King Fahd University of Petroleum & Minerals

DEPARTMENT OF MATHEMATICAL SCIENCES

Technical Report Series

TR 402

Dec 2008

Gauss-Type Quadrature Rules Using a Special Class of Polynomials

M. A. Bokhari and Asghar Qadir

Gauss-Type Quadrature Rules Using a Special Class of Polynomials

M. A. Bokhari[1] and Asghar Qadir[1, 2]
Dept. of Math. & Stat, KFUPM, Dhahran, Saudi Arabia[1]
CAMP, NUST, Rawalpindi, Pakistan[2]

Abstract— Some Gauss-type quadrature rules over [0, 1], which involve values and/or the derivative of the integrand at 0 and/or 1, are investigated. Our work is based on the orthogonal polynomials with respect to the linear weight function $\omega(t) := 1 - t$ over [0, 1] which arise from the recently developed "identity-type functions". Along the lines of Golub's work, the nodes and weights of the quadrature rules are computed from Jacobi-type matrices having simple rational entries. Computational procedures relevant to the derived rules are tested on different integrands. The proposed methods have some advantage over the respective Gauss-type rules with respect to constant weight function $\omega(t) := 1$ over [0,1].

Index Terms— Gauss/Gauss-Radau /Gauss-Lobatto quadrature rules, Jacobi-matrix, Hypergeometric series, Identity-type polynomials, 3-term recurrence relation.

[1] e-mail: mbokhari@kfupm.edu.sa

[2] e-mail: aqadirmath@yahoo.com

I. INTRODUCTION

Numerical integration is usually utilized when analytical techniques fail. On the other hand, when a high accuracy is required, the number of steps related to a quadrature rule can become too large, creating inaccuracies due to error propagation. For this reason one looks for efficient schemes that can reduce the number of steps while retaining the accuracy at the desired level. The Gaussian quadrature rule is one such scheme. It requires weighted integrand's values at the zeros of orthogonal polynomials, which lie inside the interval of integration, say [0, 1]. Several other schemes are designed that deal in addition with the values of integrand and/or its derivatives at 0 and/or 1. Among these, Gauss-Radau and Gauss-Lobatto rules are quite prominent. These rules also prove useful in determining approximate solution of boundary value problems. A lot of work is still going on in the modification of quadrature rules and their computational aspects. Gautschi and Li discussed some formulas with double end points nodes for all four Chebyshev weight functions in [5]. These are referred to as generalized Gauss-Radau and Gauss-Lobatto rules in the literature. In 1983, Golub and Kautsky [8] derived algorithms for the evaluation of Gauss knots in the presence of fixed knots by modifying the Jacobi matrix. Gautschi developed computational methods for generating Gauss-type quadrature formulae having nodes of arbitrary multiplicity at one or both end points of the interval of integration [3]. Li studied Kronrod extensions of Gauss-Radau and Gauss-Lobatto formulae having end points of multiplicity 2 and derived explicit formulas associated with the end points for the Chebyshev weights [10].

In a totally different context [1] some functions with interesting properties had recently been defined. Specifically, they are a special class of hypergeometric functions that are orthogonal relative to the weight function $\omega(t) \coloneqq 1-t$, which include what were called "identity- type polynomials" that retain the orthogonality property. Since hypergeometric functions arise in the solution of Sturm-Liouville problems, it could be expected that these functions may be useful for the quadrature schemes, especially if the quadrature is used for solving differential equations. Here we examine the possibility of using these polynomials in the modification of Gauss type rules.

The plan of the present report is as follows. In the next section we review the Gauss, Gauss-Radau and Gauss-Lobatto schemes subject to weight function $\omega(t) := 1-t$ over [0, 1]. In the subsequent section we provide a review of the identity-type polynomials. In section 4 we discuss the orthogonality of the factor polynomials. Some numerical examples are discussed in section 5. Finally, in section 6 we present a brief summary and discussion of our results.

II. GAUSS-TYPE QUADRATURE PROCEDURES

We begin with setting

$$f_\omega(t) := \begin{cases} \dfrac{f(t)-f(1)}{t-1}, & t \neq 1 \\ f'(1), & t = 1, \end{cases}$$

where $f : [0,1] \to \Re$ is an "appropriate" function and $\omega(t) := 1-t$. By use of the identity

$$\int_0^1 f(x)dx \equiv f(1) - \int_0^1 f_\omega(x)\omega(x)dx,$$

we modify the existing *n*-point Gauss-type rules after slight algebraic manipulation in the standard formulas [4, (1.4.7), (3.1.13), (3.1.26)] as follows:

i. Gauss Rule:

$$\int_0^1 f(x)dx \approx f(1) - \sum_{i=1}^{n} v_i^G f_\omega(t_i^G), \tag{1}$$

ii. Gauss-Radau Rule (Right end):

$$\int_0^1 f(x)dx \approx f(1) - \sum_{i=1}^{n-1} v_i^R f_\omega(t_i^R) - v_n^R f'(1), \tag{2}$$

iii. Gauss-Lobatto Rule:

$$\int_0^1 f(x)dx \approx f(1) + v_1^L (f(0)-f(1)) - \sum_{i=2}^{n-2} v_i^L f_\omega(t_i^L) - v_n^L f'(1). \tag{3}$$

