int reg1, reg2, initially cleared, are used as storage locations for interim results. The multiplier is loaded into the right of reg2 from locations N + 1 to 2N. Digits are extracted from the multiplicand starting with the least significant and these are used to control successive additions to reg1 from reg2. Contents of reg2 are shifted one place to the left after each digit of the multiplicand is dealt with, so simulating long multiplication. Some special internal procedures were written to effect the required shifts, digit extraction and sign organization.

Other implementations of these processes could have been used, this particular one being chosen to give programming simplicity to the algorithm, which can then be expressed at high level in the appendix.

The multiplication can be illustrated by a simple example considering 1011 003282 (multiplicand) and 21 (multiplier).

			Reg2		Reg1		
			000000	000000	000000	000000	
	multiplier			21		21	
						21	add multiplier
shift	Reg2	left		210		42	twice
						210	
						252	
						210	
						462	
						210	
						672	
						210	add 210,
						882	eight
						210	times
						1092	
						210	
						1302	
						210	
						1512	
						210	
shift	Reg2	left		2100		1722	
						2100	
						3822	add 2100,
						2100	twice
shift	Reg2	left		21000		5922	
						21000	
						26922	add
						21000	21000, three
						47922	times
						21000	
shift	Reg2	left		210000		68922	
shift	Reg2	left	2	100000			add
shift	Reg2	left	21	000000	21	000000	21 000000
					21	068922	once
shift	Reg2	left	210	000000	210	000000	add
shift	Reg2	left	2100	000000	231	068922	210 000000 once
shift	Reg2	left	21000	000000	21000	000000	add
					21231	068922	21000 000000 once
					Product		

Figure 1. Example of multiplication in double length integer arithmetic.

For division, two registers are defined by [1: N] int reg1, reg2 which are initially cleared. The dividend and the divisor are loaded into reg1 and reg2, respectively. Firstly, some of the possibilities which can occur are examined. If a zero divisor is detected the message "overflow in / " is printed out; a check is also made for a divisor being larger than the dividend, resulting in a zero quotient, and for equal dividend and divisor, to bypass the execution of the division algorithm altogether. Moreover, the consecutive zeros in the least significant digits of dividend and divisor are counted by repeatedly incrementing index numbers IA and IB, whilst shifting the registers to the right. The dividend and the divisor are then restored to the registers reg1 and reg2, and the division is carried out by a sequence of subtractions and shifts. The difference between two index values, IA and IB, determines the number of single left shifts to be applied to reg2, hence ensuring the correct position of the number in the register. The divisor is then repeatedly subtracted from reg1 while reg1 remains greater than reg2. The number of subtractions is counted and entered into the quotient. This process is repeated until the right shifts, which follow the subtractions, bring the divisor to its original location in reg2, or until reg2 exceeds reg1.

As an example 1 426010 000000 ÷ 457000 is considered. The index numbers IA = 7, IB = 3 determine the position of the number in Reg2.

	Reg1	
	1 426010 000000	
Reg2	4570 000000	subtract Reg2 repeatedly (number of subtractions = 312)
	1 421440 000000	
	4570 000000	
	1 416870 000000	
	4570 000000	
	1 412300 000000	
	⋮	
	170 000000	
Shift Reg2 right	457 000000	Reg1 > Reg2 is violated.
		Quotient = 3120000.

Figure 2. Example of division of (ℓint)s.

6.5 Applications

To begin with, rational arithmetic is used to generate the following polynomials:

	Name of the Orthogonal polynomial	Weight function	Interval	Monomials
i)	Legendre	1	$[-1, 1]$	$g_j = \begin{cases} 2/(j+1) & j \text{ even} \\ 0 & j \text{ odd} \end{cases}$
ii)	Laguerre	e^{-x}	$[0, \infty]$	$g_j = j!$
iii)	Chebyshev (first kind)	$(1-x^2)^{-1/2}$	$[-1, 1]$	$g_j = \begin{cases} \frac{\pi(j-1)!!}{j\ !!} & j \text{ even} \\ 0 & j \text{ odd} \end{cases}$

The coefficients of these polynomials up to order $n \leq 10$ are checked with those of Stroud and Secrest (1966, p.84). The results are tabulated in Tables 6, 7 and 8.

The monomials for the Chebyshev polynomials are multiples of π, which is treated as a parameter. The resulting system of algebraic equations is solved by rational arithmetic, using a symmetric version of the Gaussian elimination routine. Cholesky's method is found unsuitable for the purpose since the diagonal elements of the resulting triangular matrix involve the calculation of square roots.

Next, the polynomials for weight function $w(x) = -\log_e x$ are considered. These polynomials have been described by Stroud and Secrest (1966, p.90) as being extremely difficult to compute. In the literature, exact results are quoted only for orders $n \leq 4$, (Abramowitz and Stegun (1965, p.920) or Stroud and Secrest (1966, p.90)). Using multilength rational arithmetic this table is extended up to order 7 (Table 9), and the results recorded here justify that the method can be used with confidence for any weight function, if the moments g_j are known algebraically and their j-dependent part can be expressed in rational form.

The ability to generate Gaussian quadrature formulae, is also of considerable interest for integrating singular functions, where the singularity may be included in the weight function. The rate of convergence depends, of course, on the type of singularity. Davis and Rabinowitz (1965) and Rabinowitz (1967) have stated that the most of the common quadrature formulae converge not only for continuous functions, but also for monotonically increasing singular functions, whose singularities lie at one or both end points. As an example, a

Table 6. Legendre polynomials.

$$P_1 = x$$

$$P_2 = x^2 - \frac{1}{3}$$

$$P_3 = x^3 - \frac{3}{5}x$$

$$P_4 = x^4 - \frac{6}{7}x^2 + \frac{3}{35}$$

$$P_5 = x^5 - \frac{10}{9}x^3 + \frac{5}{21}x$$

$$P_6 = x^6 - \frac{15}{11}x^4 + \frac{5}{11}x^2 - \frac{5}{231}$$

$$P_7 = x^7 - \frac{21}{13}x^5 + \frac{105}{143}x^3 - \frac{35}{429}x$$

$$P_8 = x^8 - \frac{28}{15}x^6 + \frac{14}{13}x^4 - \frac{28}{143}x^2 + \frac{7}{1287}$$

$$P_9 = x^9 - \frac{36}{17}x^7 + \frac{126}{85}x^5 - \frac{84}{221}x^3 + \frac{63}{2431}x$$

$$P_{10} = x^{10} - \frac{45}{19}x^8 + \frac{630}{323}x^6 - \frac{210}{323}x^4 + \frac{315}{4199}x^2 - \frac{63}{46189}$$

