Comp. & Maths. with Appls. Vol. 7, pp. 147–149
Pergamon Press Ltd., 1981. Printed in Great Britain

0097–4943/81/0401–0147/$02.00/0

IN DEFENSE OF LINEAR QUADRATURE RULES

WILLIAM SQUIRE
Department of Aerospace Engineering, West Virginia University, Morgantown, WY 26506, U.S.A.

Communicated by E. Y. Rodin

(*Received January* 1980)

Abstract—It is shown that appropiate linear quadrature rules can handle integrands with singularities at or near the end points more effectively than the nonlinear methods proposed by Werner and Wuytack. A special 10 point Gauss rule gives good results. A method with exponential convergence gives high accuracy with a moderate number of nodes. Both procedures were implemented on a programmable hand calculator.

INTRODUCTION

In a recent paper Werner and Wuytack[1] advocated the use of nonlinear quadrature methods based on spline and Pade approximations for cases where the integrand has a singularity at or near one of the endpoints. For a number of examples they showed that their procedures were superior to the midpoint and Simpson's rule and claimed it was competitive with an adaptive Gaussian quadrature scheme.

The purpose of this note is to demonstrate that:

(1) a special 10 point Gauss rule for integrands with singularities at or near the endpoints proposed by Harris and Evans[2] will give results comparing favourably to any other procedure using a comparable number of nodes.

(2) A quadrature rule, which Stenger[3] has shown to have exponential convergence, gives very accurate results for such integrands with a moderate number of function evaluations.

Both these procedures were implemented on an SR 52 programmable hand calculator.

DESCRIPTION OF METHODS AND CALCULATORS

Harris and Evans computed sets of modified Gaussian weights and nodes which in addition to being exact for high order polynomials would also be exact for specified singularities such as $\ln X$ and X^{-n} at the endpoints. To maintain symmetry they were also exact for $\ln(1-X)$ and $(1-X)^{-n}$. These rules could be used for any other finite range by the standard transformation.

For general purpose use they recommend a 10 point rule which is exact for a 11th degree polynomial, $\ln X$, $X^{-1/4}$, $X^{-1/2}$ and $X^{-3/4}$. The required nodes and weights are

Table 1.

$\pm X_i$	W_i
0.2295037173	0.4501100825
0.6364758401	0.3483026852
0.9015072053	0.1744679776
0.9928383122	$0.2696299772 \times 10^{-1}$
0.9999843443	$0.1562579734 \times 10^{-3}$

They note that it requires a machine with at least a ten digit number lenght to get satisfactory results because of the closeness of the outer nodes to the endpoints.

The SR–52 implementation is given in Appendix A. A "swap" card technique described by Williams[4] is used to avoid having to read in the weights and nodes. Application of the method to the examples used by Werner and Wuytack give the results shown in Table 2.

The other method uses the quadrature rule

$$\int_a^b f\,\mathrm{d}x \simeq (b-a)\ln q \sum_{j=-N}^{N} \frac{q^j}{(1+q^j)^2} f\left[\frac{bq^j+a}{1+q^j}\right] \qquad (1)$$

where q satisfies the condition

$$\ln q \sum_{j=-N}^{N} \frac{q^j}{(1+q^j)^2} = 1. \tag{2}$$

I have discussed the application of this method to a variety of integrals in previous papers [5, 6]. In the present paper where we consider small values of N the expression

$$q = \exp - \left(2/N - \sqrt{\left(\frac{2}{N}\right)\pi}\right) \tag{3}$$

is used as an improvement over $\exp(\sqrt{(2/N)}\pi)$ used previously. The method was implemented on an SR-52 as described in Appendix B. The results for the 8 integrals considered are tabulated in Table 2 for $N = 4$, 8, 16 and 32.

Table 2. Examples of singular and near singular integrals Stenger method—equation (1)

Integral	10 point rule	$N = 4$	8	16	32
$\int_0^1 \frac{e^x}{e^x - 0.99}\,dx = 5.15229793$	5.16206	5.11411	5.15105	5.1522896	5.152297933
$\int_0^1 \frac{3+(x-1)\cdot e^x}{(3-e^x)^2}\,dx = 3.54965677$	3.54058	3.53379	3.54921	3.5496431	3.549646776
$\int_0^1 \frac{e^x}{(3-e^x)^2}\,dx = 3.04964677$	3.04140	3.03526	3.04925	3.0496434	3.049646776
$\int_1^{1.5} (1+tg^2 x)\,dx = 12.54401217$	12.5335	12.5066	12.5425	12.544005	12.54401222
$\int_0^1 \frac{2\cdot(1-x)\cdot\sin x + \cos x}{\sqrt{(1-x)}}\,dx = 2$	2.04912	1.98070	1.99629	1.9996881	1.999991270
$\int_1^0 \sqrt{x}\cdot\ln x\cdot dx = \frac{4}{9} = 0.44444$	0.444470	0.444639	0.444451	0.44444447	0.4444444444
$\int_{\pi/2}^0 \ln(2\cdot\sin x/2)\cdot dx = 0.915965$	0.915965	0.909888	0.915691	0.91596280	0.9159655911
$\int_0^1 (1-x^{1/4})^4\cdot dx = \frac{1}{70} = 0.01428571$	0.014302	0.014060	0.014276	0.0142856408	0.0142857142

DISCUSSION OF RESULTS

It can be seen that taking $N = 32$ (65 nodes) gave the results to the accuracy of the SR-52 in 5 of the 8 cases. This is better than any of the methods considered by Werner and Wuytack even though the Pade Approximation methods use $3N$ values of f and its derivatives compared to $2N + 1$ for equation (1).