Here, $\left(t_i^G, v_i^G\right), \left(t_i^R, v_i^R\right)$ and $\left(t_i^L, v_i^L\right)$ respectively denote the pairs of nodes and weights for the respective rules (i)-(iii). These nodes and weights are directly related to the zeros of the k^{th} degree orthogonal polynomials $p_k, k = 1, 2, \ldots$ with respect to weight function $\omega(t) := 1-t$ over [0, 1]. The three term recurrence relation [4] as described below is the most significant tool for the computation of $p_k, k = 1, 2, \ldots$:

$$\left.\begin{aligned} p_{k+1}(t) &= (t-\alpha_{k+1})p_k(t) - \beta_{k+1}p_{k-1}(t) \\ p_{-1}(t) &= 0,\ p_0(t) = 1 \end{aligned}\right\} \tag{4}$$

with

$$\alpha_{k+1} = \frac{\langle tp_k, p_k \rangle_w}{\langle p_k, p_k \rangle_w}, \quad \beta_{k+2} = \frac{\langle p_{k+1}, p_{k+1} \rangle_w}{\langle p_k, p_k \rangle_w}; k = 0,1,2,.... \tag{5}$$

Here, the notation $\langle F, G \rangle_\omega$ stands for $\int_0^1 F(t)G(t)\omega(t)dt$. The knowledge of the recursion coefficients α_k and β_k in (5) allows the zeros of $p_k, k = 0,1,2,...$ and the weights to be readily calculated from eigenvalues and eigenvectors of a symmetric tri-diagonal matrix [7, 9]. This idea depends on the fact [12] that the *n* weights v_i^G (cf. (1)) of the Gauss quadrature rules are given by (cf. (1), (4))

$\left(v_i^G\right)^{-1} = \sum_{k=0}^{n-1} p_k^2(t_i^G)$.

Golub and Welsh suggested an elegant procedure [9] for computing nodes and weights which is summarized in

Theorem A: *The n free nodes of the n-point Gauss quadrature formula with respect to weight function* $w(t)$ *are precisely the eigenvalues of a Jacobi-type matrix*

$$\boldsymbol{J}_{n,w} = \begin{bmatrix} \alpha_1 & \sqrt{\beta_1} & 0 & & & \cdots & 0 \\ \sqrt{\beta_1} & \alpha_2 & \sqrt{\beta_2} & 0 & & & 0 \\ 0 & \sqrt{\beta_2} & \alpha_3 & & & & \\ & & & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & \alpha_n & \sqrt{\beta_n} & 0 \\ \vdots & & & \ddots & \sqrt{\beta_n} & \alpha_{n+1}^R & \sqrt{\beta_{n+1}^L} \\ 0 & 0 & & \cdots & 0 & \sqrt{\beta_{n+1}^L} & \alpha_{n+2}^L \end{bmatrix} \tag{6}$$

where in case of

(i) *Gauss rule: we delete the last two columns as well as the last two rows in* $\boldsymbol{J}_{n,w}$

(ii) *Gauss-Radau rule* (*Right end*): *we delete the last column and the last row in* $\boldsymbol{J}_{n,w}$ *and set* $\alpha_{n+1}^R = 1 - \beta_n \frac{p_{n-1}(1)}{p_n(1)}$.

(iii) *Gauss-Lobatto rule: we choose* $\alpha_{n+1}^R = \alpha_{n+1}$ *and* α_{n+2}^L and β_{n+1}^L *as the solution of the* 2×2 *linear system*

$$\begin{bmatrix} p_{n+1}(0) & p_n(0) \\ p_{n+1}(1) & p_n(1) \end{bmatrix} \begin{bmatrix} \alpha_{n+2}^L \\ \beta_{n+1}^L \end{bmatrix} = \begin{bmatrix} 0 \\ p_{n+1}(1) \end{bmatrix}.$$

Moreover, the weights required in the respective rules are given by $\beta_0 u_{i,1}^2$, *where* $\beta_0 := \int_0^1 \omega(x)dx$ and $u_{i,1}$ is *the first component of the associated normalized eigenvector* u_i

Remark 1: *The quadrature rules (i)-(iii) are also applicable to an integral over a finite interval* $[a,b]$. *For this purpose it is enough to note that* $\int_a^b g(x)dx = \int_0^1 f(t)dt$ *with* $f(t) := (b-a)\,g\big((b-a)t+a\big)$, $t \in [0,1]$. *In addition, Rules* (i) *and* (ii) *can be considered for 0 as a fixed node instead of 1. In this case, we consider* $\int_0^1 g(x)dx = \int_0^1 f(t)dt$ *with* $f(t) := g(1-t)$, $t \in [0,1]$.

We are interested in computational aspects of the proposed rules (1)-(3) that may minimize accumulation of round off error in the process of programming. Their efficacy, indeed, relies on appropriate representation of recursion coefficients α_k, β_k (cf. (5)). With some algebraic manipulation these coefficients can be expressed by simple rational sequences which we figure out from the identity-type polynomials discussed below.

III. IDENTITY-TYPE POLYNOMIALS

It is known that the identity-type function [1, (2.17)-(2.18)]

$$\hat{e}(t;c) := \frac{\Gamma(c)}{\Gamma(c-1)} \sum_{m=0}^{\infty} \frac{(c)_m (-c)_m}{(m!)^2} t^m, \tag{7}$$

with

$(c)_0 = 1$ and $(c)_n = c(c-1)(c-2)....(c-n+1)$,

satisfies the second order differential equation

$$t(1-t)\frac{d^2y}{dt^2} + (1-t)\frac{dy}{dt} + c^2 y = 0, \qquad c > 0.$$

The hypergeometric series $\sum_{m=0}^{\infty} \frac{(c)_m (-c)_m}{(m!)^2} t^m$ in (7) is defined for all integral values of c. Since $(n)_m = 0$ for $m \geq n+1$, the right side of (7) turns out an nth degree polynomial when c is replaced by n. We set

$$e_n(t) := \sum_{m=0}^{\infty} \frac{(n)_m (-n)_m}{(m!)^2} t^m, \quad n = 1, 2, 3, ...$$