Table 7. Laguerre polynomials

$$P_1 = x - 1$$

$$P_2 = x^2 - 4x + 2$$

$$P_3 = x^3 - 9x^2 + 18x - 6$$

$$P_4 = x^4 - 16x^3 + 72x^2 - 96x + 24$$

$$P_5 = x^5 - 25x^4 + 200x^3 - 600x^2 + 600x - 120$$

$$P_6 = x^6 - 36x^5 + 450x^4 - 2400x^3 + 5400x^2 - 4320x + 720$$

$$P_7 = x^7 - 49x^6 + 882x^5 - 7350x^4 + 29400x^3 - 52920x^2 + 35280x - 5040$$

$$P_8 = x^8 - 64x^7 + 1568x^6 - 18816x^5 + 117600x^4 - 376320x^3 + 564480x^2 - 322560x + 40320$$

$$P_9 = x^9 - 81x^8 + 2592x^7 - 42336x^6 + 381024x^5 - 1905120x^4 + 5080320x^3 - 6531840x^2 + 3265920x - 362880$$

$$P_{10} = x^{10} - 100x^9 + 4050x^8 - 86400x^7 + 1058400x^6 - 7620480x^5 + 31752000x^4 - 72576000x^3 + 81648000x^2 - 36288000x + 3628800$$

Table 8. Chebyshev polynomials.

$$P_1 = x$$

$$P_2 = x^2 - \frac{1}{2}$$

$$P_3 = x^3 - \frac{3}{4}x$$

$$P_4 = x^4 - x^2 + \frac{1}{8}$$

$$P_5 = x^5 - \frac{5}{4}x^3 + \frac{5}{16}x$$

$$P_6 = x^6 - \frac{3}{2}x^4 + \frac{9}{16}x^2 - \frac{1}{32}$$

$$P_7 = x^7 - \frac{7}{4}x^5 + \frac{7}{8}x^3 - \frac{7}{64}x$$

$$P_8 = x^8 - 2x^6 + \frac{5}{4}x^4 - \frac{1}{4}x^2 + \frac{1}{128}$$

$$P_9 = x^9 - \frac{9}{4}x^7 + \frac{27}{16}x^5 - \frac{15}{32}x^3 + \frac{9}{256}x$$

$$P_{10} = x^{10} - \frac{5}{2}x^8 + \frac{35}{16}x^6 - \frac{25}{32}x^4 + \frac{25}{256}x^2 - \frac{1}{512}$$

Table 9. Polynomials for $-\log_e x$ on $[0, 1]$

$$P_0 = 1$$

$$P_1 = x - \frac{1}{4}$$

$$P_2 = x^2 - \frac{5}{7}x + \frac{17}{252}$$

$$P_3 = x^3 - \frac{3105}{2588}x^2 + \frac{5751}{16175}x - \frac{4679}{258800}$$

$$P_4 = x^4 - \frac{165196}{97641}x^3 + \frac{67227}{75943}x^2 - \frac{79564}{531601}x + \frac{2296639}{478440900}$$

$$P_5 = x^5 - \frac{17692\ 971425}{8090\ 435556}x^4 + \frac{2\ 449515\ 716800}{1\ 474481\ 880081}x^3 - \frac{112304\ 929775}{218441\ 760012}x^2 + \frac{203478\ 628075}{3567882\ 080196}x - \frac{1\ 461977\ 847751}{1155\ 993793\ 983504}$$

$$P_6 = x^6 - \frac{15700\ 658824\ 389411}{5850\ 859031\ 888599}x^5 + \frac{7\ 604560\ 816456\ 422375}{2\ 831815\ 771434\ 081916}x^4 - \frac{873930\ 519668\ 513600}{707953\ 942858\ 520479}x^3 + \frac{182360\ 435978\ 518375}{707953\ 942858\ 520479}x^2 - \frac{57568\ 874774\ 479529}{2831815\ 771434\ 081916}x + \frac{1\ 654296\ 840628\ 723409}{4995\ 323020\ 809720\ 499824}$$

$$P_7 = x^7 - \frac{374764\ 236038\ 061105\ 347545}{117814\ 060676\ 696250\ 433948}x^6 + \frac{19\ 703487\ 975349\ 105912\ 246983}{4\ 977644\ 063590\ 416580\ 834303}x^5 - \frac{193\ 910114\ 421075\ 347658\ 883625}{79\ 642305\ 017446\ 665293\ 348848}x^4 + \frac{462\ 961491\ 852963\ 682406\ 237375}{602\ 294931\ 694440\ 406280\ 950663}x^3 - \frac{280\ 872418\ 651357\ 455258\ 635313}{2409\ 179726\ 777761\ 625123\ 802652}x^2 + \frac{37\ 439927\ 629086\ 176699\ 166095}{5420\ 654385\ 249963\ 656528\ 555967}x - \frac{29\ 938206\ 191019\ 047895\ 615767}{346921\ 880655\ 997674\ 017827\ 581888}$$

weight function with square root singularities is considered. The moments are:

$$\int_0^1 \frac{x^j}{\sqrt{x(1-x)}}\,dx = \int_0^{\pi/2} 2\sin^{2j}\theta\,d\theta = \pi\,\frac{(2j-1)!!}{2j!!} \qquad (1)$$

Polynomials for $w(x) = [\,x(1-x)\,]^{-1/2}$ on $[0, 1]$ are tabulated in Table 10.

The zeros of the above polynomials denote the abscissae for the Gauss quadrature rule. They were calculated easily to machine accuracy by Bairstow's method. Tables 11 and 12 illustrate the coefficients for Gauss-Laguerre and Gauss-Chebyshev rules for $n \leq 10$. Comparisons with Table 11, Table 3 and Stroud and Secrest (1966, p.254) confirm the accuracy of the method. The evaluation of the weights is based on equation (6.2.8) and involves a considerable amount of algebra. The agreement with Stroud and Secrest is within the limits of single precision arithmetic. However, higher precision can be obtained by more refined arithmetic. Similarly, Tables 13 and 14 illustrate the coefficients for the polynomials listed in Tables 9 and 10.

