Numerical implementations of Gaussian rules

Alvise Sommariva

University of the National Education Commission,
Kraków (Poland)
December 9th, 2024

Purpose

In this presentation we consider the Gaussian rules, for numerical integration over an interval (a, b), $-\infty \leq a < b \leq +\infty$, via a weighted sum, that is

$$\int_a^b f(x)w(x)dx \approx \sum_{k=1}^{n} w_k f(x_k).$$

where w is a weight function on (a, b).
In particular, we give

- a basic introduction of weight functions and related orthogonal polynomials (with historical remarks),
- some notes on the computation of Gaussian rules,
- some numerical experiments,
- some routines that implement Gauss-Kronrod rules and anti-Gaussian rules for computing integrals and have some error estimates.

For details, see e.g. [1, p.95], [2].

Definition (Weight function)

Let $w : (a, b) \to \mathbb{R}$ be a function such that

- w è is nonnegative in (a, b) (not necessarily bounded),
- $\int_a^b |x|^n \, w(x) \, dx < +\infty$ for all $n \in \mathbb{N}$;
- $\int_a^b g(x) \, w(x) \, dx = 0$ for some continuous and nonegative function g implies $g \equiv 0$ in (a, b).

Such a w is named weight function.

Example

The classical weight functions are

1. $w(x) = 1$ with $x \in [-1, 1]$ (Legendre weight);
2. $w(x) = \frac{1}{\sqrt{1-x^2}}$ con $x \in (-1, 1)$ (Chebyshev weight);
3. $w(x) = (1 - x^2)^{\gamma-(1/2)}$ con $x \in (-1, 1)$, $\gamma > (-1/2)$ (Gegenbauer weight);
4. $w(x) = (1 - x)^{\alpha} \cdot (1 + x)^{\beta}$ con $x \in (-1, 1)$, $\alpha > -1$, $\beta > -1$ (Jacobi weight);
5. $w(x) = \exp(-x)$ con $x \in (0, +\infty)$ (Laguerre weight);
6. $w(x) = \exp(-x^2)$ con $x \in (-\infty, +\infty)$ (Hermite weight);

Definition (Orthogonal polynomials)

A triangular family of polynomials $\{\phi_k\}_{k=0,\ldots,n}$ (that is $\deg(\phi_k) = k$) is orthogonal w.r.t. the weight function w on its reference interval (a, b) if and only if

$$(\phi_i, \phi_j)_{2,w} := \int_a^b \phi_i(x)\phi_j(x)w(x)dx = c_i\,\delta_{i,j}, \quad i, j = 0, \ldots, n$$

where

- $\delta_{i,j}$ is the Kronecker delta,
- $c_i > 0$, $i = 0, \ldots, n$.

The family is orthonormal if $c_i = 1$ for $i = 0, \ldots, n$.

Historical note.

- Orthogonal polynomials were already known by Legendre, Laplace (1810) and Jacobi (1826).
- The term orthogonal was introduced by Schmidt in 1905. In particular the term orthogonal polynomials appears in Szëgo (1918-1919).
- The Jacobi weight

$$w(x) = (1-x)^{\alpha} \cdot (1+x)^{\beta}, \quad x \in (-1,1),\ \alpha > -1, \beta > -1$$

  was introduced by Jacobi (1859) and studied by Mehler in numerical cubature (1864).
- The Laguerre weight function was introduced by this scientist in 1879.
- The Hermite weight function was investigated in 1864, though already known to Lagrange (1762), Abel (1826), Murphy (1835), Chebyshev (1859) and the relative orthogonal polynomials used by Laplace (1810).

In this framework it is fundamental the following theorem.

Teorema (Zeros of orthogonal polynomials, Christoffel 1877)

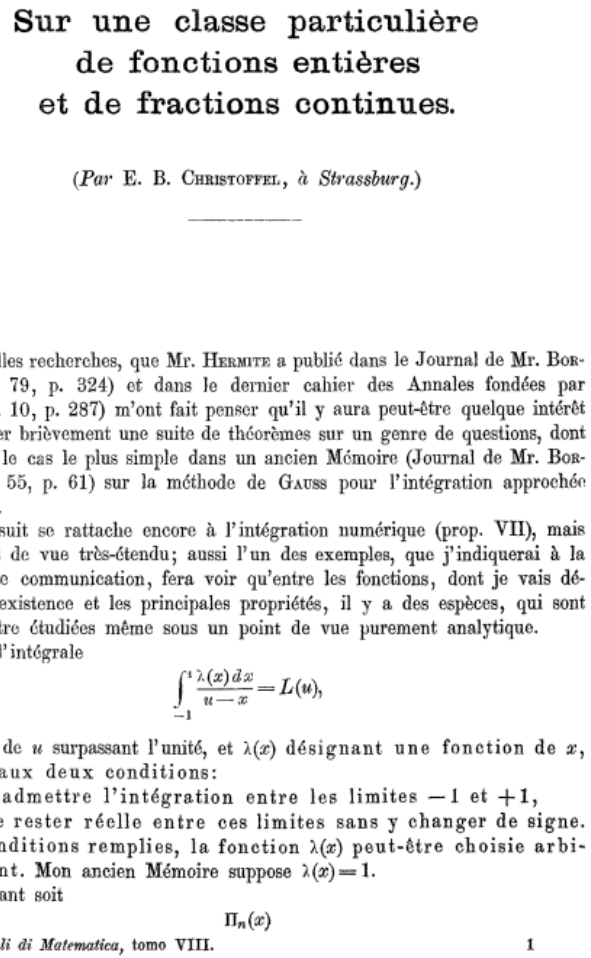
Let $\{\phi_k\}_{k=0,\dots,n}$ be a triangular family of orthogonal polynomials in (a, b) w.r.t. the weight function "w". The zeros of ϕ_n

- *are exactly n,*
- *they have multiplicity 1,*
- *belong to the open interval (a, b).*

Remark (Historical)

This theorem is due to E.B. Christoffel and appears in the work Sur une classe particuliere de fonctions entieres et de fractions continues, limited to the case of bounded intervals.

In the theorem he also considered the interlacing property of the zeros of ϕ_n w.r.t. ϕ_{n-1}.



Sur une classe particulière de fonctions entières et de fractions continues.

(*Par* E. B. Christoffel, *à Strassburg.*)

Les belles recherches, que Mr. Hermite a publié dans le Journal de Mr. Borchardt (t. 79, p. 324) et dans le dernier cahier des Annales fondées par Clebsch (t. 10, p. 287) m'ont fait penser qu'il y aura peut-être quelque intérêt de présenter brièvement une suite de théorèmes sur un genre de questions, dont j'ai traité le cas le plus simple dans un ancien Mémoire (Journal de Mr. Borchardt, t. 55, p. 61) sur la méthode de Gauss pour l'intégration approchée numérique.