The *nth* degree polynomial is the solution of the Sturm-Liouville problem [6]

$$\frac{d}{dt}\left(t\frac{dy}{dt}\right)+\frac{n^2}{1-t}y=0\,.$$

To meet our objective, we reproduce some properties of e_n already discussed in [1]:

(a) $(1-t)$ is a factor of each e_n, n = 1, 2,… . In particular, we can write

$$e_n(t) := (1-t)e_{n-1}^*(t) \tag{8}$$

where first few factor polynomials e_{n-1}^*, n = 1, 2,…, are given below:

$$e_1^*(t)=1$$
$$e_2^*(t)=(1-3t)$$
$$e_3^*(t)=(1-8t+10t^2)$$
$$\vdots$$

(b) The polynomials $e_n, n=0,1,2,...$ are orthogonal with respect to $\omega^*(t)=\frac{1}{(1-t)}$ over [0,1]. In fact,

$$\left\langle e_n, e_m\right\rangle_{\omega^*}=\begin{cases}0 & \text{if } n\neq m\\ \dfrac{1}{2n} & \text{if } n=m\end{cases}\quad. \tag{9}$$

IV. Orthogonality of Factor Polynomials

Our interest lies in certain properties of the factor polynomials e_{n-1}^* which appear in (8). Using the notion of Gauss hypergeometric functions and some transformations, it has been shown in [1, (3.4)] that

$$e_{n-1}^*(1)=(-1)^{n+1}n$$

Next we define monic polynomials

$$p_{n-1}(t) := \frac{-1}{\kappa_n}e_{n-1}^*(t)\ ,\quad n=1,\,2,\ldots\ldots$$

where the notation κ_n stands for the leading coefficient of $e_n(t)$. Some properties of the monic polynomials $p_{n-1}(t)$ are stated in

Theorem1. *The polynomials* p_{n-1} , *n = 1, 2,… are orthogonal with respect to weight function* $\omega(t) := 1-t$ *over* [0, 1] *and satisfy the 3-term recurrence* (cf. (4))

$$p_{n+1}(t)=(t-\alpha_n)p_n(t)-\beta_n p_{n-1}(t)$$

with

$$\left.\begin{aligned} \alpha_n &= \frac{\left\langle te_{n+1}^*, e_{n+1}^* \right\rangle_{\omega^*}}{\left\langle e_{n+1}^*, e_{n+1}^* \right\rangle_{\omega^*}} \\ \beta_{n+1} &= \left(\frac{\kappa_{n+1}}{\kappa_{n+2}}\right)^2 \frac{\left\langle e_{n+2}^*, e_{n+2}^* \right\rangle_{\omega^*}}{\left\langle e_{n+1}^*, e_{n+1}^* \right\rangle_{\omega^*}} \end{aligned}\right\}, \quad n = 0,1,2,... \tag{10}$$

In addition,

$$\kappa_n = (-1)^n \frac{(2n-1)!}{n!(n-1)!}, \tag{11}$$

$$\left.\begin{aligned} p_{n-1}(0) &= -\kappa_n^{-1} \\ p_{n-1}(1) &= (-1)^n \, n\kappa_n^{-1} \end{aligned}\right\}, \tag{12}$$

Proof: *The orthogonality of* $p_{n-1}, n = 1,2,...$ *follows from the identity* $\left\langle p_{n-1}, p_{m-1} \right\rangle_\omega = \frac{1}{\kappa_n \kappa_m} \left\langle e_n^*, e_m^* \right\rangle_{\omega^*}$ *and the relation* (*9*)*. In order to prove* (*10*)*, we have to show that*

$$\alpha_n = \frac{\left\langle tp_n, p_n \right\rangle_\omega}{\left\langle p_n, p_n \right\rangle_\omega}, \quad \beta_n = \frac{\left\langle p_n, p_n \right\rangle_\omega}{\left\langle p_{n-1}, p_{n-1} \right\rangle_\omega} \quad .$$

For this, we express each of the inner products appearing in (*10*) *in terms of the monic polynomials* p_n *. As an example, one can see that*

$$\left\langle te_{n+1}^*, e_{n+1}^* \right\rangle_{\omega^*} = \int_0^1 t \frac{(1-t)^2}{\tau_{n+1}^2} p_n^2(t) \omega^*(t) dt = \frac{1}{\tau_{n+1}^2} \left\langle tp_n, p_n \right\rangle_\omega .$$

To prove (*11*)*, recall that* $\kappa_n = \frac{(n)_n (-n)_n}{(n!)^2}$ *. Here, the right side expression immediately simplifies to* $(-1)^n \frac{(2n-1)!}{n!(n-1)!}$.

The relations in (*12*) *depend on the relation* $e_n(0) = \frac{(n)_0 (-n)_0}{(0!)^2}$ *and the description of* p_n *. This completes the proof.*

A simple and explicit representation of the recursion coefficients in (10) plays a vital role to avoid accumulation of round off error while programming the proposed quadrature rules. We conclude this in