Chebyshev polynomials and the polynomials for the weight function $w(x) = [\,x(1-x^2)\,]^{-1/2}$ on $[0, 1]$ are special cases for which the abscissae x_i and the weights of the corresponding quadrature rules are obtainable in an analytic form. Thus, for the Chebyshev case,

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\,dx \approx \frac{\pi}{n}\sum_{i=1}^{n} f\left(\cos\frac{2i-1}{2n}\pi\right) \qquad (2)$$

as quoted in the literature (Hildebrand (1956, p.331), Krylov (1962, p.179), Kopal (1961, p.384) or Hamming (1973, p.327)). Similarly,

Table 10. Polynomials for $[x(1-x)]^{-1/2}$ on $[0, 1]$.

$$P_1 = x - \frac{1}{2}$$

$$P_2 = x^2 - x + \frac{1}{8}$$

$$P_3 = x^3 - \frac{3}{2}x^2 + \frac{9}{16}x - \frac{1}{32}$$

$$P_4 = x^4 - 2x^3 + \frac{5}{4}x^2 - \frac{1}{4}x + \frac{1}{128}$$

$$P_5 = x^5 - \frac{5}{2}x^4 + \frac{35}{16}x^3 - \frac{25}{32}x^2 + \frac{25}{256}x - \frac{1}{512}$$

$$P_6 = x^6 - 3x^5 + \frac{27}{8}x^4 - \frac{7}{4}x^3 + \frac{105}{256}x^2 - \frac{9}{256}x + \frac{1}{2048}$$

$$P_7 = x^7 - \frac{7}{2}x^6 + \frac{77}{16}x^5 - \frac{105}{32}x^4 + \frac{147}{128}x^3 - \frac{49}{256}x^2 + \frac{49}{4096}x - \frac{1}{8192}$$

$$P_8 = x^8 - 4x^7 + \frac{13}{2}x^6 - \frac{11}{2}x^5 + \frac{165}{64}x^4 - \frac{21}{32}x^3 + \frac{21}{256}x^2 - \frac{1}{256}x + \frac{1}{32768}$$

$$P_9 = x^9 - \frac{9}{2}x^8 + \frac{135}{16}x^7 - \frac{273}{32}x^6 + \frac{1287}{256}x^5 - \frac{891}{512}x^4 + \frac{693}{2048}x^3 - \frac{135}{4096}x^2 + \frac{81}{65536}x - \frac{1}{131072}$$

$$P_{10} = x^{10} - 5x^9 + \frac{85}{8}x^8 - \frac{25}{2}x^7 + \frac{2275}{256}x^6 - \frac{1001}{256}x^5 + \frac{2145}{2048}x^4 - \frac{165}{1024}x^3 + \frac{825}{65536}x^2 - \frac{25}{65536}x + \frac{1}{524288}$$

Table 11. Coefficients for the Gauss-Laguerre Quadrature Rule.

n	x_i	α_i
2	0.58578644	0.85355339
	3.41421356	0.14644661
3	0.41577456	0.71109301
	2.29428036	0.27851773
	6.28994508	(-1)0.10389257
4	0.32254769	0.60315410
	1.74576110	0.35741869
	4.53662030	(-1)0.38887909
	9.39507091	(-3)0.53929471
5	0.26356032	0.52175561
	1.41340306	0.39866811
	3.59642577	(-1)0.75942450
	7.08581001	(-2)0.36117587
	(1)1.26408008	(-4)0.23369973
6	0.22284660	0.45896467
	1.18893210	0.41700083
	2.99273633	0.11337338
	5.77514357	(-1)0.10399197
	9.83746742	(-3)0.26101720
	(1)1.59828740	(-6)0.89854794
7	0.19304368	0.40931895
	1.02666490	0.42183128
	2.56787675	0.14712635
	4.90035308	(-1)0.20633514
	8.18215345	(-2)0.10740101
	(1)1.27341803	(-4)0.15865464
	(1)1.93957279	(-7)0.31703167

n	x_i	α_i
8	0.17027963	0.36918859
	0.90370178	0.41878678
	2.25108663	0.17579499
	4.26670017	(-1)0.33343492
	7.04590539	(-2)0.27945362
	(1)1.07585160	(-4)0.90765091
	(1)1.57406786	(-6)0.84857444
	(1)2.28631317	(-8)0.10480156
9	0.15232223	0.33612642
	0.80722002	0.41121398
	2.00513516	0.19928753
	3.78347397	(-1)0.47460563
	6.20495678	(-2)0.55996266
	9.37298524	(-3)0.30524976
	(1)1.34662369	(-5)0.65921235
	(1)1.88833598	(-7)0.41107641
	(1)2.63740719	(-10)0.32909036
10	0.13779347	0.30844111
	0.72945455	0.40111993
	1.80834290	0.21806829
	3.40143370	(-1)0.62087456
	5.55249614	(-2)0.95015168
	8.33015271	(-3)0.75300842
	(1)1.84378593	(-4)0.28259223
	(1)1.62792578	(-6)0.42493149
	(1)2.19965858	(-8)0.18395616
	(1)2.99206970	(-12)0.99190046

Table 12. Coefficients for the Gauss-Chebyshev Quadrature Rule.

n	x_i	α_i
2	±0.70710678	1.57079633
3	±0.86602540	1.04719755
	0.00000000	1.04719755
4	±0.38268343	0.78539816
	±0.92387953	0.78539816
5	±0.58778525	0.62831853
	±0.95105652	0.62831853
	0.00000000	0.62831853
6	±0.25881905	0.52359878
	±0.70710678	0.52359878
	±0.96592583	0.52359878
7	0.43388374	0.44879895
	±0.78183148	0.44879895
	±0.97492791	0.44879895
	0.00000000	0.44879895
8	±0.19509032	0.39269908
	±0.55557023	0.39269908
	±0.83146961	0.39269908
	±0.98078528	0.39269908
9	±0.34202014	0.34906585
	±0.64278761	0.34906585
	±0.86602541	0.34906585
	±0.98480775	0.34906585
	0.00000000	0.34906585
10	±0.15643447	0.31415927
	±0.45399050	0.31415927
	±0.70710678	0.31415927
	±0.89100652	0.31415927
	±0.98768834	0.31415927