Ce qui suit se rattache encore à l'intégration numérique (prop. VII), mais d'un point de vue très-étendu; aussi l'un des exemples, que j'indiquerai à la fin de cette communication, fera voir qu'entre les fonctions, dont je vais démontrer l'existence et les principales propriétés, il y a des espèces, qui sont dignes d'être étudiées même sous un point de vue purement analytique.

Posons l'intégrale

$$\int_{-1}^{+1} \frac{\lambda(x)\,dx}{u-x} = L(u),$$

le module de u surpassant l'unité, et $\lambda(x)$ désignant une fonction de x, soumise aux deux conditions:

A) d'admettre l'intégration entre les limites -1 et $+1$,

B) de rester réelle entre ces limites sans y changer de signe.

Ces conditions remplies, la fonction $\lambda(x)$ peut-être choisie arbitrairement. Mon ancien Mémoire suppose $\lambda(x)=1$.

Maintenant soit

$$\Pi_n(x)$$

Annali di Matematica, tomo VIII. 1

Figure: First page of Sur une classe particuliere de fonctions entieres et de fractions continues.

Definition (Monic polynomial)

A polynomial $p_n(x) = \sum_{k=0}^{n} a_k x^k$ is monic if $a_n = 1$.

Theorem (Three terms recursion (Christoffel 1877, Darboux 1878, Stieltjes 1884))

Let $\{\phi_k\}_{k=0,\dots,n}$ be a triangular family of monic polynomial, orthogonal in (a, b) w.r.t. a weight function $w : (a, b) \to$ 'R.

Let $\phi_{-1}(x) = 0$, $\phi_0(x) = 1$, Then for $n \geq 1$

$$\phi_{n+1}(x) = (x - \alpha_n)\phi_n(x) - \beta_n \phi_{n-1}(x)$$

where

$$\alpha_n = \frac{(x\phi_n, \phi_n)_{2,w}}{(\phi_n, \phi_n)_{2,w}}, \quad \beta_n = \frac{(\phi_n, \phi_n)_{2,w}}{(\phi_{n-1}, \phi_{n-1})_{2,w}}.$$

Observe that

- given ϕ_0 and ϕ_1, the procedure determines the triangular family of orthogonal monic polynomials of degree $n+1$, as soon as the coefficients α_k, β_k, $k = 0, \ldots, n$ are available,
- we required some information on orthogonal polynomials (i.e.they are monic), if ϕ_n is such that $(\phi_n, \phi_k) = 0$ for $k = 0, \ldots, n-1$ then for $\tau \neq 0$ also $\tilde{\phi}_n = \tau\phi_n$ is such that

$$(\tilde{\phi}_n, \phi_k) = (\tau\phi_n, \phi_k) = \tau(\phi_n, \phi_k) = 0, \;\; k = 0, \ldots, n-1$$

  is an orthogonal polynomial of degree n.
- The case in which the family of orthogonal polynomials $\{\hat{\phi}_k\}_{k=0,\ldots,n}$ is asked to be orthonormal can be derived by that in monic form.

Gaussian rules

Problema. (Formulas with n nodes and ADE equal to $2n-1$)

Do exist $x_1, \ldots, x_n$ and weights $w_1, \ldots, w_n$ (the so called Gauss-weight name) for which the relative quadrature rules of interpolatory type have ADE $\delta = 2n-1$, that is they compute exactly $\int_a^b p(x)w(x)\,dx$ for any polynomial p whose degree is inferior or equal to $2n-1$?

Teorema (Existence and uniqueness of Gaussian rules (Jacobi, 1826))

For each $n \geq 1$ exist and are unique $x_1, \ldots, x_n$ and weights $w_1, \ldots, w_n$ such that the degree of exactness of the relative rule is $2n-1$, that is

$$I_w(p_{2n-1}) := \int_a^b p_{2n-1}(x)w(x)\,dx = \sum_{k=1}^{n} w_k p_{2n-1}(x_k), \quad p_{2n-1} \in \mathbb{P}_{2n-1}$$

The nodes are the zeros of an orthogonal polynomial of degree n (w.r.t. w),

$$\phi_n(x) = A_n \cdot (x - x_1) \cdot \ldots \cdot (x - x_n)$$

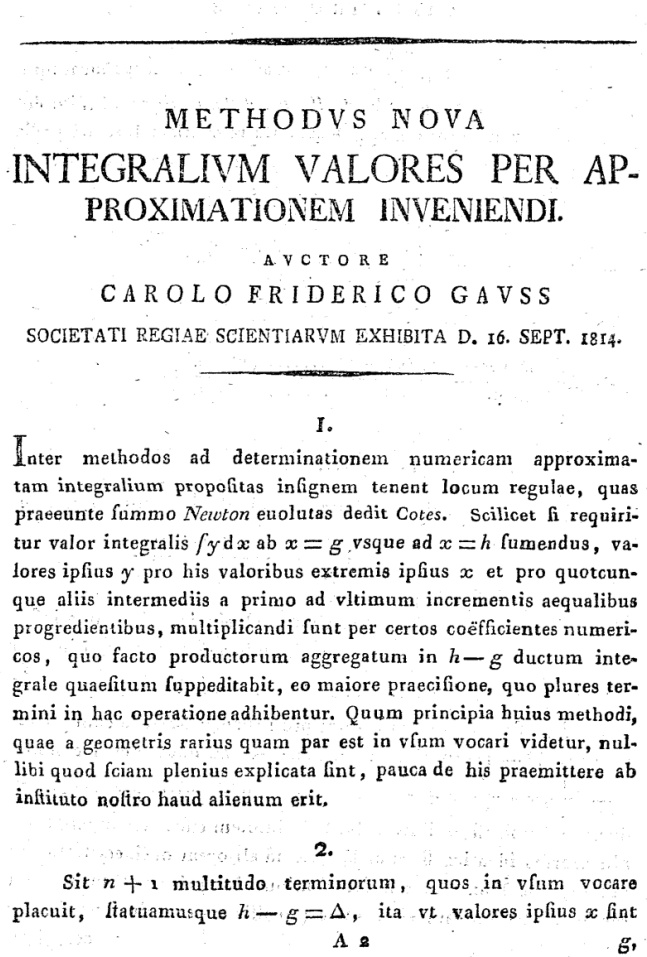
and the weights are

$$w_i = \int_a^b L_i(x)w(x)dx = \int_a^b L_i^2(x)w(x)dx > 0, \quad i = 1, \ldots, n$$

Historical note.