Theorem 2: *Let* $\omega(t) := 1-t$ *be the weight function in Theorem A. Then*

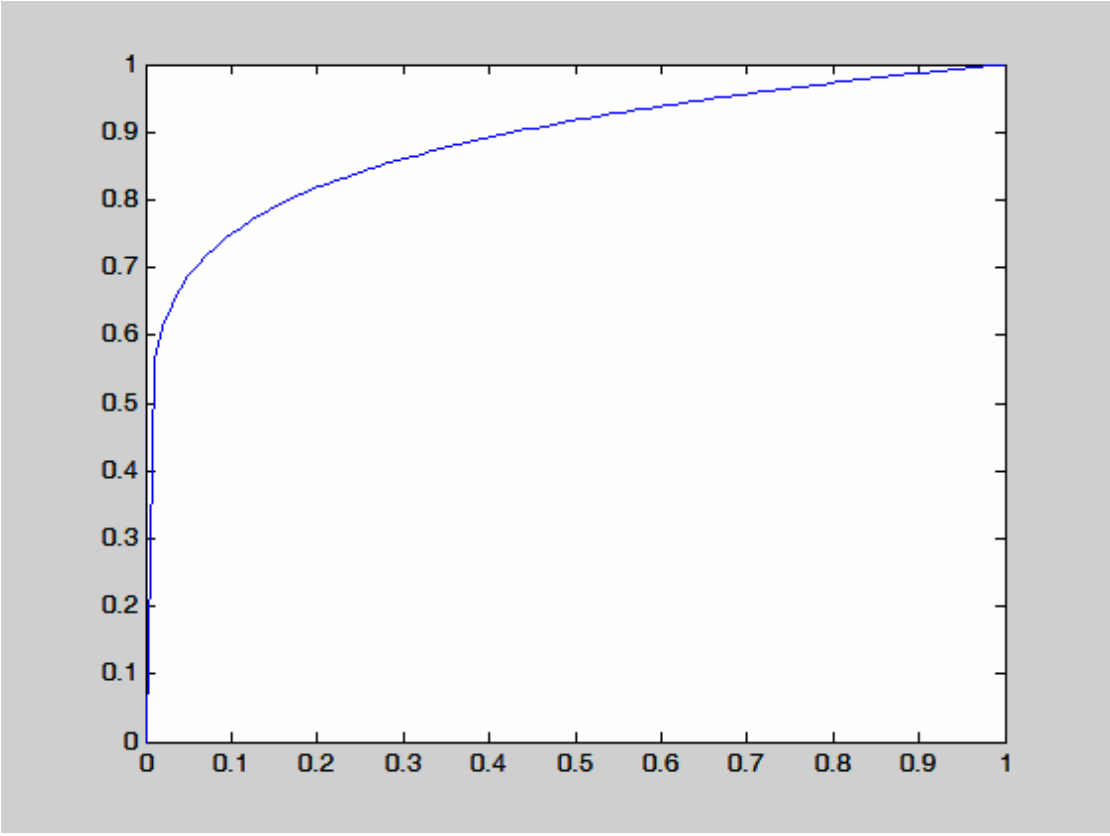
$$\left.\begin{aligned}
&(1)\ \alpha_n = \frac{2n^2-1}{4n^2-1} \\
&(2)\ \beta_n = \frac{n(n+1)}{4(2n+1)^2} \\
&(3)\ \alpha_{n+1}^{R,1} = \frac{3n^2+6n+2}{4n^2+6n+2} \\
&(4)\ \alpha_{n+2}^{L} = \frac{n+2}{2n+3} \\
&(5)\ \beta_{n+1}^{L} = \frac{(n+2)^2}{2(2n+3)^2}
\end{aligned}\right\},\ n = 1,2,\ldots$$

Remark 2: *The sequences* $\{\alpha_n\}$ and $\{\beta_n\}$ *as determined in the above theorem provide an explicit representation of the Jacobian-type matrix* (*cf* (*6*)) *in terms of n. Thus the nodes and weights required in the proposed quadrature formulae* (*1*) - (*3*) *are easily obtainable by an application of Theorem A.*

