Thermochimica Acta, 44 (1981) 313—323
Elsevier Scientific Publishing Company, Amsterdam — Printed in Belgium

THE EVALUATION OF THE ARRHENIUS INTEGRAL WITH SPECIAL REFERENCE TO SMALL COMPUTERS

A.C. NORRIS and A.A. PROUDFOOT

Department of Chemistry, Portsmouth Polytechnic, Portsmouth (Gt. Britain)

(Received 10 September 1980)

ABSTRACT

This paper discusses the factors which influence the choice and implementation of computer methods to evaluate the Arrhenius integral, $I_A = \int_0^{T_\alpha} \exp(-E/RT)dT$. It also identifies the sources of computational error and inefficiency. It is shown that, amongst numerical integration techniques, the classical trapezoidal and Simpson rules have little to recommend them compared with the method of Gaussian quadrature. A technique for preserving the accuracy of the Gauss method at very high values of E/RT_α is also described and evaluated. Rational (Padé) approximation is found to compare favourably with Gaussian quadrature in efficiency and accuracy, and is simpler to use. The discussion also reveals that six decimal-digit arithmetic is adequate for all practical purposes.

INTRODUCTION

The Arrhenius integral, I_A, is important in solid-state kinetics because of its frequent occurrence in the analysis of non-isothermal kinetic data [1—5]. The integral can be written as

$$I_A = \int_0^{T_\alpha} \exp(-E/RT)\, dT \qquad (1)$$

where E is the activation energy, R is the gas constant (8.314 J mole^{-1} K^{-1}) and T_α is the thermodynamic temperature at which the extent of reaction is α. The form of I_A means that the definite integral has no analytical solution and it must therefore be evaluated using a numerical method. Methods reported in the literature include the composite trapezoidal rule [3], Gauss quadrature [1] and rational approximation [4].

The features required of any numerical method for estimating I_A are firstly, high accuracy; in particular, the error introduced by the computational method must be negligible compared with the experimental error, and secondly, efficient use of computer time; this is particularly important in real-time applications.

The choice of method usually involves a compromise between these criteria. Although several studies [1—5] have discussed the evaluation of I_A, the nature of this compromise has received little attention, primarily because it

0040-6031/81/0000—0000/$02.50

is seldom a limiting factor for mainframe computers on which most work has been carried out to date. It can, however, be of considerable importance with mini- or microcomputers, which are becoming increasingly common in both research and routine analytical laboratories.

This paper considers the factors that influence the choice and implementation of numerical methods used to estimate I_A. The results are derived mainly for small computers, but they are also relevant to the evaluation of I_A on larger machines.

COMPUTER EVALUATION OF THE ARRHENIUS INTEGRAL

As eqn. (1) shows, the Arrhenius integral is the area under the curve $f(T) = \exp(-E/RT)$ between the bounds $T = 0$ and $T = T_\alpha$. A closed numerical integration method approximates this area by a weighted sum, I_{num}, of the function values, $f(T_i)$, as shown in eqn. (2)

$$I_{num} = \sum_{i=0}^{n} w_i f(T_i) , \quad 0 \leqslant T_i \leqslant T_\alpha \tag{2}$$

In this equation, the weight factors, w_i, are determined by the method, and the function values, $f(T_i)$, are computed at each of $n + 1$ base points, T_i.

When implemented on a computer, all calculations involved in the estimation of I_{num} are carried out in floating-point arithmetic [6] in which a real number, r, i.e. a number with a fractional part (which can be zero) is represented as

$$r = a\beta^b \tag{3}$$

In eqn. (3), a is the fractional part of the number, normalised so that the first digit is non-zero, β is the number base of the computer and b is the exponent. On a machine using binary arithmetic, $\beta = 2$ and $0.5 \leqslant |a| < 1$. The floating-point representation is implemented as a sequence of binary digits (bits) which is divided into two parts to hold the fraction and the exponent. Since, however, the sequence of bits is of finite length, the fraction can only be held to a finite number of significant digits and the exponent can only vary over a finite range. The limited number of significant digits in the fraction is the origin of the build-up of round-off error in floating-point calculations while the exponent, operating in conjunction with the number base of the machine, determines the number range. Typically, the range of real numbers which can be represented on a large, mainframe computer is $5.4 \times 10^{-79} < |r| < 7.2 \times 10^{75}$ and the precision to which these numbers are held is 7D (seven, significant decimal digits) for single-precision arithmetic and 16D for double-precision arithmetic. In contrast, minicomputers typically have a real number range of $1.5 \times 10^{-39} < |r| < 1.7 \times 10^{38}$ with 6D in single-precision and, depending on the machine, 9—16D in double-precision working.