- *Gauss in Methodus nova integralium valores per approximationem inveniendi proved the existence of Gaussian rules in* $[-1, 1]$*, for* $w(x) = 1$ *using continuous fractions (1814).*
- *Jacobi in Ueber Gauss' neue Methode, die Werthe der Integrale näherungsweise zu finden with his clarity and simplicity, derived the previous result on the basis of polynomial divisibility arguments. The central concept that emerges in Jacobi's work is orthogonality. It is curious that formulas for Jacobi weight were introduced by Mehler in 1864.*
- *From Gautschi, Gauss quadrature rules on infinite intervals appear first in Radau (1883), who considers* $w(x) = \exp(-x)$ *in* $[0, +\infty)$ *and in Gourier (1883) who studied* $w(x) = \exp(-x^2)$ *in* $(-\infty, \infty)$*.*
- *This and many other historical details can be found in A Survey of Gauss-Christoffel Quadrature Formulae by W. Gautschi.*

Historical note.



METHODVS NOVA
INTEGRALIVM VALORES PER APPROXIMATIONEM INVENIENDI.

AVCTORE
CAROLO FRIDERICO GAVSS

SOCIETATI REGIAE SCIENTIARVM EXHIBITA D. 16. SEPT. 1814.

I.

Inter methodos ad determinationem numericam approximatam integralium propoſitas inſignem tenent locum regulae, quas praeeunte ſummo *Newton* euolutas dedit *Cotes*. Scilicet ſi requiritur valor integralis $\int y\,dx$ ab $x = g$ vsque ad $x = h$ ſumendus, valores ipſius y pro his valoribus extremis ipſius x et pro quotcunque aliis intermediis a primo ad vltimum incrementis aequalibus progredientibus, multiplicandi ſunt per certos coëfficientes numericos, quo facto productorum aggregatum in $h - g$ ductum integrale quaeſitum ſuppeditabit, eo maiore praeciſione, quo plures termini in hac operatione adhibentur. Quum principia huius methodi, quae a geometris rarius quam par est in vſum vocari videtur, nullibi quod ſciam plenius explicata ſint, pauca de his praemittere ab inſtituto noſtro haud alienum erit.

2.

Sit $n + 1$ multitudo terminorum, quos in vſum vocare placuit, ſtatuamusque $h - g = \Delta$, ita vt valores ipſius x ſint

Figure: First page of Methodus nova integralium valores per approximationem inveniendi.

Historical note.

UEBER GAUSS' NEUE METHODE,
DIE WERTHE DER INTEGRALE NÄHERUNGSWEISE ZU FINDEN.

Crelle Journal für die reine und angewandte Mathematik, Bd. 1 p. 301—308.

1.

In den *Principiis* von Newton liest man eine Methode, wie man durch eine Anzahl gegebener Punkte eine parabolische Curve legen könne. Diese Aufgabe erscheint analytisch als Interpolationsproblem, aus mehreren Gliedern einer Reihe das allgemeine zu finden. Es ist der bekanntere Fall, wenn die Intervalle der Ordinaten der gegebenen Punkte gleich grofs sind oder, analytisch ausgedrückt, wenn die Werthe des reihenden Elements, für welche auch die Werthe der entsprechenden Glieder der Reihe gegeben sind, eine arithmetische Progression bilden. Aber der elegante, mit Unrecht weniger gekannte Algorithmus, den Newton giebt, erstreckt sich schon auf den allgemeineren Fall, wenn jene Intervalle der Ordinaten der gegebenen Punkte oder jene Werthe des reihenden Elements ganz beliebige sind. Newton hat hiervon eine Anwendung auf die Quadraturen gemacht. Durch mehrere Punkte der zu quadrirenden Curve, für welche die Ordinaten berechnet worden sind, legt er die parabolische Curve, und deren Quadratur zwischen denselben Grenzen, zwischen denen die gegebene Curve quadrirt werden sollte, giebt einen Näherungswerth.

Newton hat von jenem Interpolationsproblem und seiner Anwendung auf die Quadraturen ferner in einem Tractätchen gehandelt, welches *Methodus*

1*

Figure: First page of Ueber Gauss' neue Methode, die Werthe der Integrale näherungsweise zu finden by C. Jacobi.

Gaussian rules in Matlab

With the exception of few cases (as in the case of Chebyshev weight function), the nodes and the weights of a Gaussian rule are not explicit.

- Many years ago they were listed in specific manuals. They were computed by hand and later listed in cards. To this purpose in [6] the author writes

  Hand calculations led the way for over a century.
  Tallquist (1905), Moors (1905), Nyström (1930), and Bayly (1938) used fountain pens and dogged determination to calculate the quadrature nodes for $n \leq 12$.
  Eventually, with presumably a small army of human calculators Lowan, Davids, and Levenson (1942) tabulated the nodes and weights for $n = 1, \ldots, 16$ *for the Mathematical Tables Project.*

  For the original papers, see

  1. Table of the zeros of the Legendre polynomials of order 1-16 and the weight coefficients for Gauss' mechanical quadrature formula
  2. Mathematical Tables Project (U.S.)

The accompanying table computed by the Mathematical Tables Project gives the roots x_i for each $P_n(x)$ up to $n=16$, and the corresponding weight coefficients a_i, to 15 decimal places.