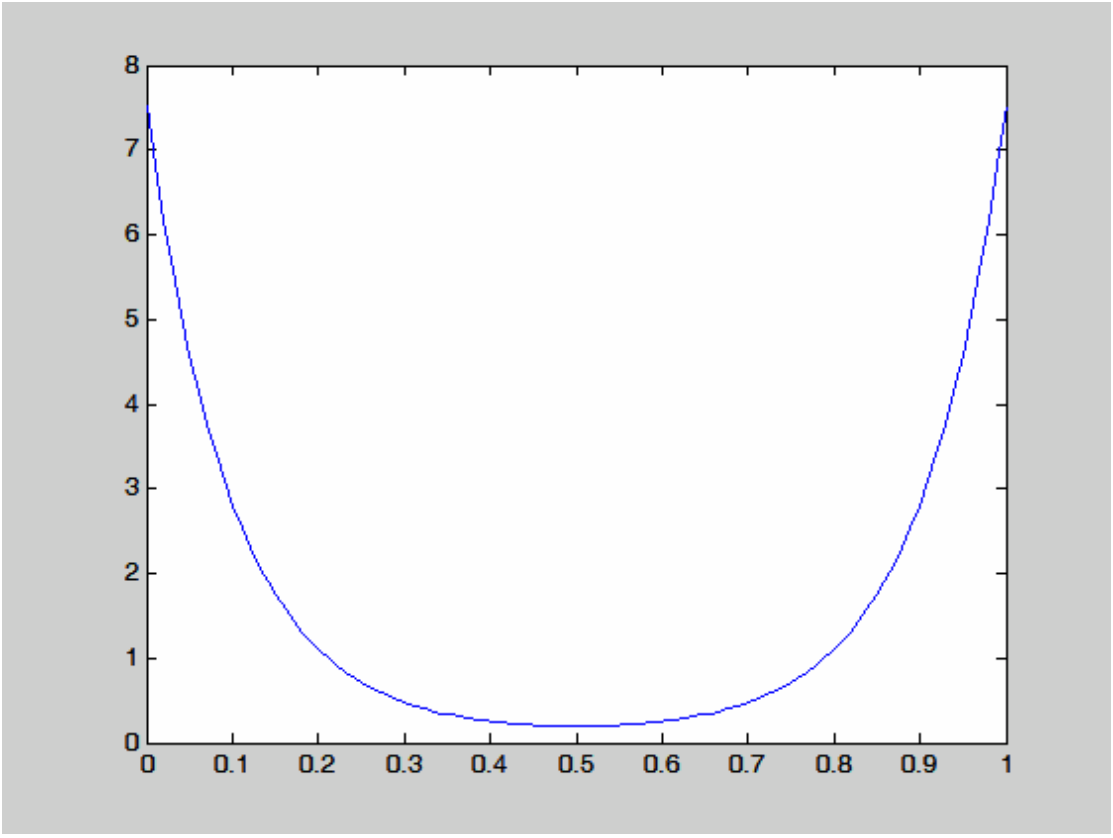
V. Numerical Examples

We have applied the quadrature rules (1)-(3) to four functions having different characteristics:

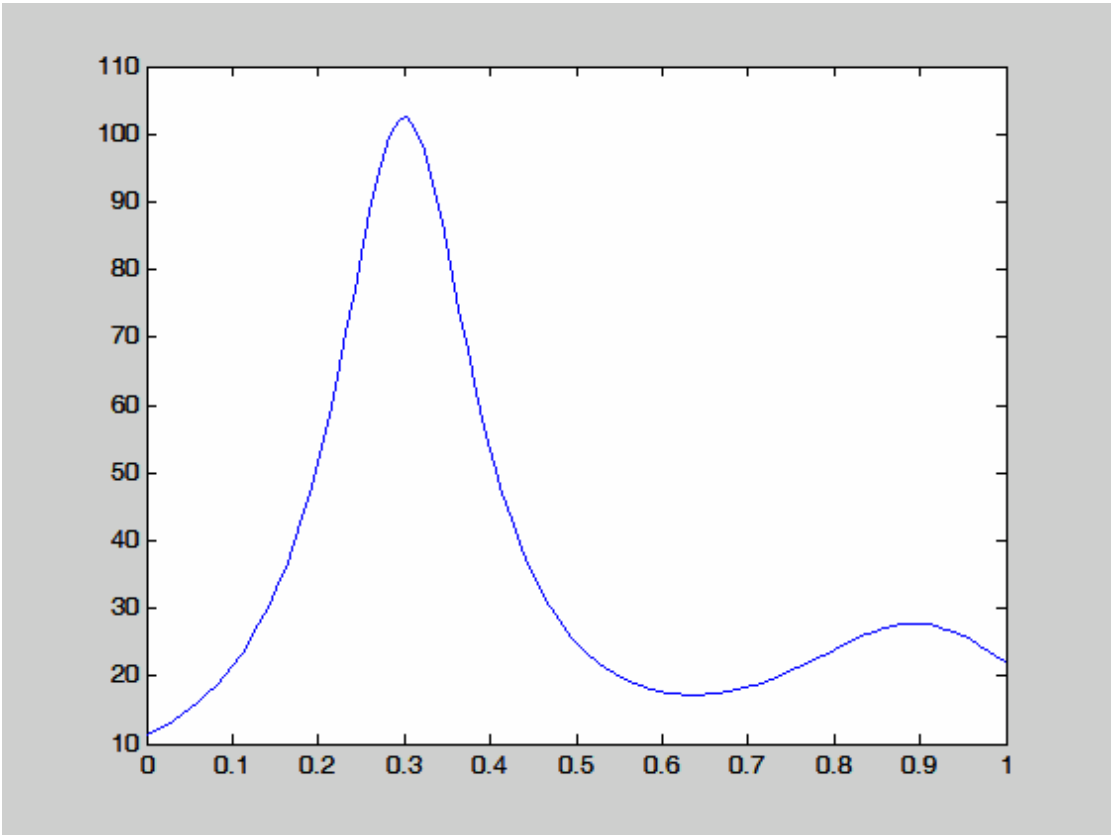
1) $f_1(x) = x^{1/8}$



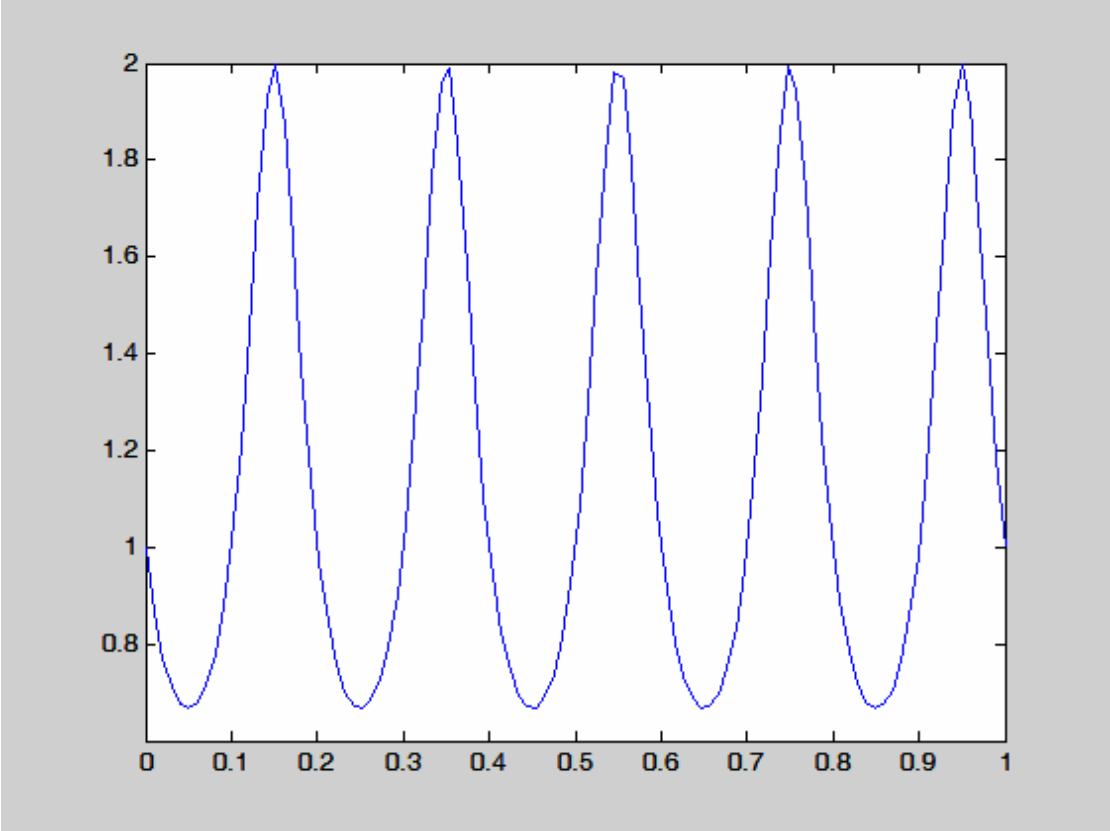
2) $f_2(x) = \cosh^2\left(5(x-0.5)\right)/5$