Table 13. Coefficients for $-\int_0^1 \log_e x \, f(x)\, dx$

n	x_i	α_i
2	0.11200880	0.71853932
	0.60227691	0.28146068
3	(-1)0.63890793	0.51340455
	0.36899706	0.39198004
	0.76688030	(-1)0.94615407
4	(-1)0.41448480	0.38346407
	0.24527491	0.38687532
	0.55616545	0.19043513
	0.84898240	(-1)0.39225487
5	(-1)0.29134472	0.29789347
	0.17397721	0.34977623
	0.41170252	0.23448829
	0.67731417	(-1)0.98930460
	0.89477136	(-1)0.18911552
6	(-1)0.21634006	0.23876366
	0.12958339	0.30828657
	0.31402045	0.24531742
	0.53865722	0.14200876
	0.75691534	(-1)0.55454622
	0.92266885	(-1)0.10168959
7	(-1)0.16719355	0.19616939
	0.10018568	0.27030265
	0.24629425	0.23968187
	0.43346349	0.16577577
	0.63235099	(-1)0.88943227
	0.81111862	(-1)0.33194305
	0.94084817	(-2)0.59327870

Table 14. Coefficients for $\int_0^1 [x(1-x)]^{-1/2} f(x)\, dx$

n	x_i	α_i
2	0.14644661	1.57079633
	0.85355339	1.57079633
3	(-1)0.66987298	1.04719755
	0.50000000	1.04719755
	0.93301270	1.04719755
4	(-1)0.38060234	0.78539816
	0.30865828	0.78539816
	0.69134172	0.78539816
	0.96193977	0.78539816
5	(-1)0.24471742	0.62831853
	0.20610737	0.62831853
	0.50000000	0.62831853
	0.79389263	0.62831853
	0.97552826	0.62831853

n	x_i	α_i
6	(-1)0.17037087	0.52359877
	0.14644661	0.52359877
	0.37059048	0.52359878
	0.62940952	0.52359878
	0.85355339	0.52359878
	0.98296291	0.52359877
7	(-1)0.12536044	0.44879895
	0.10908426	0.44879898
	0.28305813	0.44879895
	0.49999999	0.44879897
	0.71694187	0.44879894
	0.89091574	0.44879897
	0.98746396	0.44879894

$$\int_0^1 \frac{f(x)}{\sqrt{x(1-x)}}\,dx \simeq \frac{\pi}{n}\sum_{i=1}^{n} f\left(\cos^2\ \frac{2i-1}{2n}\,\frac{\pi}{2}\right) \tag{3}$$

which is useful for checking purposes.