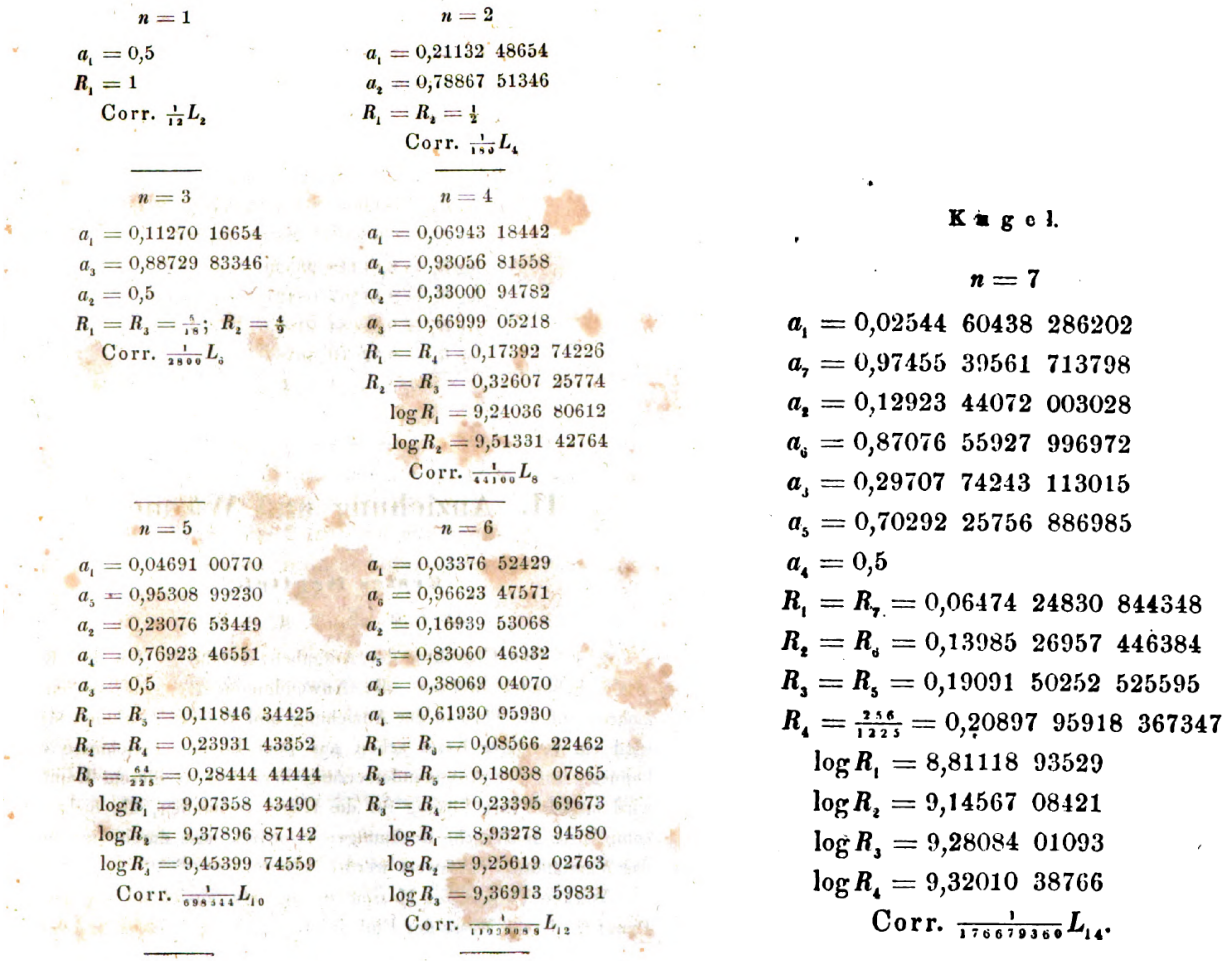
The first such table, computed by Gauss gave 16 places up to $n=7$.[3] More recently work was done by Nyström,[4] who gave 7 decimals up to $n=10$, but for the interval $(-1/2, +1/2)$. B. de F. Bayly has given the roots and coefficients of $P_{12}(x)$ to 13 places.[5]

Figure: An interesting comment on the numerical work by Gauss.

The citation refers to

- E. Heine Handbuch der Kugelfunctionen, vol. 2, 1881, p. 15 (but they are at pp. 295-296.),
- E.W. Hobson, Spherical Harmonics, pp. 80-81.

Gaussian rules in Matlab



$n = 1$

$a_1 = 0{,}5$
$R_1 = 1$
Corr. $\frac{1}{12} L_2$

$n = 2$

$a_1 = 0{,}21132\ 48654$
$a_2 = 0{,}78867\ 51346$
$R_1 = R_2 = \frac{1}{2}$
Corr. $\frac{1}{150} L_4$

$n = 3$

$a_1 = 0{,}11270\ 16654$
$a_3 = 0{,}88729\ 83346$
$a_2 = 0{,}5$
$R_1 = R_3 = \frac{5}{18}$; $R_2 = \frac{4}{9}$
Corr. $\frac{1}{2800} L_6$

$n = 4$

$a_1 = 0{,}06943\ 18442$
$a_4 = 0{,}93056\ 81558$
$a_2 = 0{,}33000\ 94782$
$a_3 = 0{,}66999\ 05218$
$R_1 = R_4 = 0{,}17392\ 74226$
$R_2 = R_3 = 0{,}32607\ 25774$
$\log R_1 = 9{,}24036\ 80612$
$\log R_2 = 9{,}51331\ 42764$
Corr. $\frac{1}{44100} L_8$

$n = 5$

$a_1 = 0{,}04691\ 00770$
$a_5 = 0{,}95308\ 99230$
$a_2 = 0{,}23076\ 53449$
$a_4 = 0{,}76923\ 46551$
$a_3 = 0{,}5$
$R_1 = R_5 = 0{,}11846\ 34425$
$R_2 = R_4 = 0{,}23931\ 43352$
$R_3 = \frac{64}{225} = 0{,}28444\ 44444$
$\log R_1 = 9{,}07358\ 43490$
$\log R_2 = 9{,}37896\ 87142$
$\log R_3 = 9{,}45399\ 74559$
Corr. $\frac{1}{698544} L_{10}$

$n = 6$

$a_1 = 0{,}03376\ 52429$
$a_6 = 0{,}96623\ 47571$
$a_2 = 0{,}16939\ 53068$
$a_5 = 0{,}83060\ 46932$
$a_3 = 0{,}38069\ 04070$
$a_4 = 0{,}61930\ 95930$
$R_1 = R_6 = 0{,}08566\ 22462$
$R_2 = R_5 = 0{,}18038\ 07865$
$R_3 = R_4 = 0{,}23395\ 69673$
$\log R_1 = 8{,}93278\ 94580$
$\log R_2 = 9{,}25619\ 02763$
$\log R_3 = 9{,}36913\ 59831$
Corr. $\frac{1}{1102995} L_{12}$

Kegel.

$n = 7$

$a_1 = 0{,}02544\ 60438\ 286202$
$a_7 = 0{,}97455\ 39561\ 713798$
$a_2 = 0{,}12923\ 44072\ 003028$
$a_6 = 0{,}87076\ 55927\ 996972$
$a_3 = 0{,}29707\ 74243\ 113015$
$a_5 = 0{,}70292\ 25756\ 886985$
$a_4 = 0{,}5$
$R_1 = R_7 = 0{,}06474\ 24830\ 844348$
$R_2 = R_6 = 0{,}13985\ 26957\ 446384$
$R_3 = R_5 = 0{,}19091\ 50252\ 525595$
$R_4 = \frac{256}{1225} = 0{,}20897\ 95918\ 367347$
$\log R_1 = 8{,}81118\ 93529$
$\log R_2 = 9{,}14567\ 08421$
$\log R_3 = 9{,}28084\ 01093$
$\log R_4 = 9{,}32010\ 38766$
Corr. $\frac{1}{176679360} L_{14}$.