Since numerical integration methods evaluate the function $f(T_i) = \exp(-E/RT_i)$ over the integration range, $[0, T_\alpha]$, some of the evaluations at the

lower end of the integration range can generate numbers that are too small to be represented by the computer and are thus indistinguishable from zero. On many machines, this "exponent underflow" results in the unreported setting of the function evaluations to zero. This is potentially a major source of error since the smaller T_α becomes (for any particular activation energy), the greater is the proportion of function evaluations set to zero. The conditions under which this error becomes troublesome are discussed in the next section.

Thermogravimetric data from a single experiment, coupled with the evaluation of I_A, typically yield an activation energy with a maximum error of about ±10%. To ensure that computational error makes no significant contribution to this figure, the calculation errors should therefore be no more than about 0.01%. With 6D arithmetic, the machine error is well below this figure at $\simeq 10^{-4}$% and the desired accuracy can therefore be achieved provided the numerical method and its implementation are chosen carefully.

COMPARISON OF METHODS USED FOR THE EVALUATION OF I_A

In the results reported here, values of I_A have been estimated for physically-reasonable values of E, from 100 to 350 kJ mole^{-1}, and temperatures from 300 to 1100 K. These limits correspond to the range $11 < E/RT_\alpha < 140$.

The integrals were determined by different numerical methods on a Texas Instruments 980B minicomputer which has 6D in single-precision arithmetic and a range of $1.5 \times 10^{-39} < |r| < 1.7 \times 10^{38}$. Reference values of I_A were obtained from standard tables [7] and, where necessary, by using a 32-point Gauss quadrature formula (see later) with double-precision arithmetic on an ICL 2960 mainframe computer. All reference values are accurate to at least 9D. The computer programs were written in FORTRAN IV.

In the calculation of I_A, the total relative error of integration is computed as a percentage from the relationship

$$\epsilon_{total} = 100\ (I_{ref} - I_{num})/I_{ref}$$

where I_{ref} is the reference value and I_{num} is the corresponding numerical estimate of a given value of I_A.

The discussion so far identifies round-off and underflow errors as the cause of discrepancies between I_{ref} and I_{num}. We must now consider the error contributed by the numerical method itself.

Newton—Cotes methods

The most common numerical integration techniques belong to the class known as composite, Newton—Cotes methods [8]. The essential features of these methods are the division of the integration range into a number of constant-width subintervals, and the piecewise approximation of the original function over each subinterval, or group of subintervals, by a polynomial of low degree. If the degree is one, the approximating function is a straight line

and the procedure is known as the composite trapezoidal rule. When the degree is two, the integrand is approximated over each pair of subintervals by a parabola to give the composite Simpson rule. In either case, replacement of the original function introduces an approximation or truncation error, which makes a third contribution to the total integration error.

The Newton—Cotes methods have the advantage that they generate straightforward integration formulae with simple weight factors. Thus, for the trapezoidal and Simpson rules, the numerical integrals $I_{\text{num,T}}$ and $I_{\text{num,S}}$ that approximate I_{A} are

$$I_{\text{num,T}} = h[\tfrac{1}{2}\{\text{f}(0) + \text{f}(T_\alpha)\} + \sum_{i=1}^{n-1} \text{f}(T_i)] \tag{4}$$

$$I_{\text{num,S}} = (h/3)[\text{f}(0) + \text{f}(T_\alpha) + \sum_{i=1}^{n-1} a_i \text{f}(T_i)] \tag{5}$$

where f(0) (note that f(0) = 0) and f(T_α) are the function values at the endpoints of the interval, the subinterval width, $h = \text{f}(T_{i+1}) - \text{f}(T_i)$, is the separation between consecutive base points, and $n = T_\alpha/h$ is the number of subintervals. The weight factors, w_i, of eqn. (2) are seen to be functions of h and, in eqn. (5), the coefficient a_i is 4 when i is odd and 2 when i is even.