Appendix

The following procedures are defined for long rational arithmetic.

```
mode ℓint=struct(boolℓ si,[1:n] int x);
c defines an n-tuple length integer with sign +ve if boolean is false c

mode ℓrat=struct(ℓint num,den);
c long rational, lrat, is defined consisting of numerator, num, and
denominator, den, of mode lint. c

ℓint zp,zq:
si of zp←falℓse; si of zq←falℓse;
for i to n-1 do((x of zp)[i]←0; (x of zq)[i]←0);
(x of zp)[n]←1; (x of zq)[n]←0;
c Two lints zp and zq are declared and assigned unity and zero. c
ℓint one=zp, zero=zq;

proc ad=(int n,ref[ ]int a,b,c):
begin
c In the particular implementation the values of mode int are packed into
six-digit machine words, hence the addition operation applies to each
six-digit integer part. The interaction between each part is taken
care of by the carry digit, ca, where necessary. Array c retains the
summation of a and b. c
   int ca←0;
   for i from n by -1 to 1 do
   begin
     c[i]←a[i]+b[i]+ca;
     (c[i]≥1000000 | c[i] minus 1000000; ca←1 | ca←0 )
   end
end;
```

```
proc mii=(int n,ref[ ]int a,b,c):
begin
c This procedure defines the subtraction operation, the array c retains
the subtraction result, ca and cb effect as carry digits. c
  int ca←0, cb←0;
  for i from n by -1 to 1 do
  begin
    (a[i]<b[i]+cb | a[i] plus 1000000; ca←1 | ca←0);
    c[i]←a[i]-b[i]-cb; cb←ca
  end
end;

proc dig=(int n,i,[ ]int x) int:
begin
c This is a procedure that extracts the i'th digit from x, and is used
in operator *. c
  int iw,ic,ia,ib,id,id1←1;
  iw←n-(i-1)÷6; ic←i-((i-1)÷6)*6;
  to ic-1 do id1 times 10; id←id1*10;
  (ic≠6 | ia←(x[iw] ÷id)*id; ib←x[iw]-ia | ib←x[iw]);
  (ic≠1 | ia←ib÷id1 | ia←ib)
end;

proc putdig=(int n,i,ref[ ]int x, int in):
begin
c This is a procedure to deposit digits in x and is used in operator ÷ . c
  int iw,ic,id←1; iw←n-(i-1)÷6;
  ic←i-((i-1)÷6)*6;
  to ic-1 do id times 10; x[iw] plus (ic≠1 | in*id | in)
end;
```

```
op>=(ref[ ]int a,b) bool:
begin
  int n←upb a;  c upper bound of a c
  bool bl; c True is delivered if value of a is greater than b,
  otherwise false is delivered. c
  for i to n do
    if a[i] = b[i] then skip
                   else (a[i]>b[i] | bl←true | bl←false); goto l1
    fi;
  bl←false;
l1: skip ;
  bl
end;

proc shiftl=(int n,ref[ ]int x):
begin
c This is a procedure to shift the contents of the array x one digit
to the left. c
  int ca;
  for i to n do
  begin
    (i≠n | ca←x[i+1] ÷ 1000000; x[i+1] minus ca*1000000 | ca←0);
    x[i]←x[i] *10+ca
  end
end;
```

```
proc shiftr=(int n,ref[ ]int x):
begin
c This is a procedure to shift the contents of the array x one digit
to the right. c
int ca;
for i from n by -1 to 1 do
  begin
    (i≠1 | ca←x[i-1]-10*(x[i-1] ÷10) | ca←0);
    x[i]←x[i]÷10+ca*1000000
  end
end;

op>=(ℓint a,b) boolℓ:
begin
  boolℓ bl; c true is delivered if value of a is greater than b c
  if si of a and not (si of b) then bl←falℓse
  elℓsf not (si of a) and si of b then bl←true
  elℓsf not (si of a) and not (si of b) then
    for i to n do
      if (x of a)[i]=(x of b)[i]then skip
      elℓse ( (x of a)[i]<(x of b)[i] | bl←falℓse | bl←true); goto ℓ1
      fi;
  bl←falℓse;
ℓ1: skip
  elℓsf si of a and si of b then
    for i to n do
      if (x of a)[i]=(x of b)[i] then skip
      elℓse if (x of a)[i]<(x of b)[i]then bl←true elℓse bl←falℓse fi;
      goto ℓ2
      fi;
    bl←falℓse;
```

```
ℓ2: skip
  fi;
  b1
end;

op<=(ℓint a,b) boolℓ:
begin
  boolℓ b1; c true is delivered if value of b is greater than a c
  if si of a and not (si of b) then b1←true
  elsf not (si of a) and si of b then b1←falℓse
  elsf not (si of a) and not (si of b) then
    for i to n do
      if (x of a)[i] =(x of b)[i]then skip
      elℓse ( (x of a)[i]<(x of b)[i] | b1←true | b1←falℓse); goto ℓ1
      fi;
   b1←falℓse;
 ℓ1: skip
   elsf si of a and si of b then
     for i to n do
       if (x of a)[i] =(x of b)[i] then skip
       elℓse if (x of a)[i]<(x of b)[i] then b1←falℓse elℓse b1←true fi;
       goto ℓ2
       fi;
     b1←falℓse;
```

```
ℓ2: skip
  fi;
  bl
end;

op==(ℓint a,b) boolℓ:
begin
  boolℓ bl; c true is delivered if a=b, and false otherwise c
  if (si of a and not (si of b)) or (not(si of a) and si of b) then bl←falseℓ
  elℓse for i to n do
        if (x of a)[i]=(x of b)[i] then skip
        elℓse bl←falseℓ; goto ℓ1 fi;
  bl←true;
ℓ1: skip
  fi;
  bl
end;

op eq0 =(ℓint a) boolℓ:
begin
  boolℓ bl; c true is delivered if a is zero and false otherwise c
  for i to n do
    if (x of a)[i]=0 then skip elℓse bl←falseℓ; goto ℓ1 fi;
    bl←true;
ℓ1:bl
end;

op abs =(ℓint a) ℓint:
begin
c delivers +ve sign for lint a c
  ℓint c←a; si of c←falseℓ;
  c
end;
```

```
op - =(ℓint a)ℓint:
begin
c changes the sign of lint a c
  ℓint c←a;
  si of c←not (si of a);
  c
end;

op + =(ℓint aa,bb)ℓint:
begin
c Addition of two lints is carried out and the sum is delivered. c
  ℓint c,a,b; a←aa; b←bb;
  if not (si of a)and not (si of b)
  then ad(n,x of a, x of b, x of c); si of c←false
  elsf si of a and si of b
    then ad(n, x of a, x of b, x of c); si of c←true
    elsf not (si of a) and si of b and abs a>abs b
      then mii(n, x of a, x of b, x of c); si of c←false
      elsf not (si of a) and si of b and abs a<abs b
        then mii(n, x of b, x of a, x of c); si of c←true
        elsf si of a and not (si of b) and abs b>abs a
          then mii(n, x of b, x of a, x of c); si of c←false
          elsf si of a and not (si of b) and abs b<abs a
            then mii(n, x of a, x of b, x of c); si of c←true
            elsf eq0 a then c←b elsf eq0 b then c←a
            else c←zero
  fi;
  c
end;
```

```
proc mul =(ref[ ]int a,b,c):
begin
c This is a procedure which delivers c, the product of a and b.
A detailed description is given in the text. c
  int n←upb a,j;
  [1:2*n] int reg1, reg2; c two registers c
  clear reg1; clear reg2;
  reg2 [n+1:2*n]←b;
  for i to 6*n do
  begin
    j←dig (n,i,a);
    (j=0 | skip | for i1 to j do ad(2*n,reg1,reg2,reg1) );
    shiftl (2*n,reg2)
  end;
  c←reg1
end;

op - =(lint a,b)lint:
begin
c dyadic operator delivering a-b c
  lint c; c←a+(-b)
end;

op * =(lint a,b)lint:
begin
c dyadic operator delivering a*b, as described in the text, the
multiplier is added to the sum of partial products stored in reg1, as
many times as required by each multiplicand digit, specified by proc dig c
```

```
  ℓint c,d; int j; [1:2*n] int reg1,reg2;
  cℓear reg1; cℓear reg2;
  reg2 [n+1:2*n]← x of b;
  for i to 6*n do
  begin
    j←dig(n,i,x of a);
    (j=0 | skip | for il to j do ad(2*n,reg1,reg2,reg1) );
    shiftℓ (2*n,reg2)
  end;
  x of d←reg1 [1:n];
  (not (eq0 d) | print ((newline, "overflow in *", newline)) );
  c To ensure that the product of two n*6 digit long integers can be
  accommodated within a 2*n-tuple integer, a check for overflow is
  carried out. c
  x of c←reg1 [n+1:2*n];
  if (si of a and si of b) or (not (si of a) and not (si of b))
  then si of c←falℓse elℓse si of c←true
  fi;
  c
end;
op ÷ =(ℓint a,b)ℓint:
begin
c dyadic operator delivering a÷b as described in the text, the number
of subtractions is counted and the correct quotient digit is deposited
by the procedure putdig c
```

```
  ℓint c;
  int ia←0, ib←0, id;
  [1:n] int reg1,reg2,nort;
  for i1 to n do nort[i1]←0;
  cℓear reg1; cℓear reg2; cℓear (x of c);
  reg1←x of a; reg2←x of b;
  if reg2>reg1 then c←zero
  eℓsf b=zero then print ((newline, "overflow in /", newline));