Figure: Gaussian rules listed by Heine (notice they are in $(0,1)$ and not $(-1,1)$).

Gaussian rules in Matlab

$n = 15$

$x_0 = 0.000000000000000$	$a_0 = 0.202578241925561$
$x_1 = 0.201194093997435$	$a_1 = 0.198431485327111$
$x_2 = 0.394151347077563$	$a_2 = 0.186161000015562$
$x_3 = 0.570972172608539$	$a_3 = 0.166269205816994$
$x_4 = 0.724417731360170$	$a_4 = 0.139570677926154$
$x_5 = 0.848206583410427$	$a_5 = 0.107159220467172$
$x_6 = 0.937273392400706$	$a_6 = 0.070366047488108$
$x_7 = 0.987992518020485$	$a_7 = 0.030753241996117$

$n = 16$

$x_1 = 0.095012509837637$	$a_1 = 0.189450610455069$
$x_2 = 0.281603550779259$	$a_2 = 0.182603415044924$
$x_3 = 0.458016777657227$	$a_3 = 0.169156519395003$
$x_4 = 0.617876244402644$	$a_4 = 0.149595988816577$
$x_5 = 0.755404408355003$	$a_5 = 0.124628971255534$
$x_6 = 0.865631202387832$	$a_6 = 0.095158511682493$
$x_7 = 0.944575023073233$	$a_7 = 0.062253523938648$
$x_8 = 0.989400934991650$	$a_8 = 0.027152459411754$

Figure: Gaussian rules listed in 1944, having 15 and 16 points. Notice that also $-x_k$ must be taken with the same weight!

Gaussian rules in Matlab

- In 1969 Golub and Welsch introduced in Calculation of Gauss Quadrature Rules a novel algorithm for the computation of the nodes and the weights. Walter Gautschi and collaborators wrote a collection of codes for the computation of Gaussian rules for a large variety of weight functions.

  The paper contained a microfiche section with ALGOL60 procedures for performing the algorithms `GAUSSQUADRATURERULE`, `GENORTHOPOLY`, `CLASSICORTHOPOLY`.

- In his Computation of Gauss-type quadrature formulas, D. Laurie comments

  The timeless fact about the Golub–Welsch algorithm is that any advance in our ability to solve the symmetric tridiagonal eigenproblem immediately implies a corresponding advance in our ability to compute Gaussian formulas.

  In other words, linear algebra has a key role in computing the Gaussian rule.

Calculation of Gauss Quadrature Rules*

By Gene H. Golub** and John H. Welsch

Abstract. Several algorithms are given and compared for computing Gauss quadrature rules. It is shown that given the three term recurrence relation for the orthogonal polynomials generated by the weight function, the quadrature rule may be generated by computing the eigenvalues and first component of the orthornormalized eigenvectors of a symmetric tridiagonal matrix. An algorithm is also presented for computing the three term recurrence relation from the moments of the weight function. ■

Gaussian rules in Matlab

- In last years there have been an increasing interest on algorithms for the computation of Gaussian rules up to several thousands points (see, e.g., Fast and Accurate Computation of Gauss-Legendre and Gauss-Jacobi Quadrature Nodes and Weights by N. Hale and A. Townsend and references therein). Specific routines from Chebfun package are legpots.m jacpts.m, lagpts.m, hermpts.m.
- In 2014 Bogaert presented explicit asymptotic formulas for the Gauss–Legendre quadrature weights and nodes, which allows for computation of nodes and weights for values of n exceeding one billion in less than 20 seconds, see Iteration-free computation of Gauss-Legendre quadrature nodes and weights.

Gaussian rules in Matlab

Just to give an idea of these developments, consider this demo.

```
function demo_legpts

NV=10.^(1:8);
ntest=10;

timeA=[];
for N=NV
    timeV=[];
    for j=1:ntest
        tic;
        [X,W]=legpts(N);
        tt=toc;
        timeV(end+1)=tt;
    end
    timeA(end+1)=exp(mean(log(timeV)));
    fprintf('\n \t n: %-1.2e time: %1.3e',N,timeA(end));
end

fprintf('\n');
```

As results we get the average cputime for computing "n "nodes.

```
>> demo_legpts
   n: 1.00e+01 time: 1.684e-04
   n: 1.00e+02 time: 2.765e-04
   n: 1.00e+03 time: 1.713e-04
   n: 1.00e+04 time: 2.267e-04
   n: 1.00e+05 time: 2.417e-03
   n: 1.00e+06 time: 2.994e-02
   n: 1.00e+07 time: 2.810e-01
   n: 1.00e+08 time: 2.963e+00
>>
```

Now we briefly introduce the classical approach by Golub and Welsch, in the implementation by W. Gautschi, that has the advantage of being used for a large class of weight functions.

As example, first we compute *n recurrence coefficients*, relatively to the Jacobi weight function $w(x) = (1-x)^a (1+x)^b$, by the routine r_jacobi.m whose typical call is

```
ab=r_jacobi(n,a,b)
```

Note that

- a, b are the exponents of Jacobi weight function (and not the interval extrema!);
- the recurrence coefficients are stored in the matrix ab and
  1. the first column of ab contains the terms $\alpha_0, \ldots, \alpha_{n-1}$,
  2. the second of ab contains the terms $\beta_0, \ldots, \beta_{n-1}$.

Gaussian rules in Matlab

Next one calls gauss.m as

```
xw=gauss(N,ab)
```

defined as

```
function xw=gauss(N,ab)
N0=size(ab,1); if N0<N, error('input array ab too short'), end
J=zeros(N);
for n=1:N, J(n,n)=ab(n,1); end
for n=2:N
  J(n,n-1)=sqrt(ab(n,2));
  J(n-1,n)=J(n,n-1);
end
[V,D]=eig(J);
[D,I]=sort(diag(D));
V=V(:,I);
xw=[D ab(1,2)*V(1,:)'.^2];
```

- This routine, based on the computation of eigenvalues and eigenvectors, for n equal to the number of rows of `ab` provides a matrix `xw` in $\mathbb{R}^{n\times 2}$ in which
  - the first column stores the n nodes,
  - the second column stores the n weights,

  of the Gaussian rule relatively to the choosen Jacobi-weight.
- For details see Algorithm 726: ORTHPOL–a package of routines for generating orthogonal polynomials and Gauss-type quadrature rules.