The truncation errors are also defined by comparatively simple expressions [8] and, for integration with n subintervals over the range $[0, T_\alpha]$, the percentage errors are

$$\epsilon_{\text{trun,T}} \simeq -100\ (h^2/12)\ \text{f}'(T_\alpha)/I_{\text{ref}}$$

$$\epsilon_{\text{trun,S}} \simeq -100\ (h^4/180)\ \text{f}'''(T_\alpha)/I_{\text{ref}}$$

where $\text{f}'(T_\alpha)$ and $\text{f}'''(T_\alpha)$ are the first- and third-order derivatives of the function $\text{f}(T) = \exp(-E/RT)$ evaluated at T_α. Clearly, in both cases, truncation error can be decreased in magnitude by increasing the number of subintervals, n, and thereby decreasing the subinterval width, h. Since, however, $|\epsilon_{\text{trun,S}}|$ is proportional to h^4 whereas $|\epsilon_{\text{trun.T}}|$ is proportional to h^2, the truncation error of the Simpson rule decreases much more rapidly.

Unfortunately, the benefit of the simplicity of these methods is offset by the comparatively large number of subintervals needed to reduce truncation error to an acceptable level. Not only are the necessary calculations very time-consuming but they increase the upper limit on round-off error [8], which is roughly proportional to $1/h$, and the bound on a fourth error component, exponential algorithm error [9], which arises from the computer routine used to evaluate $\exp(-E/RT)$. A theoretical analysis of how these various errors combine would be difficult and of limited value since it would depend upon the numerical method, the computer, and the system's software. We can observe, however, that the combined error is dominated by truncation effects at low n and by computational errors at high n, causing the magnitude of the total error to go through a minimum as n increases. Because of the rapid decrease of $|\epsilon_{\text{trun,S}}|$ noted earlier, this minimum is smaller, and occurs at lower n, for the Simpson method than for the trapezoidal rule.

These points are illustrated for the evaluation of I_A by the results presented in Table 1. The data were obtained on the TI980B machine with 6D accuracy using $E = 200$ kJ mole^{-1} and $E/RT_\alpha = 60$ ($T_\alpha \simeq 400$ K). Under these conditions, I_A is approximately 5.67×10^{-26}. For both methods, truncation error is negative and is clearly the major contribution to the total error at low n. As n increases, however, the round-off and exponential algorithm errors become relatively more important and, at high n, the total error fluctuates within the approximate range $\pm 6 \times 10^{-4}$%. Simpson's rule takes only 500 subintervals to reach this level whereas the trapezoidal method needs over 4000 subintervals. In both cases, there is no advantage in increasing n beyond the values stated, and in fact the acceptable level of 0.01% error is realised with ~150 Simpson, and ~2000 trapezoidal subintervals, respectively.

In this discussion we have ignored the effect of underflow error which occurs when the value of I_A approaches s, the smallest positive number that the computer can distinguish from zero. This error, which has not previously been reported in this context, allows the return of a non-zero function value f(T) only when

$$f(T) = \exp(-E/RT) \geqslant s$$

i.e. when $E/RT \leqslant -\ln s$ ($\ln s$ is of course a negative quantity). Equivalently, for a given value of E there exists a threshold temperature, T_t given by

$$T_t = -E/R \ln s \qquad (6)$$

below which all function evaluations are set to zero. For large n (e.g. $n \geqslant 500$), the percentage, p, of function evaluations set to zero by underflow error over the range $[0, T_\alpha]$ is approximated well by the expression

$$p \simeq 100\, T_t/T_\alpha \qquad (7)$$

As indicated above, a typical minicomputer has $s \simeq 1.5 \times 10^{-39}$ and so $-\ln s \simeq 89.4$ giving $T_t \simeq E/89.4R$ and

$$p \simeq 1.12\, E/RT_\alpha$$

Values of p calculated from this last equation for feasible values of E and T_α are shown in Table 2.

The percentages are surprisingly large under all conditions. For example, the values of E/RT_α in Table 2 range from 11 to 140 and $p > 50$% whenever $E/RT_\alpha > 44.7$. The most obvious consequence of exponential underflow is therefore the highly inefficient use of computer time and this is all the more important with the Newton—Cotes methods which need so many function evaluations to produce acceptable accuracy.