3) $f_3(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04}$



4) $f_4(x) = \dfrac{2}{2+\sin(10\pi x)}$.



The features for which they were chosen were that one should start at zero and end up non-zero monotonically, one should start non-zero and go to a non-zero value with a minimum between, so that the ill conditioning near the edges could be tested for, one should have a couple of maxima and a minimum in the domain and one should have many oscillations. The performance of the rules in each case could then be assumed to be representative of such functions. Similar functions have been considered as test integrands in ([2], [11] and [12]). We have computed percentage errors that occurred in the approximation of all four integrals by using n-Gauss-type rules, n = 2, 3,…, 11, with respect to both weight functions $\omega(t) := 1-t$ and $\omega(t) := 1$. The data for each integrand is provided in separate tables for the sake of comparison. The following abbreviations have been used to identify the rule in each table:

- G(1) := Gauss rule w.r.t. $\omega(t) := 1$
- G $(1 - t)$:= Gauss rule w.r.t. $\omega(t) := 1-t$
- G-R(1) := Gauss-Radau rule (Right end point) w.r.t. $\omega(t) := 1$
- G-R$(1 - t)$:= Gauss-Radau rule (Right end point) w.r.t. $\omega(t) := 1-t$.
- G-L(1) := Gauss-Lobatto rule w.r.t. $\omega(t) := 1$
- G-L$(1 - t)$:= Gauss-Lobatto rule w.r.t. $\omega(t) := 1-t$.

Table 1.1

$f_1(x) = x^{1/8}$

n	%Error G (1)	%Error G (1-*t*)
2	0. 922	**0.622**
3	0. 423	**0.317**
4	0. 238	**0.189**
5	0. 151	**0.125**
6	0.103	**0.088**
7	0. 075	**0.065**
8	0.056	**0.049**
9	0.043	**0.039**
10	0.035	**0.031**
11	0.028	**0.026**

Table 1.2

$f_1(x) = x^{1/8}$

n	%Error G-R (1)	%Error G-R (1-*t*)
2	0.622	**0.120**
3	0.317	**0.041**
4	0.189	**0.063**
5	0.125	**0.058**
6	0.088	**0.049**
7	0.065	**0.041**
8	0.049	**0.034**
9	0.039	**0.028**
10	0.031	**0.024**
11	0.026	**0.020**

Table 1.3

$f_1(x) = x^{1/8}$

n	%Error G-L (1)	%Error G-L (1-*t*)
2	5.693	**4.424**
3	3.198	**2.604**
4	2.025	**1.702**
5	1.386	**1.192**
6	1.002	**0.878**
7	0.755	**0.671**
8	0.587	**0.527**
9	0.468	**0.425**
10	0.381	**0.349**
11	0.316	**0.291**

Table 2.1

$f_2(x) = \cosh^2\left(5(x-0.5)\right)/5$

n	%Error G (1)	%Error G (1-*t*)
2	36.901	**3.082**
3	6.523	**0.202**
4	0.647	**9.72**$\times 10^{-3}$
5	0.041	**3.48**$\times 10^{-4}$
6	1.83×10^{-3}	**9.54**$\times 10^{-6}$
7	6.02×10^{-5}	**2.05**$\times 10^{-7}$
8	1.50×10^{-6}	**3.58**$\times 10^{-9}$
9	2.99×10^{-8}	**5.15**$\times 10^{-11}$
10	4.79×10^{-10}	**4.76**$\times 10^{-13}$
11	6.29×10^{-12}	**1.40**$\times 10^{-14}$

Table 2.2

$f_2(x) = \cosh^2\left(5(x-0.5)\right)/5$

n	%Error G-R (1)	%Error G-R (1-*t*)
2	3.082	**14.037**
3	0.202	**1.103**
4	9.72×10^{-3}	**0.06.2**
5	3.48×10^{-4}	**2.57**$\times 10^{-3}$
6	9.54×10^{-6}	**8.00**$\times 10^{-5}$
7	2.05×10^{-7}	**1.93**$\times 10^{-6}$
8	3.58×10^{-9}	**3.72**$\times 10^{-8}$
9	5.12×10^{-11}	**5.84**$\times 10^{-10}$
10	6.86×10^{-13}	**7.80**$\times 10^{-12}$
11	8.41×10^{-14}	**3.36**$\times 10^{-13}$