  c check for a zero divisior is made c
  c←zero
  eℓsf a=b then c←one
  eℓse whiℓe reg1>nort do (shiftr  (n,reg1); ia pℓus 1);
       whiℓe reg2>nort do (shiftr  (n,reg2); ib pℓus 1);
       reg1←x of a; reg2←x of b;
       to ia-ib do shiftℓ(n,reg2);
       for ic from ia-ib+1 by -1 to 1 do
       begin
         id←0;
       ℓ1:skip;
         if reg2>reg1 then skip
         eℓse mii(n,reg1,reg2,reg1); id pℓus 1; goto ℓ1 fi;
         putdig (n,ic,x of c, id);
         shiftr (n,reg2)
       end
  fi;
  if (si of a and si of b) or (not si of a) and not (si of b) )
  then si of c←faℓse eℓse si of c←true fi;
  c
end;
```

```
proc hcf =(ℓint i,j)ℓint:
begin
c Highest common factor of two lint's is evaluated by Euclid's algorithm. c
  ℓint i0,i1,a,i2,i3;
  (eq0 i or eq0 j | a←one; goto ℓ1);
  (i=j | a←i; goto ℓ1 | : i<j | i0←j; i1←i | i0←i;i1←j);
  while not (eq0 i1) do (i3←i0÷i1; i2←i0-i1*i3; i0←i1; i1←i2);
  a←i0;
ℓ1:a
end;

proc ℓcm = (ℓint i,j)ℓint:
begin
c Lowest common multiple of two lint's is delivered. c
  ℓint i0,i1; i0←hcf(i,j); i1←i÷i0;
  j*i1
end;

op % =(ℓrat a) ℓrat:
begin
c Possible cancellations of the numerator with denominator are
considered and the simplest form of a is delivered. c
  ℓint i1←hcf (abs (num of a), abs (den of a));
  ℓrat c; num of c←num of a÷i1;
  den of c←den of a÷i1;
  ℓrat←c
end;
```

```
proc printli = (lint a):
begin
c outputs lint a c
  format fl = $ 6dx $;
  (si of a | print ("-") | print ("+"));
  outf (stand out, fl, x of a)
end;

op + =(lrat a,b)lrat:
begin
c sums up two lrat's and delivers a lrat in the simplest form c
  lint il; lrat c;
  den of c←lcm (abs(den of a), abs(den of b));
  il←den of c;
  num of c←(il÷(den of a))*num of a + (il÷(den of b))*num of b;
  lrat←% c
end;

op - =(lrat a,b)lrat:
begin
c performs the subtraction and delivers the result in the simplest form c
  lint il; lrat c;
  den of c←lcm (abs (den of a), abs (den of b));
  il←den of c;
  num of c←(il÷(den of a))*num of a - (il÷(den of b))*num of b;
  lrat←% c
end;
```

```
op * =(lrat a,b)lrat:
begin
c  All cancellations on the numerators and denominators are performed
and the product of a and b is delivered.  c
  lint i1,i2,i3,i4,i5,i6; lrat c;
  i1←hcf(abs (num of a), abs(den of b));
  i2←hcf(abs (den of a), abs(num of b));
  (not (i1=one) | i3←num of a÷i1; i4←den of b÷i1
                | i3←num of a; i4←den of b);
  (not (i2=one) | i5←den of a÷i2; i6←num of b÷i2
                | i5←den of a; i6←num of b);
  num of c←i3*i6; den of c←i4*i5;
  lrat←c
end;
op / =(lrat a,b)lrat:
begin
c All cancellations on the numerators and denominators are performed
and the resulting lrat is delivered c
  lint i1,i2,i3,i4,i5,i6; lrat c;
  i1←hcf(abs (num of a), abs(num of b));
  i2←hcf(abs (den of a), abs(den of b));
  (not (i1=one) | i3←num of a÷i1; i4←num of b÷i1
                | i3←num of a; i4←num of b);
  (not (i2=one) | i5←den of a÷i2; i6←den of b÷i2
                | i5←den of a; i6←den of b);
  num of c←i3*i6; den of c←i4*i5;
  lrat←c
end;
```

```
op - =(ℓrat a)ℓrat:
begin
c monadic operator effects a change of sign c
  ℓrat c; num of c← -num of a;
  den of c←den of a;
  ℓrat←c
end;

op £ = (ℓint a)ℓrat:
begin
c assigns lint one as a denominator for lint a, and delivers a lrat c
  (ℓrat c; num of c←a; den of c←one;
  ℓrat←c
end;

op = =(ℓrat a,b)boolℓ:
begin
c delivers true if the value of two lrat's are equal, and false otherwise c
  boolℓ c;
  c←((num of a=num of b) and (den of a=den of b)) or
    ((num of a= -num of b)and (den of a=-den of b)); c
end;

op sign = (ℓrat a)boolℓ:
begin
c delivers true if sign of lrat a is negative, and false otherwise c
  if (si of num of a) then (si of den of a | false | true)
                      else (si of den of a | true | false)
  fi
end;
```

```
op < = (lrat a,b)bool:
begin
c delivers true if lrat b is greater than lrat a and false otherwise c
  if not (a=b) then if sign a and not (sign b) then true
                    elsf not (sign a) and sign b then false
                    elsf not (sign a) and not (sign b)
                      then sign (a-b)
                      else not (sign (a-b))
                    fi
               else false
  fi
end;
op > = (lrat a,b)bool:
begin
c delivers true if lrat a is greater than lrat b and false otherwise c
  if not (a=b) then if sign a and not (sign b) then false
                    elsf not (sign a) and sign b then true
                    elsf not (sign a) and not (sign b)
                    then not sign (a-b)
                    else (sign (a-b))
                    fi
               else false
  fi
end;
proc printlr = (lrat a):
begin
c outputs lrat a c
  printli (num of a);
```

```
  (n>8 | newline (stand out) | print (("/"));
  printlli (den of a)
end;

op @ = (llint a)real:
begin
c converts lint a to the corresponding real number c
  real s←0.0, c←1.0;
  for i1 from n by -1 to 1 do
  (s plus (x of a)[i1] *c; c times 1.0 & + 06);
  (si of a | -s | s)
end;

op @ = (lrat a)real:
c converts lrat a to the corresponding real number c
  ( @(num of a)/ @(den of a));

op + = (lrat a, lint b)lrat:
begin
c  Addition of mixed modes, lrat a with lint b is performed and resulting
lrat is delivered. c
  lrat c; num of c←num of a+b*den of a;
  den of c←den of a;
  lrat←% c
end;

op + = (lint a, lrat b)lrat:
begin
c Addition of mixed modes, lint a with lrat b is performed and resulting
lrat is delivered. c
```

```
  lrat c; num of c←num of b + a*den of b;
  den of c←den of b;
  lrat←% c
end;

op - = (lrat a, lint b)lrat:
begin
c Subtraction of mixed modes, lrat a and lint b is performed and
resulting lrat is delivered. c
  lrat c; num of c←num of a-b*den of a;
  den of c←den of a;
  lrat←% c
end;

op - = (lint a, lrat b)lrat:
begin
c Subtraction of mixed modes, lint a and lrat b is performed and
resulting lrat is delivered. c
  lrat c; num of c←a*den of b-num of b;
  den of c←den of b;
  lrat←% c
end;

op * = (lrat a, lint b)lrat:
begin
c Multiplication with mixed modes is performed and the resulting
lrat is delivered. c
  lrat c; lint il;
  il←hcf(abs b, abs (den of a));
  (not (il=one) | num of c←num of a*(b÷il);
                  den of c←den of a÷il |
                  num of c←num of a*b; den of c←den of a);
```

```
  lrat←c
end;

op * = (lint a, lrat b)lrat:
begin
c Multiplication with mixed modes is performed and the resulting
lrat is delivered. c
  lrat c; lint il;
  il←hcf (abs a, abs (den of b));
  (not (il=one) | num of c←num of b*(a÷il);
                  den of c←den of b÷il  |
                  num of c←num of b*a; den of c←den of b);
  lrat←c
end;

op / = (lint a, lrat b) lrat:
begin
c  Division with mixed modes is performed and the resulting
lrat is delivered. c
  lrat c; lint il;
  il←hcf (abs a, abs (num of b));
  (not (il=one) | num of c←den of b*(a÷il);
                  den of c←num of b÷il |
                  num of c←den of b*a; den of c←num of b);
  lrat←c
end;

op / = (lrat a, lint b)lrat:
begin
c Division with mixed modes is performed and the resulting
lrat is delivered. c
```

```
  lrat c; lint i1;
  i1←hcf (abs b, abs (num of a));
  (not (i1=one) | num of c←num of a÷i1;
                  den of c←den of a*(b÷i1) |
                  num of c←num of a; den of c←den of a*b);
  lrat←c
end;

op abs = (lrat a)lrat:
begin
c delivers the absolute value of lrat a c
  lrat c←a;
  si of num of c←false;
  si of den of c←false;
  lrat←c
end;
```