While much less accurate, the 10 point Gauss is an excellent supplement since it gives better results than any other procedure using a comparable number of function evaluations. The rapid loss of accuracy of equation (1) as N is reduced is a natural result of its exponential convergence.

The accuracy of the evaluation by the 10 point Gauss method could be increased by subdividing the range of integration at the expense of requiring more function evaluations. I have also made a preliminary evaluation of another nonlinear technique, partition-extrapolation [7] to some of the present examples but it appears to require a large number of function evaluations to get accurate results.

It therefore seems reasonable to conclude that appropriate linear methods not involving any change in variable or subtractions of singularities compare very favourably with nonlinear techniques for quadrature in the presence of singularities.

REFERENCES

1. H. Werner and L. Wuytack, Nonlinear quadrature rules in the presence of a singularity. *Comput. Maths Applies* **4**, 237–245 (1978).
2. C. G. Harris and W. A. B. Evans, Extension of numerical quadrature formulae to cater for end point singular behaviour over finite intervals. *Int. J. Comput. Maths.* **6B**, 219–227 (1977).
3. F. Stenger, Integration formulae based on the trapezoidal formula, *J. Inst. Maths. Applics.* **12**, 103–114 (1973).
4. D. T. Williams, Program provides card storage of SR-52 data-memory contents. *Electronics* **48**, 118–119 (14 October 1976).

5. W. Squire, A quadrature Method for finite intervals. *Int. J. Numer. Math. Engng*, **10**, 708–712 (1976).
6. W. Squire, Comment on quadrature in presence of singularities. *Int. J. Comput. Maths* **7B**, 239–241 (1979).
7. W. Squire Partition-extrapolation methods for numerical quadrature, *Int. J. Comput. Maths* **5B**, 81–91 (1975).

APPENDIX A

Harris-Evans 10 point rule

The SR-52 does not have the capacity to incorporate the weight and nodes directly into the program so they are stored in the memory registers and recalled as needed. The equivalent program incorporating the constants directly would be appreciably faster. This could be done on a machine such as the TI-59. To avoid entering the constants manually each time the program is read in a "swap" card is used. This card is prepared by punching:

```
10 STO 90 GTØ 168 LRN
*LBL A (RCL 90 STØ 98+70)
STØ 99 IND RCL 98 IND EXC 99
IND STØ 98 1+/-SUM 99 SUM 98
RCL 98 IF PØS 184
LBL B RCL 90 HLT
LBL C STO 90 HLT
```

The constants are stored in locations 01–10 and *A* is pressed. This switches the locations of these memory locations to registers 70–89 which are normally used to store instructions. The program can store up to 19 constants; the number to be stored being recorded in register 90. After the constants are swapped into the proper register the card is recorded. Then when the main program is to be run the "swap" card is first read in and (*A*) pushed to swap the constants into the operating memory. They are retained there when the program is entered. Only the instructions are replaced or erased.

```
*LBL B STØ 16
((RCL+RCL 14) E +(RCL 14-RCL 16) E)* rtn           ENTER b
*LBL A STO 12 STØ 14                               B(v)=E(m+v)+E(m-v)
(RCL-RCL 11)÷2=STØ 13                               s=(b-a)÷2
INV SUM 14                                          m=b-s
RCL 13×((RCL×RCL 01)      ×B×RCL 02                 q=s×(W1×B(s×x1)
 +RCL 04×(RCL 03 RCL 13) ×B                         +W2×B(s×x2)
 +RCL 06×(RCL 05 RCL 13) ×B                         +W3×B (s×x3)
 +RCL 08×(RCL 07 RCL 13) ×B                         +W4×B (s×x4)
 +RCL 10×(RCL 09 RCL 13) ×B                         +W5×B (s×x5))
*rtn                                                SHOW q
*LBL E....*rtn                                      DEFINE E(v)
```

First define integrand as *E*.

After storing lower limit, *a*, in register 11 enter upper limit *b* and press *A*.

APPENDIX B

SR-52 program for Stenger quadrature (equation 1)

```
*LBL D STØ 07
(RCL×RCL 02+RCL 01)÷(1+RCL 07=E
 ×RCL 07÷(1+RCL 07) *X^2=*rtn                       ENTER n

*LBL C STØ 00                                       D(v)=v/(1+v)^2 E((a+b×v)/(1+v))

*1/X×2=-*√X×*π=+/-STØ 03                            lq=(√(2/n))×π-2/n

INV lnx STØ 04                                      q=exp (lq)
1 STØ 05                                            v=1
D STØ 06                                            int=D (v)
*LBL SUM                                            LØØP n TIMES
RCL 04 *PRØD 05                                     v=v×q
RCL 05 D SUM 06                                     int=int+D (v)
RCL 05 *1/X D SUM 06                                int=int+D (1/v)
*dsz SUM                                            END OF LØØP
RCL 06×RCL 03×(RCL 02-RCL 01=                       int=int×lq×(b-a)
*rtn                                                SHOW int
*LBL E...*rtn                                       DEFINE integrand
```

After defining integrand store lower limit, *a*, in 01 and upper limit *b* in 02. The enter *N* and press *C*.

Note that program would run faster if E were placed at beginning of program, but this would require using INS when changing integrand.