We must now examine the contribution of the underflow error to the total error in I_A. Taking $E = 200$ kJ mole^{-1} and $T_\alpha = 1000$ K ($p = 26.9$%), we find $f(T_\alpha) \simeq 3.6 \times 10^{-11}$ whereas the first non-zero function evaluation involved in the integral summation is close to the negligible quantity $s \simeq 1.5 \times 10^{-39}$. Naturally, the importance of underflow error increases as the upper limit T_α approaches the threshold temperature T_t. To reveal the extent of the problem, I_A has been calculated with $E = 200$ kJ mole^{-1} and

TABLE 1

Effect of number of subintervals, n, on percentage error, ϵ, in the evaluation of I_A by Newton—Cotes methods (E = 200 kJ mole^{-1}; E/RT_α = 60)

n	$\epsilon_{trun,T}$	$\epsilon_{total,T}$	$\epsilon_{trun,S}$	$\epsilon_{total,S}$
150	-1.38	-1.37	-1.32×10^{-2}	-1.29×10^{-2}
250	-4.96×10^{-1}	-4.95×10^{-1}	-1.72×10^{-3}	-1.26×10^{-3}
500	-1.24×10^{-1}	-1.23×10^{-1}	-1.07×10^{-4}	3.24×10^{-4}
1000	-3.10×10^{-2}	-3.06×10^{-2}	-6.71×10^{-6}	3.24×10^{-4}
2000	-7.75×10^{-3}	-7.27×10^{-3}	-4.19×10^{-7}	5.01×10^{-4}
4000	-1.94×10^{-3}	-1.44×10^{-3}	-2.62×10^{-8}	5.01×10^{-4}
10000	-3.10×10^{-4}	-5.58×10^{-4}	-6.71×10^{-10}	-2.05×10^{-4}
20000	-7.75×10^{-5}	-3.82×10^{-4}	-4.19×10^{-11}	-2.05×10^{-4}

E/RT_α = 85 (p = 95%) using 6D arithmetic and Simpson's rule with 500 subintervals. The numerical integral is $I_{num} = 3.902 \times 10^{-37}$ whereas the reference value is $I_{ref} = 3.957 \times 10^{-37}$. Although the integral is very small in magnitude, the error in I_A of 1.4% caused by underflow is unacceptably large compared with the errors discussed previously and may cause difficulties in subsequent calculations with the computed estimate.

Because of the extended number range, underflow error is unlikely to cause any serious trouble on mainframe computers for which frequently $-\ln s = 180.2$ and $p \simeq 0.55$ E/RT$_\alpha$. Thus, to obtain p = 95% it is necessary to have $E/RT_\alpha \simeq 169$, a ratio which is physically quite unrealistic.

The impact of underflow on the efficiency and accuracy of small machine calculations can be reduced very simply but, in view of the comparatively high truncation error and low computational efficiency of Newton—Cotes methods, it seems better to reserve these improvements for a more effective technique.

Gauss quadrature

Whereas the composite Newton—Cotes methods use piecewise polynomial approximation, the technique of Gauss quadrature [8] uses a single approx-

TABLE 2

Percentage, p, of function evaluations set to zero for the Newton—Cotes methods ($-\ln s$ = 89.4)

T_α(K)	E(kJ mole^{-1})					
	100	150	200	250	300	350
300	44.8	67.3	89.7	100	100	100
500	26.9	40.4	53.8	67.3	80.7	94.2
700	19.2	28.8	38.4	48.0	57.6	67.3
900	15.0	22.4	29.9	37.4	44.8	52.3
1100	12.2	18.3	24.5	30.6	36.7	42.8

imating polynomial over the entire integration range. To do this, the Gauss method dispenses with the constant subinterval width imposed by the composite methods and positions the base points so that the summation of eqn. (2) minimises the truncation error. To implement the procedure, the integration range for I_A must be shifted from [0, T_α] to [—1, 1] by the simple change of variable

$$z = 2T/T_\alpha - 1 \quad \text{or} \quad T = (T_\alpha/2)(z + 1) \tag{8}$$

With this transformation, $dT = (T_\alpha/2)\,dz$, and the integral becomes

$$I_A = \int_0^{T_\alpha} f(T)\,dT = (T_\alpha/2) \int_{-1}^{1} g(z)\,dz$$

where

$$\begin{aligned} g(z) &= \exp[-2E/RT_\alpha(z + 1)] \\ &= \exp(-x) \\ &\equiv \exp(-E/RT) = f(T) \end{aligned} \tag{9}$$