REFERENCES

ABRAMOWITZ M. and STEGUN I.A. (1965). Handbook of Mathematical functions, *New York: Dover Publication Inc.*

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

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

ALAYLIOGLU A., EVANS G.A. and HYSLOP J. (1974b). The use of Chebyshev series for the evaluation of oscillatory integrals, *Mathematics Research No.41, Loughborough University.*

ALAYLIOGLU A., EVANS G.A. and HYSLOP J. (in press). Automatic generation of quadrature formulae for oscillatory integrals. *The Computer Journal.*

BAKHVALOV N.S. and VASIL'EVA L.G. (1968). Evaluation of the integrals of oscillating functions by interpolation at nodes of Gaussian quadratures, *Zh. Vychisl. Mat.mat.Fiz.* Vol. 8, part 1, p.241-249.

BALBINE G. and FRANKLIN J.N. (1966). The calculation of Fourier integrals, *Math. Comp.*, Vol.20, p.570-589.

BIRKOFF G. and MACLANE S. (1963). A Survey of Modern Algebra. *New York: MacMillan.*

BLAKEMORE M., EVANS G.A. and HYSLOP J. (1974). The evaluation of two-centre molecular integrals involving one-electron Green's functions. *Mathematics Research No.40, Loughborough University.*

CHISHOLM J.S.R., GENZ A. and ROWLANDS G.E. (1972). Accelerated convergence of sequences, *J. Comput. Phys.* Vol.10, p.284-307.

CLENDENIN W.W. (1966) A method for numerical calculation of Fourier integrals, *Numer. Math.*, Vol.8, p.422-436.

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

CONTE S.D. (1965). Elementary Numerical Analysis, *New York: McGraw Hill.*

COULSON C.A. and MCWEENEY R. (1949). The computation of wave functions in momentum space - I : The Helium atom, *Proc. Phys. Soc.*, A62, p.509-519.

DAVIS P.J. (1963). Interpolation and Approximation, *New York: Blaisdell.*

DAVIS P.J. and RABINOWITZ P. (1965). Ignoring the singularity in approximate integration, *SIAM J. Numer. Anal.*, Vol.2, p.367-383.

DAVIS P.J. and RABINOWITZ P. (1967). Numerical Integration, *New York: Blaisdell.*

ELLIOTT D. (1965). Truncation errors in two Chebyshev series approximations, *Math. Comp.*, Vol.19, p.234-248.

EVANS G.A. (1969). Some problems in low Reynolds number flow, *D.Phil. Thesis, Oxford.*

EVANS G.A. and OCKENDON J. (1972). The drag on a sphere in low Reynolds number flow, *J. Aero. Sci.*, Vol.3, p.237-242.

EVANS G.A. (1973). The forces on a small sphere due to the interactive inertia effects of a neighbouring sphere, *Mathematics Research No.21, Loughborough University.*

EYRING H., WALTER J. and KIMBALL G.E. (1944). Quantum Chemistry, *New York: Wiley.*

FADDEEV D.K. and SOMINSKII I.S. (1965). Problems in Higher Algebra, *San Francisco: Freeman.*

FILON L.N.G. (1928). On a quadrature formula for trigonometric integrals, *Proc. Roy. Soc. Edinburgh*, Vol.49, p.38-47.

FLINN E.A. (1960). A modification of Filon's method for numerical integration, *J. Assoc. Comput. Mach.*, Vol.7, p.181-184.

FOX L. (1964). An Introduction to Numerical Linear Algebra, *London: Oxford University Press.*

FRASER W. and WILSON M.W. (1966). Remarks on the Clenshaw-Curtis quadrature scheme, *SIAM Review*, Vol.8, p.322-327.

GAUTSCHI W. (1968). Construction of Gauss-Christoffel quadrature formulas, *Math. Comp.*, Vol.22, p.251-270.

GENTLEMAN W.M. (1972). Implementing Clenshaw-Curtis quadrature, I methodology and experience, *CACM*, Vol.15, p.337-342.