Table 2.3

$f_2(x) = \cosh^2\left(5(x-0.5)\right)/5$

n	%Error G-L (1)	%Error G-L (1-*t*)
2	9.284	**1.078**
3	0.834	**0.045**
4	0.050	**1.49**$\times 10^{-3}$
5	2.16×10^{-3}	**3.87**$\times 10^{-5}$
6	6.92×10^{-5}	**8.02**$\times 10^{-7}$
7	1.70×10^{-6}	**1.35**$\times 10^{-8}$
8	3.33×10^{-8}	**1.88**$\times 10^{-10}$
9	5.28×10^{-10}	**2.13**$\times 10^{-12}$
10	6.95×10^{-12}	**2.80**$\times 10^{-14}$
11	9.81×10^{-14}	**1.26**$\times 10^{-13}$

Table 3.1

$$f_3(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04}$$

n	%Error G (1)	%Error G (1-*t*)
2	12.988	**32.88**
3	29.185	**9.812**
4	23.274	**17.806**
5	5.735	**12.173**
6	5.474	**4.624**
7	6.900	**0.919**
8	3.740	**2.995**
9	0.567	**2.323**
10	1.038	**0.806**
11	1.195	**0.249**

Table 3.2

$$f_3(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04}$$

n	%Error G-R (1)	%Error G-R (1-*t*)
2	32.884	**34.468**
3	9.812	**17.083**
4	17.806	**2.339**
5	12.173	**7.241**
6	4.624	**7.420**
7	0.919	**3.496**
8	2.995	**0.175**
9	2.323	**1.265**

Table 3.3

$$f_3(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04}$$

n	%Error G-L (1)	%Error G-L (1-*t*)
2	42.957	**0.143**
3	20.888	**17.980**
4	4.551	**15.027**
5	4.340	**4.979**
6	6.547	**1.502**
7	4.076	**3.257**
8	0.792	**2.344**
9	1.018	**0.792**
10	1.229	**0.268**
11	0.672	**0.581**

Table 4.1

$$f_4(x) = \frac{2}{2+\sin(10\pi x)}$$

n	%Error G (1)	%Error G (1-*t*)
2	10.688	**3.721**
3	11.511	**13.906**
4	3.733	**20.939**
5	1.804	**23.850**
6	0.0102	**49.801**
7	7.133	**22.412**
8	4.733	**12.604**
9	4.080	**3.055**
10	0.585	**1.059**
11	5.71	**0.535**

Table 4.2

$$f_4(x) = \frac{2}{2+\sin(10\pi x)}$$

n	%Error G-R (1)	%Error G-R (1-*t*)
2	3.721	**58.697**
3	13.906	**21.056**
4	20.939	**10.518**
5	23.850	**12.648**
6	49.801	**1.632**
7	22.412	**10.528**
8	12.604	**1.417**
9	3.055	**1.269**
10	1.059	**4.751**
11	0.535	**3.091**

Table 4.3

$$f_4(x) = \frac{2}{2+\sin(10\pi x)}$$

n	%Error G-L (1)	%Error G-L (1-*t*)
2	4.105	**20.742**
3	5.516	**27.428**
4	3.904	**0.387**
5	3.032	**35.580**
6	8.408	**36.267**
7	0.486	**11.073**
8	3.160	**4.061**
9	1.214	**0.605**
10	5.516	**0.150**
11	2.617	**1.017**

VI. CONCLUSION

We have introduced Gauss-type quadrature rules (cf. (1) - (3)) on the interval [0, 1] which, except for rule (1), involve the derivative of the integrand at the right end of the interval. These rules also preserve the exactness and convergence properties like the corresponding Gauss quadrature rules. However, our proposed rules require an additional functional value as compared to the rules based on weight function 1. Four functions of different characteristic were selected for implementation of the proposed rules. Percentage error is used as a criterion of accuracy.

For f_1 we note that the rules using weight function $\omega(t) := 1-t$ are consistently better than those for weight function $\omega(t) := 1$. Further, Gauss-Radau is the best rule for this function and Gauss-Lobatto the worst. The difference between the accuracy of the Gauss scheme and the Gauss-Radau scheme is relatively small while the ratio between these two and Gauss-Lobatto is an order of magnitude. The accuracy of the Gauss scheme with $\omega(t) := 1-t$ is about the same as that of the Gauss-Radau with weight factor $\omega(t) := 1$. Since the two schemes are of nearly the same percentage accuracy, it is worth while to define a measure of *relative accuracy*, α_r, defined as the ratio of the percentage accuracies. We see that for the Gauss scheme $\alpha_r \approx 1.13$ (i.e. 13 % more accurate) in favour of $\omega(t) := 1-t$, at n = 10. At n = 35 it is found to reduce to about 1.03. For the Gauss-Radau scheme at n = 10, we have $\alpha_r \approx$ 1.29. At n = 35 it reduces to about 1.05. For the Gauss-Lobatto the relative accuracy is about the same as for the Gauss scheme. Thus, for very large n the schemes tend to converge in accuracy but for smaller n the differences are significant. It is very interesting to note that the percentage accuracy of the Gauss-Radau scheme with $\omega(t) := 1$ is the same as for the Gauss scheme with $\omega(t) := 1-t$ for all n that we checked for this function.

For f_2 the percentage error is extremely small at n = 10 in all cases. Checking at n = 20 we see that it has already reached the limit of numerical stability at n =10 or so. Here the Gauss-Lobatto is the best and the Gauss-Radau the worst. Further, the weight function $\omega(t) := 1$ is *better* for the Gauss-Radau than $\omega(t) := 1-t$. Here $\alpha_r \approx 11.4$ in favour of $\omega(t) := 1$. On the other hand for the Gauss scheme, $\alpha_r \approx$ 993.7 in favour of $\omega(t) := 1-t$. This is the biggest advantage for this weight function. For the Gauss-Lobatto we have $\alpha_r \approx 40.3$ in favour of $\omega(t) := 1-t$.