The summation of eqn. (2) for a polynomial of degree m (m + 1 base points) then becomes

$$I_{num} = (T_\alpha/2) \sum_{i=0}^{m} w_i g(z_i)\,, \quad -1 < z_i < 1 \tag{10}$$

The weight factors and (16) base points for the test polynomial of degree m = 15 used in this work are given in Table 3. Data for (Gauss—Legendre) polynomials of degree up to 95 are tabulated in ref. 7.

As this description shows, the disadvantages of the Gauss approach are the need to transform the integration range, and the awkwardness of the weight factors and base points which renders the technique unsuitable for hand calculation. A further difficulty is the complex analysis of truncation error [8]. However, for computer implementation, these constraints are far outweighed by the potentially very high accuracy of the method. In fact, the number of function evaluations needed to reduce truncation error to an acceptable level is so small that the summation of eqn. (10) is extremely rapid and suffers negligible contamination from the build-up of round-off error.

TABLE 3

Weight factors, w_i, and base points, z_i, for a Gauss quadrature polynomial of degree m = 15

w_i	$\pm z_i$	w_i	$\pm z_i$
0.189451	0.095013	0.124629	0.755404
0.182603	0.281604	0.095159	0.865631
0.169157	0.458017	0.062254	0.944575
0.149596	0.617876	0.027152	0.989401

These points are demonstrated (Table 4) by computing I_A with 6D arithmetic and a fixed polynomial degree, $m = 15$, i.e. 16 function evaluations. With $E = 200$ kJ mole^{-1} and values of E/RT_α from 10 to 80 ($6.62 \times 10^{-35} < I_A < 9.21 \times 10^{-3}$) the total error is usually comparable with the machine's minimum error and is lower in magnitude than the error found with 250 Simpson subintervals. When these calculations are repeated with 9D arithmetic, the error peaks observed at $E/RT_\alpha = 40$ and 80 are reduced significantly, suggesting that the high values are due mainly to exponential error. The higher-precision results show an increase in total error above the round-off level as E/RT_α increases, and this trend is probably due to truncation effects.

Although the Gauss procedure is faster and generally more accurate than the Newton—Cotes methods, it is still susceptible to inefficiency and error due to underflow when $E/RT_\alpha \simeq -\ln s$. Thus, the function value f(z) will be set to zero for all z below a threshold value obtained from eqns. (6) and (8) as

$$z_t = 2T_t/T_\alpha - 1 = -2E/(RT_\alpha \ln s) - 1$$

For several reasons, however, it is not possible to use this value of z_t to write an expression analogous to eqn. (7) for p, the percentage of function evaluations set to zero. Firstly, the number of function evaluations is too small to make the approximation used to derive eqn. (7) valid for Gauss quadrature. Secondly, the value of p depends upon the polynomial degree, m, which determines the values of z at which f(z) is evaluated and, finally, small fractional weight factors can make the product $w_i g(z_i) < s$.

To indicate the potentially disastrous nature of underflow we attempt to calculate I_A with $m = 15$, $E = 200$ kJ mole^{-1} and $E/RT_\alpha = 85$ on a machine having $-\ln s \simeq 89.4$. Only the two largest values of z_i shown in Table 3 give $g(z_i) > s$ and the smaller of these gives a quantity less than s when multiplied by the appropriate weight factor. Thus, only one of the function evaluations returns a non-zero value and the estimated integral of 2.971×10^{-37} differs from the reference value, 3.957×10^{-37}, by 24.9%! This quite unacceptable error is reduced drastically by multiplying the function values, $g(z_i)$, in eqn. (10) not by w_i, but by the product $w_i T_\alpha/2$. This simple inclusion of the constant $T_\alpha/2$ within the summation, normally regarded as a highly inefficient

TABLE 4

Effect of polynomial degree on percentage error, ϵ, in the evaluation of I_A by Gauss quadrature (G) and rational approximation (R) ($E = 200$ kJ mole^{-1})