GRADSHTEYN I.S. and RYZHIK I.M. (1965). Table of Integrals, Series and Products, *New York: Academic Press.*

GOLUB G.H. and WELSCH J.H. (1969). Calculation of Gauss quadrature rules, *Math. Comp.*, Vol.23, p.221-230.

HALL G.G. (1967). A variation principle for discontinuous wave functions, *Chem. Phys. Letters*, Vol.1, p.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, p.195-204.

HALL G.G., HYSLOP J. and REES D. (1970). A minimum principle for molecular systems allowing the use of discontinuous wave functions, *Intern. J. Quantum Chem.*, Vol.4, p.5-20.

HARRIS F.E. and MICHELS H.H. (1967). The evaluation of molecular integrals for **Slater**-type orbitals, *Adv. in Chem. Phys.*, Vol.13, p.205-265.

HAMMING R.W. (1973). Numerical Methods for Scientists and Engineers, *New York: McGraw-Hill.*

HILDEBRAND F. (1956). Introduction to Numerical Analysis, *New York: McGraw-Hill.*

HURWITZ H. and ZWEIFEL P.F. (1956). Numerical quadrature of Fourier transform integrals, *MTAC*, Vol.10, p.140-149.

HURWITZ H., PFEIFER R.A. and ZWEIFEL P.F. (1959). Numerical quadrature of Fourier transform integrals II, *MTAC*, Vol.13, p.87-90.

HYSLOP J. (1972) Private Communications.

HYSLOP J. (1973) A six-dimensional quadrature procedure, *Theoretical Chim. Acta*, Vol.31, p.189-194.

KOPAL Z. (1961). Numerical Analysis, *New York: Wiley.*

KRYLOV V.I. (1962). Approximate Calculation of Integrals, *New York: MacMillan.*

LANGLOIS W.E. (1964). Slow Viscous Flow, *New York: MacMillan.*

LEVIN D. (1973). Development of non-linear transformations for improving convergence of sequences, *Intern. J. Computer Math. Section B,* Vol.3, p.371-388.

LONGMAN I.M. (1960). A method for the numerical evaluation of finite integrals of oscillatory functions, *Math. Comp.,* Vol.14, p.53-59.

LUKE Y.L. (1954). On the computation of oscillatory integrals, *Proc. Camb. Phil. Soc.,* Vol.50, p.269-277.

McWEENEY R. (1949). The computation of wave functions in momentum space - II: The Hydrogen molecule ion, *Proc. Phys. Soc.,* A62, p.519-528.

MORSE P.M. and FESHBACH H. (1953). Methods of Theoretical Physics, *New York: McGraw-Hill.*

MOTT N.F. and SNEDDON I.N. (1948). Wave Mechanics and Its Applications, *Oxford University Press.*

O'HARA H. and SMITH F.J. (1968). Error estimation in the Clenshaw-Curtis quadrature formula, *The Computer Journal*, Vol.11, p.213-219.

PIESSENS R. (1970). Gaussian quadrature formulas for the integration of oscillating functions, *ZAMM*, Vol.50, p.698-700.

PIESSENS R. and HAEGEMANS A. (1973). Numerical calculation of Fourier transform integrals, *Electron. Lett.*, Vol.9, No.5, p.108-109.

PIESSENS R. and POLEUNIS F. (1971). A numerical method for the integration of oscillatory functions, *BIT*, Vol.11, p.317-327.

RABINOWITZ P. (1967). Gaussian integration in the presence ofasingularity, *SIAM J. Numer. Anal.*, Vol.4, p.191-201.

ROBINSON P.D. and EPSTEIN S.T. (1970). On energy bounds from the conjugate eigenvalue problem, *Intern. J. Quantum Chem.*, Vol.4, p.453-463.

ROBINSON P.D., WARNOCK T.T. and ANDERSON N. (1970). Appraisal of an iterative method for bound states, *Physical Rev. A*, Vol.1, p.1314-1320.

ROMBERG W. (1955). Vereinfacte numerische integration, *Det. Kong. Norske Vid. Sel. Forh.*, Vol.28, p.30-36.

ROSENHEAD L. (1963). Laminar Boundary Layers, *Oxford.*

RUTISHAUSER H. (1962). On a modification of the QD-algorithm with Graeffe-type convergence. *Z.A.M.P.*, Vol.13, p.493-496.

SACKS R.G. (1953). Nuclear Theory, *Cambridge: Addison-Wesley.*

SACK R.A. and DONOVAN A.F. (1972). An algorithm for Gaussian quadrature given modified moments, *Numer. Math.*, Vol.18, p.465-478.

SAENGER A. (1964). On a numerical quadrature of Fourier transforms, *J. Math. Anal. Appl.*, Vol.8, p.1-3.

SAFFMAN P.G. (1964). The lift on a small sphere in slow shear flow, *J. Fluid Mech.*, Vol. 22, p.385-400.

SCHEID F. (1968). Theory and Problems of Numerical Analysis, *New York: McGraw-Hill.*

SCHLICHTING H. (1960). Boundary Layer Theory, *New York: McGraw-Hill.*

SHANKS D. (1955). Nonlinear transformations of divergent and slowly convergent sequences, *J. Math. Phys.*, Vol.34, p.1-42.

SQUIRE W. (1970). Integration for Engineers and Scientists, *New York: Elsevier.*

SQUIRE W. (1973). Comment on numerical calculation of Fourier-transform integrals, *Electron. Lett.*, Vol.9, No.13, p.291.

STIEFEL E. (1961). Einführung in die Numerische Mathematik, *Stuttgart: Teubner.*

STOKES G.G. (1851). On the effect of internal friction of fluids on the motion of pendulums, *Camb. Phil. Trans.*, Vol.9, p.8-106.

STROUD A.H. and SECREST D. (1966). Gaussian Quadrature Formulas, *Englewood Cliffs N.J.: Prentice Hall Inc.*

TUCK E.O. (1967). A simple 'Filon-Trapezoidal' rule, *Math. Comp.*, Vol.21, p.239-241.

VAN DYKE M. (1964). Perturbation Methods in Fluid Mechanics, *New York: Academic Press.*

WYNN P. (1956). On a device for computing the $e_m(s_n)$ transformation, *Maths. Comput.*, Vol.10, p.91-96.