Now we can assembly these routines to approximate integrals w.r.t. Jacobi-weights by integration_gauss_jacobi.m,

```
function [I,x,w]=integration_gauss_jacobi(N,...
                                          alpha,beta,f)
% INPUT:
% N: number of nodes.
% alpha,beta: w(x)=((1-x)^alpha)*((1+x)^beta)
% f: function such that the integrand is f(x) w(x) in
    (-1,1).
% OUTPUT:
% I: approximation of I(f*w,a,b)
% x,w: nodes and weights of the rule.
ab=r_jacobi(N,alpha,beta); % recursive terms
xw=gauss(N,ab); % matrix of nodes and weights
x=xw(:,1); w=xw(:,2);% nodes and weights
fx=feval(f,x); % function evaluation
I=w'*fx; % value of the integral
```

Gaussian rules in Matlab: example 1

As first test we consider

$$\int_{-1}^{1} x^{20}\, dx \approx 0.095238095238095233 00$$

obtaining

```
>> format long e;
>> N=11; ajac=0; bjac=0; a=-1; b=1; g=@(x) x.^20;
>> [I_jac,x_jac,w_jac]=integration_gauss_jacobi(N,ajac
    ,bjac,g);
>> I_jac
I_jac =
     9.523809523809568e-02
>> % RELATIVE ERROR: 4.662936703425657e-15
```

Note that the rule has degree of exactness 21 so it integrates exactly a polynomial of degree 20.

Gaussian rules in Matlab: example 2

Let us consider the integral

$$\int_{-1}^{1} \exp(x)\sqrt{1-x}\,dx = 1.779143654691910. \tag{1}$$

To achieve this result we can use for instance Chebfun suite or Matlab built in `integral`, getting

```
>> format long e;
>> f=@(x) exp(x).*(1-x).^(1/2);
>> F=chebfun(f,'splitting','on'); S=sum(F)
S = 1.779143654691910e+00
>> I=integral(f,-1,1,'AbsTol',10^(-15),'RelTol'
    ,10^(-15))
I = 1.779143654691910e+00
```

Esempio 2

Now observe that $w(x) = \sqrt{1-x}$ is a Gauss-Jacobi weight

$$w(t) = (1-t)^{\alpha}(1+t)^{\beta}$$

where $\alpha = 1/2$ and $\beta = 0$.

Consequently, if

- $g(x) = \exp(x),\ \ w_1(x) = \sqrt{1-x}$,
- $f(x) = \exp(x),\ \ w_2(x) = 1$,

then

$$\int_{-1}^{1} \exp(x)\sqrt{1-x}dx = \int_{-1}^{1} g(x)w_1(x)dx = \int_{-1}^{1} f(x)w_2(x)dx$$

Thus the integrand can be written as $f \cdot w_1$ where $f(x) = \exp(x)$ and $w_1(x) = \sqrt{1-x}$.

Gaussian rules in Matlab: example 2

```
>> a=-1; b=1;
>> % GAUSS-JACOBI
>> [I_jac,x_jac,w_jac]=integration_gauss_jacobi(10,1/2,0,
    @exp);
>> fprintf('\n \t [GAUSS-JACOBI]: %15.20f',I_jac);
   [GAUSS-JACOBI]: 1.77914365469190930000
>> % ABSOLUTE ERROR: 4.440892098500626e-016
>> [I_jac,x_jac,w_jac]=...
 integration_gauss_jacobi(10,0,0,inline('exp(x).*sqrt(1-x)')
     );
>> fprintf('\n \t [GAUSS-LEGENDRE]: %15.20f',I_jac)
   [GAUSS-LEGENDRE]: 1.77984112101478020000
>> % ABSOLUTE ERROR: 6.9747e-004
```

The formulas above have the same cardinality, but give different errors

- circa $4.44 \cdot 10^{-16}$ per Gauss-Jacobi con $a = 1/2$, $b = 0$,
- circa $6.97 \cdot 10^{-4}$ per Gauss-Legendre (cioè Gauss-Jacobi con $a = 0$, $b = 0$).

The key is that using the right weight function w, with a regular f, gives better results than using a less regular integrand fw w.r.t. to Gauss-Legendre weight.

Gauss-Kronrod rules

Fixed an n-point Gaussian rule, the purpose of Gauss-Kronrod rules is to add $n+1$ points to achieve a rule with

$$ADE = \begin{cases} 3n+1 \text{ if n is even} \\ 3n+2 \text{ if n is odd} \end{cases}$$

(see, e.g. [1, p. 106]).

- The price that one pays is the full rule in general does not preserve for the first n points the weights. The idea is that with few function evaluations, i.e. $2n+1$, one has two rules with different order, both providing an approximation of the integral as well as an error estimate.
- The codes are a complicated modification of the Golub-Welsch approach, and were introduced in Calculation of Gauss-Kronrod quadrature rules by D.P. Laurie.
- Kronrod extensions exist not only for the usual Gauss-Legendre weight but also for the Gauss-Gegenbauer weight function $w(x) = (1-x^2)^\mu$ with $-1/2 \leq \mu \leq 3/2$.

Our purpose is to introduce the routines to run the numerical experiments.
We start by its core, that is the procedure r_kronrod.
D. Laurie comments his code as follows:

- `ab=R_KRONROD(n,ab0)` produces the alpha- and beta-elements in the Jacobi-Kronrod matrix of order $2n+1$ for the weight function (or measure) w.
- The input data for the weight function w are the recurrence coefficients of the associated orthogonal polynomials, which are stored in the array `ab0` of dimension

$$\lceil (3n/2) + 1 \rceil \times 2.$$

- The alpha-elements are stored in the first column, the beta-elements in the second column, of the

$$(2n+1) \times 2$$

array `ab`.

Gauss-Kronrod rules: Matlab implementation

The routine r_kronrod consists in this code.

```
function ab=r_kronrod(N,ab0)

if length(ab0)<ceil(3*N/2)+1, error('array ab0 too short'), end
a=zeros(2*N+1,1); b=a;
k=0:floor(3*N/2); a(k+1)=ab0(k+1,1);
k=0:ceil(3*N/2); b(k+1)=ab0(k+1,2);
s=zeros(floor(N/2)+2,1); t=s; t(2)=b(N+2);
for m=0:N-2,
  k=floor((m+1)/2):-1:0; l=m-k;
  s(k+2)=cumsum((a(k+N+2)-a(l+1)).*t(k+2)+b(k+N+2).*s(k+1)-b(l+1).*s(k+2));
  swap=s; s=t; t=swap;
end
j=floor(N/2):-1:0; s(j+2)=s(j+1);
for m=N-1:2*N-3,
  k=m+1-N:floor((m-1)/2); l=m-k; j=N-1-l;
  s(j+2)=cumsum(-(a(k+N+2)-a(l+1)).*t(j+2)-b(k+N+2).*s(j+2)+b(l+1).*s(j+3));
  j=j(length(j)); k=floor((m+1)/2);
  if rem(m,2)==0, a(k+N+2)=a(k+1)+(s(j+2)-b(k+N+2)*s(j+3))/t(j+3);
  else b(k+N+2)=s(j+2)/s(j+3);
  end
  swap=s; s=t; t=swap;
end
a(2*N+1)=a(N)-b(2*N+1)*s(2)/t(2);
ab=[a b];
```

The previous routine is used by kronrod to provide the desired rule. D. Laurie provides these comment to the routine.