E/RT_α	$\epsilon_{total,G15}$ $m = 15$	$\epsilon_{total,R3}$ $m = 3$	$\epsilon_{total,R4}$ $m = 4$	$\epsilon_{total,R5}$ $m = 5$
10	2.55×10^{-4}	1.67×10^{-3}	1.47×10^{-4}	3.8×10^{-5}
20	6.34×10^{-4}	6.34×10^{-4}	-2.50×10^{-4}	-2.50×10^{-4}
40	1.00×10^{-3}	3.44×10^{-4}	3.44×10^{-4}	3.44×10^{-4}
60	3.24×10^{-4}	3.24×10^{-4}	3.24×10^{-4}	3.24×10^{-4}
80	9.87×10^{-3}	5.07×10^{-4}	5.07×10^{-4}	5.07×10^{-4}

procedure, retrieves the second non-zero function evaluation and reduces the observed error to 0.94%.

Although this reduction is considerable, the remaining error is still not acceptable if I_A is to be used in subsequent calculations. The effect of underflow error can be reduced further, however, simply by adding a positive quantity, y, to the index of the exponential function in eqn. (9) so that $\exp(y-x)$ is evaluated instead of $\exp(-x)$ for all $x > 85$. When the modified summations are complete, the total is multiplied by $\exp(-y)$ before adding to the unaltered sum obtained with $x \leqslant 85$. This device is naturally only of value if the modified sum is greater than or equal to $\exp(y)$s. This criterion is satisfied for the test integration and the inclusion of the device in the algorithm reduces the total error to an acceptable level of 1.09×10^{-3}%. The same device can be used to extend artificially the number range of a mainframe computer but the available number range is usually sufficient to render the extension unnecessary.

The results described in this section show that the modified Gauss method of degree 15 is always preferable to the Newton—Cotes methods for the evaluation of I_A.

Rational approximation

A rational approximation is a ratio of polynomial functions [10] of the type

$$R_{mk}(x) = P_m(x)/Q_k(x)$$

where R_{mk} is an approximation to a function, f(x), and $P_m(x)$ and $Q_k(x)$ are polynomials of degree at most m and k, respectively. Such approximations are used extensively in computer algorithms [9] to calculate transcendental functions such as $\cos(x)$, $\exp(x)$, $\log(x)$, etc. The incentive behind their use to evaluate I_A springs from the ease with which the Arrhenius integral can be expressed in terms of the exponential integral, $E_1(x)$, for which there are many alternative rational approximations [11].

I_A is related to E_1 by making the linear change of variable $x = E/RT$ in eqn. (1). Noting that $dT/dx = -E/Rx^2$, we have

$$I_A = (E/R) \int_{x_\alpha}^{\infty} x^{-2} \exp(-x)\, dx$$

where $x_\alpha = E/RT_\alpha$. Integration by parts then yields

$$I_A = (E/R)[\exp(-x_\alpha)/x_\alpha - E_1(x_\alpha)]$$

which involves the exponential integral

$$E_1(x_\alpha) = \int_{x_\alpha}^{\infty} x^{-1} \exp(-x)\, dx$$

This approach was developed by Senum and Yang [4] who studied the truncation error involved in replacing $E_1(x_\alpha)$ by a series of Padé approxima-

tions [10,11], $[\exp(-x_\alpha)/x_\alpha]\, R_{mk}(x_\alpha)$, in which the polynomial degrees m and k were equal. For example, with $m = k = 2$

$$E_1(x_\alpha) = [\exp(-x_\alpha)/x_\alpha]\, R_{22}(x_\alpha)$$
$$= [\exp(-x_\alpha)/x_\alpha]\,(x_\alpha^2 + 5x_\alpha + 2)/(x_\alpha^2 + 6x_\alpha + 6)$$

and the corresponding approximation to the Arrhenius integral is

$$I_A = T_\alpha \exp(-x_\alpha)[1 - R_{22}(x_\alpha)]$$
$$= [T_\alpha \exp(-x_\alpha)]\,(x_\alpha + 4)/(x_\alpha^2 + 6x_\alpha + 6) \qquad (11)$$