In the case of f_3, the famous humps function, the percentage error is seen to fluctuate up to n = 11. As such we cannot draw any conclusions from it. One can look at the general change in accuracy as we go

over the ranges n =12 – 16, 17 – 21, 22 – 26, 27 – 31, 32 – 36, by taking averages over the ranges. We see that the error decreases as one goes to higher n ranges. The differences are given in Table 5.

Table 5

	G(1)	**G(1-t)**	**G-R(1)**	**G-R(1-t)**	**G-L(1)**	**G-L(1-t)**
I (12 – 16)	2.5×10^{-1}	2.5×10^{-1}	2.5×10^{-1}	1.2×10^{-1}	1.3×10^{-1}	1.6×10^{-1}
II (17 – 21)	2.1×10^{-2}	2.7×10^{-2}	2.7×10^{-2}	2.2×10^{-2}	2.1×10^{-2}	1.3×10^{-2}
III (22 -26)	3.9×10^{-3}	2.3×10^{-3}	2.3×10^{-3}	2.6×10^{-3}	2.6×10^{-3}	1.8×10^{-3}
IV (27 – 31)	4.6×10^{-4}	3.3×10^{-4}	3.3×10^{-4}	2.3×10^{-4}	2.3×10^{-4}	2.6×10^{-4}
V (32-26)	4.0×10^{-5}	4.4×10^{-5}	4.4×10^{-5}	2.6×10^{-5}	2.6×10^{-5}	2.4×10^{-5}

We note that the identity between G(1-t) and G-R(1) carries over in this case despite the fluctuations. Further, for these functions, G(1), G(1-t), G-R(1) are comparable in accuracy and G-R(1-t), G-L(1) and G-L(1-t) are comparable in accuracy. The latter group is more accurate than the former group. For all schemes the accuracy goes up roughly by an order of magnitude from one range to the next. The difference in accuracy between the two groups of schemes is nearly a factor of two.

The last test function, f_4, is highly oscillating. Here, we note that the percentage error fluctuates wildly with change in n for all the six cases (cf. Tables 4.1 - 4.3). It appears that the greater the number of local extrema the greater the oscillations. For the hump function ranges of 5 are seen to be adequate. For this function ranges of 10 turn out to be barely adequate for larger values of n. It seems that we need to average over ranges of n of *twice* the number of local extrema in the domain.

There is a significant observation about the stability of the proposed methods. It may be noted that the recurrence coefficients and parameters appear as simple rational expressions with respect to the weight function $\omega(t) := 1 - t$. There is no accumulated round-off error in the computation of these coefficients. Therefore, we have observed stability of the three methods up to n = 2000 in this case. On the other hand, the methods G-L(1) and G-R(1) fail to work respectively after n = 272 and n = 539 because of the following observations in the MATLAB programming:

i) The 2x2 linear system (cf Theorem 3 (iii)) that determines the additional parameters for G-L(1) method turns out inconsistent beyond n = 272.

ii) In case of G-R(1), the value $p_n(1)$ after n = 539 becomes zero.

An underlying reason for the drawback is the instability due to accumulation of the round-off error in the computation of the recurrence coefficient (cf (5)) in case of weight $\omega(t) := 1$.

As mentioned in Remark 2 the proposed rules can be easily used for any finite interval $[a,b]$ instead of $[0,1]$ by appropriate translation and re-scaling.

ACKNOWLEDGMENT

Both authors are grateful to KFUPM for the excellent research facilities availed during the preparation of this article.

REFERENCES

[1] M.A. Chaudhry and A. Qadir, "Identity-type functions and polynomials", *Int. J. Math. & Math*. Sci. **2007**, ID 94204, 10 pages.

[2] P.J. Davis and P. Rabinowitz, "Methods of Numerical Integration", Second Edition, Academic Press, 1984.

[3] W. Gautschi, "Orthogonal Polynomials: applications and computation", in A. Iserles (Ed.), *Acta Numerica*, Cambridge Univ. Press (1996).

[4] W. Gautschi, *Orthogonal Polynomials: Computation and Approximation*, Oxford University Press, Oxford, 2004.

[5] W. Gautschi and S. Li, "Gauss-Radau and Gauss-Lobatto Quadratures with Double End Points", *J. Comp. Appl. Math. 34*(3)*,* 343-360 (1991).

[6] I.S. Gradshteyn and I.M. Ryzhik, *Tables of Integrals, Series and Products*, A. Jeffrey (Ed.), Academic Press, 1994.

[7] G. H. Golub, "Some modified matrix eigenvalue problems", *SIAM Rev. 15,* 318-334 (1973).

[8] G. H. Golub and J. Kautsky, "Calculation of Gauss quadratures with multiple free and fixed knots", *Numer. Math* (41) (1983) 147-163.

[9] G. H. Golub and J. H. Welsh, Calculation of Gauss quadrature rules. *Math Comp.* (23) (1969) 221-230.

[10] S. Li, "Kronrod extension of generalized Gauss-Radau and Gauss-Lobatto formulae", *Rocky Mountain J. Math. 26*(4), 1455-1472 (1996).

[11] C. B. Möler, *Numerical computing with MATLAB*, SIAM (2004).

[12] G. Recktenwald, *Numerical Methods with MATLAB: Implementation and Application*, Prentice Hall (2000).

[13] H. S. Wilf, *Mathematics for Physical Sciences*, Wiley, (1962).