- `xw=KRONROD(n,ab)` generates the $(2n+1)$-point Gauss-Kronrod quadrature rule for the weight function w encoded by the recurrence matrix ab of order

$$\lceil (3n/2) + 1 \rceil \times 2$$

containing in its first and second column respectively the alpha- and beta-coefficients in the three-term recurrence relation for w.

- The $2n+1$ nodes, in increasing order, are output into the first column, the corresponding weights into the second column, of the

$$(2n+1) \times 2$$

array `xw`.

Gauss-Kronrod rules

Below you find the Matlab code of kronrod.

```
function xw=kronrod(N,ab)

ab0=r_kronrod(N,ab);
if(sum((ab0(:,2)>0))~=2*N+1) error('Gauss-Kronrod does not exist'), end
J=zeros(2*N+1);
for k=1:2*N
    J(k,k)=ab0(k,1);
    J(k,k+1)=sqrt(ab0(k+1,2));
    J(k+1,k)=J(k,k+1);
end
J(2*N+1,2*N+1)=ab0(2*N+1,1);
[V,D]=eig(J);
d=diag(D);
e=ab0(1,2).*(V(1,:).^2);
[x,i]=sort(d);
w=e(i)';
xw=[x w];
```

Gauss-Kronrod rules: demo

To see how these rules work we consider the demo demo_kronrod.

```
function demo_kronrod(f,alpha,beta,N)

% Integration of a function 'f' in [-1,1] and Gauss-Kronrod rules.
if nargin < 1, f=@(x) exp(x).*sqrt(1-x.^2); end
if nargin < 3, alpha=0; beta=0; end % Jacobi weight
if nargin < 4, N=10; end

dim=ceil(3*N/2)+1; % length Kronrod rule
ab=r_jacobi(dim,alpha,beta); % Gauss-Legendre recurrence (as required by kronrod)
xwg=gauss(N,ab); xwk=kronrod(N,ab);

g=@(x) f(x).*(1-x).^alpha.*(1+x).^beta;
I=integral(g,-1,1,'AbsTol',10^(-15),'RelTol',10^(-15));

fk=feval(f,xwk(:,1)); % this evaluation includes that at Gauss-Legendre nodes!

Sg=(xwg(:,2))'*fk(2:2:end);
Sk=(xwk(:,2))'*fk;

fprintf('\n * Gauss-Legendre rule \n \n'); xwg
fprintf('\n * Gauss-Legendre-Kronrod rule \n \n'); xwk
fprintf('\n \t I gauss : %1.15e',Sg);
fprintf('\n \t I kronr : %1.15e',Sk);
fprintf('\n \t I exact : %1.15e',I);
fprintf('\n \n');
fprintf('\n \t E gauss : %1.3e',abs(I-Sg));
fprintf('\n \t E estim : %1.3e',abs(Sk-Sg));
fprintf('\n \n');
```

Gauss-Kronrod rules: demo

We run this first experiment.

```
>> clear all
>> f=@(x) exp(x).*sqrt(1-x.^2); alpha=0; beta=0; N=5;
>> demo_kronrod(f,alpha,beta,N);

 * Gauss-Legendre rule
xwg =
   -9.0618e-01   2.3693e-01
   -5.3847e-01   4.7863e-01
    4.1434e-17   5.6889e-01
    5.3847e-01   4.7863e-01
    9.0618e-01   2.3693e-01

 * Gauss-Legendre-Kronrod rule
xwk =
   -9.8409e-01   4.2582e-02
   -9.0618e-01   1.1523e-01
   -7.5417e-01   1.8680e-01
   -5.3847e-01   2.4104e-01
   -2.7963e-01   2.7285e-01
   -1.9783e-16   2.8299e-01
    2.7963e-01   2.7285e-01
    5.3847e-01   2.4104e-01
    7.5417e-01   1.8680e-01
    9.0618e-01   1.1523e-01
    9.8409e-01   4.2582e-02

         I gauss : 1.783762504838484e+00
         I kronr : 1.775930588360792e+00
         I exact : 1.775499689212181e+00

         E gauss : 8.263e-03
         E estim : 7.832e-03
```

We observe that

- nodes of Gauss-Legendre rule are nested in those of Gauss-Kronrod rule, though the weights are different;
- the error estimate is quite good though the error is large;
- one can use a better Gegenbauer weight function that is

$$(1-x^2)^{1/2} = (1-x)^{1/2}(1+x)^{1/2}, \quad x \in (-1,1).$$

  and so one intends to make experiments with this new weight function, setting the Jacobi exponents as $\alpha = 1/2$, $\beta = 1/2$ and $f(x) = \exp(x)$.
- it is proven that this weight function admits Kronrod extension.

Gauss-Kronrod rules: demo

Thus we run this second experiment.

```
>> clear all
>> f=@(x) exp(x); alpha=1/2; beta=1/2; N=5;
>> demo_kronrod(f,alpha,beta,N);

 * Gauss-Legendre rule
xwg =
   -8.6603e-01   1.3090e-01
   -5.0000e-01   3.9270e-01
   -5.1486e-17   5.2360e-01
    5.0000e-01   3.9270e-01
    8.6603e-01   1.3090e-01

 * Gauss-Legendre-Kronrod rule
xwk =
   -9.6593e-01   1.7537e-02
   -8.6603e-01   6.5450e-02
   -7.0711e-01   1.3090e-01
   -5.0000e-01   1.9635e-01
   -2.5882e-01   2.4426e-01
   -3.4235e-17   2.6180e-01
    2.5882e-01   2.4426e-01
    5.0000e-01   1.9635e-01
    7.0711e-01   1.3090e-01
    8.6603e-01   6.5450e-02
    9.6593e-01   1.7537e-02

         I gauss : 1.775499688781380e+00
         I kronr : 1.775499689212182e+00
         I exact : 1.775499689212181e+00

         E gauss : 4.308e-10
         E estim : 4.308e-10
```

Gauss-Kronrod rules: demo

We observe that

- nodes of the Gauss-Jacobi rule are nested in those of Gauss-Jacobi-Kronrod rule, though the weights are different;
- the error estimate is quite good but this time the error is rather small (due to the good choice of the weight function).