Luke [11] tabulates the polynomial coefficients of $R_{mk}(x_\alpha)$ for values of m and k from 1 to 6 and it is worth noting that the approximation errors decrease as m and k increase and are smallest, for a given total degree $m + k$, when $m = k$ or $k + 1$. As eqn. (11) also shows, rational approximation involves negligible build-up of round-off error, no build-up of exponential algorithm error and no underflow error at all (until total underflow occurs). This is because there is only one exponential function evaluation and the number of arithmetic operations is very small. The values of total percentage error obtained with rational approximations of degrees $m = k = 3$, 4 and 5 are shown in Table 4 together with the Gauss quadrature results discussed earlier. At high values of E/RT_α ($= x_\alpha$), the third-degree rational approximation to $E_1(x_\alpha)$ yields estimates of I_A which are comparable in accuracy to the Gaussian values. At lower values of E/RT_α, the rational approximation is much less accurate and the overall degree of the approximation must be increased to obtain acceptable truncation error. The value of $\epsilon_{total,Rm}$ increases with decreasing E/RT_α for all values of m studied but when $m = 4$, the total error is comparable with machine error over the whole range of E/RT_α. Recalculation with higher-precision arithmetic reduces the errors obtained to the levels recorded by Senum and Yang.

These results reveal that a rational approximation of degree $m = 4$ for I_A is comparable with or superior to the Gauss method in accuracy and is computationally much simpler. In fact the calculation is so straightforward that it can be implemented with ease on a hand calculator. Finally, if the value of E/RT_α is close to $-\ln s$, the remaining problem of finite number range may be avoided entirely by taking the logarithm of the rational approximation $R_{mk}(x_\alpha)$ and using

$$\ln I_A = \ln\{T_\alpha[1 - R_{mk}(x_\alpha)]\} - x_\alpha$$

in subsequent calculations.

CONCLUSIONS AND RECOMMENDATIONS

The work described here produces several clear-cut recommendations for computer evaluation of the Arrhenius integral, I_A. The results show that the classical trapezoidal and Simpson methods of numerical integration have nothing to recommend them beyond simplicity. In contrast, Gauss quadrature is a very rapid method and a 16-point formula reduces total integration

error to $\leqslant 10^{-3}\%$ under all practical conditions if evaluation of the exponential function is modified for high E/RT_α. However, the inefficiency and underflow error to which these numerical integration methods are susceptible are avoided by the use of a rational approximation technique which is also simpler and more accurate. Accordingly, a Padé approximation of degree four is recommended for calculation of I_A when $E/RT_\alpha > 10$. At lower values of E/RT_α, a Gauss procedure or a rational approximation of higher degree should be used.

Most mini- and mainframe computers carry out single-precision calculations with six or seven digit accuracy, i.e. an arithmetic operation has a relative error of magnitude $\leqslant 10^{-4}\%$. This level of error is quite satisfactory for most applications involving the computation of I_A and so the Gauss and rational methods can be implemented without recourse to double-precision arithmetic.

FORTRAN function subprograms implementing the Gauss and rational methods are available from the authors on request.

ACKNOWLEDGEMENT

The authors would like to thank Dr. M.I. Pope for helpful discussions.

REFERENCES

1 G. Gyulai and E.J. Greenhow, J. Therm. Anal., 6 (1974) 279.
2 V.M. Gorbachev, J. Therm. Anal., 8 (1975) 349.
3 W.Y. Wen, J. Therm. Anal., 10 (1976) 315.
4 G.I. Senum and R.T. Yang, J. Therm. Anal., 11 (1977) 445.
5 M.A. Brown and C.A.R. Philpotts, J. Chem. Educ., 55 (1978) 556.
6 L.F. Shampine and R.C. Allen, Numerical Computing: An Introduction, Saunders, Philadelphia, 1973, Chap. 1.
7 M. Abramowitz and I.A. Segun, Handbook of Mathematical Functions, Dover, New York, 5th edn., 1968.
8 W.S. Dorn and D.D. McCracken, Numerical Methods with FORTRAN IV Case Studies, Wiley, New York, 1972, Chap. 5.
9 J.F. Hart, Computer Approximations, Wiley, New York, 1968, pp. 100—102.
10 A. Ralston and P. Rabinowitz, A First Course in Numerical Analysis, McGraw-Hill, New York, 2nd edn., 1978, Chap. 7.
11 Y.L. Luke, Mathematical Functions and their Approximations, Academic Press, New York, 1975, pp. 103—113.