An alternative are the so called anti-Gaussian rules introduced in Anti-Gaussian quadrature formulas by D. Laurie.

- An anti-Gaussian quadrature formula is an $(n+1)$-point formula of degree $2n-1$ which integrates polynomials of degree up to $2n+1$ with an error equal in magnitude but of opposite sign to that of the n-point Gaussian formula.
- In the algorithm,
  1. by `r_jacobi` one determines N+1 recursive coefficients $\boldsymbol{\alpha} = (\alpha_k)_{k=1,\ldots,N+1}$, $\boldsymbol{\beta} = (\beta_k)_{k=1,\ldots,N+1}$ w.r.t. the weight w;
  2. sets $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}$,
  3. sets $\boldsymbol{\beta}^*_k = \boldsymbol{\beta}_k$ for $k = 1, \ldots, N$, $\boldsymbol{\beta}^*_{N+1} = 2\boldsymbol{\beta}_{N+1}$;
  4. computes the Gaussian rule S^G_N with N nodes as well as the anti-Gaussian S^{AG}_N rule with $N+1$ nodes;
  5. the error $I(f) - S^G_N(f)$ is equal to $-(I(f) - S^{AG}_N(f))$ and thus, easily,
$$I(f) - S^G_N(f) = (S^{AG}_N(f) - S^G_N(f))/2.$$

Anti-Gaussian rules

In the last point we said

- the error $I(f) - S_N^G(f)$ is equal to $-(I(f) - S_N^{AG}(f))$ and thus, easily,

$$I(f) - S_N^G(f) = (S_N^{AG}(f) - S_N^G(f))/2.$$

This comes from the fact that

$$\begin{cases} I(f) - S_N^G(f) = E_N \\ I(f) - S_N^{AG}(f) = -E_N \end{cases}$$

and subtracting the second equation to the first one and rearranging

$$E_N = \frac{S_N^{AG}(f) - S_N^G(f)}{2},$$

that is

$$I(f) - S_N^G(f) = E_N = \frac{S_N^{AG}(f) - S_N^G(f)}{2}.$$

Anti-Gaussian rules in Matlab

```
function demo_antigaussian(f,alpha,beta,N)

% Integration of a function 'f' in [-1,1] and Gauss-Kronrod rules.
if nargin < 1, f=@(x) exp(x); end
if nargin < 3, alpha=0; beta=0; end % Jacobi weight
if nargin < 4, N=3; end

dim=N+1; % length Kronrod rule
ab=r_jacobi(dim,alpha,beta); % Gauss-Legendre recurrence (as required by kronrod)
xwg=gauss(N,ab(1:N,:));
ab_agss=ab; ab_agss(end,2)=2*ab_agss(end,2);
xwag=gauss(N+1,ab_agss);

g=@(x) f(x).*(1-x).^alpha.*(1+x).^beta;
I=integral(g,-1,1,'AbsTol',10^(-15),'RelTol',10^(-15));

fg=feval(f,xwg(:,1)); % this evaluation includes that at Gauss-Legendre nodes!
Sg=(xwg(:,2))'*fg;

fg=feval(f,xwag(:,1)); % this evaluation includes that at anti-gauss nodes!
Sag=(xwag(:,2))'*fg;
Saver=(Sg+Sag)/2;

fprintf('\n * Gauss-Legendre rule \n \n'); xwg
fprintf('\n * Gauss-Legendre-anti rule \n \n'); xwag
fprintf('\n \t I gauss    : %1.15e',Sg);
fprintf('\n \t I agaus    : %1.15e',Sag);
fprintf('\n \t I aver     : %1.15e',Saver);
fprintf('\n \t I exact    : %1.15e',I);
fprintf('\n \n');
fprintf('\n \t I-Sg       : %1.3e',(I-Sg));
fprintf('\n \t I-Sag      : %1.3e',(I-Sag));
fprintf('\n \t I-laver    : %1.3e',(I-(Sg+Sag)/2));
fprintf('\n \t (Sag-Sg)/2 : %1.3e \n \n',(Sag-Sg)/2);
```

Anti-Gaussian rules in Matlab

Here, we test the anti-Gaussian rules with $N+1$ points on $f(x) = \exp(x)$, that though it is not a polynomial of degree $2N-1$ allows realiable error estimates (why does it happen?).

```
>> demo_antigaussian

 * Gauss-Legendre rule
xwg =
   -0.7746    0.5556
   -0.0000    0.8889
    0.7746    0.5556

 * Gauss-Legendre-anti rule
xwag =
   -0.9643    0.1998
   -0.4294    0.8002
    0.4294    0.8002
    0.9643    0.1998

   I gauss     : 2.350336928680012e+00
   I agaus     : 2.350467853389318e+00
   I aver      : 2.350402391034665e+00
   I exact     : 2.350402387287603e+00

   I-Sg        : 6.546e-05
   I-Sag       : -6.547e-05
   I-Iaver     : -3.747e-09
   (Sag-Sg)/2  : 6.546e-05
>>
```

Conclusion

We have provided a brief survey of the Gaussian-rules including some of the latest advances.

We have not included in the discussion special topics as:

- how these algorithms actually work.
- comparison with other rules as written in Is Gauss Quadrature Better than Clenshaw–Curtis?;
- integration formulas of Gauss-type with preassigned abscissas as Gauss-Radau or Gauss-Lobatto (useful in the numerical solution of ODEs), see e.g. [1, p.101];
- integration rules with derivative data.

Bibliography

- P.J. Davis and P. Rabinowitz, Methods of Numerical Integration, Dover 1984.
- W. Gautschi, Orthogonal polynomials: applications and computation, Acta Numerica (1996), pp.45–119.
- D.P. Laurie, Computation of Gauss-type quadrature formulas , Journal of Computational and Applied Mathematics, Volume 127, Issues 1–2 (2001), pp. 201–217.
- A.N. Lowan, N. Davids, A. Levenson, Table of the zeros of the Legendre polynomials of order 1-16 and the weight coefficients for Gauss' mechanical quadrature formula, Bull. Amer. Math. Soc. 48(10), pp.739–743 (October 194
Volume 127, Issues 1–2, 15 January 2001, Pages 201-217, Volume 127, Issues 1–2, 15 January 2001, Pages 201-217.2).
- The Online Books Page, Mathematical Tables Project (U.S.).
- A. Townsend, The race for high order Gauss–Legendre quadrature.