

ガウス型積分公式の分点と重みの計算

平山 弘

機械システム工学科

Calculation of Abscissas and Weight Factors for Gaussian Integration

Hiroshi HIRAYAMA

Abstract

The Gaussian quadrature formula is well known as an effective numerical integration method. But the computation of abscissas and weights for a given integration is very difficult for poor precision operation of computers. It is possible to compute adequate precision values of abscissas and weights by using the high precision system. The numerical examples are shown to use the program.

Key Words: Gaussian quadrature, C++ language, high precision, multiple precision, algebraic equation

1. はじめに

重み関数 $W(x)$ と整数 n を与えたとき、 $f(x)$ が $(2n-1)$ 次以下の多項式の時、近似式

$$\int_a^b W(x)f(x)dx \approx \sum_{i=1}^n \omega_i f(x_i) \quad (1.1)$$

が厳密になるように、重み ω_i と分点 x_i を探し出す事ができる場合がある。この重み ω_i と分点 x_i を使って計算する数値積分法をガウス(Gauss)型数値積分法と呼ばれている。上の式からわかるように、重み ω_i と分点 x_i が分かれば、その計算は単純で非常に効率的である。これらの数値積分公式は、高精度な結果を与える公式[1]として良く知られている。

ガウス型数値積分公式には、特定の分点を利用するようにした公式[2]も存在する。一つの点を固定した積分公式として、ラダー(Radau)の公式が有名である。この公式は

$$\int_{-1}^1 f(x)dx = \frac{2}{n^2} f(-1) + \sum_{i=1}^{n-1} w_i f(x_i) \quad (1.2)$$

と書ける。また、2点を固定した固定した公式として、ロバット(Lobatto)の公式が有名である。この公式は

$$\int_{-1}^1 f(x)dx = \frac{2}{n(n-1)} [f(1) + f(-1)] + \sum_{i=2}^{n-1} w_i f(x_i)$$

(1.3)

と書ける。上の一点固定および二点固定の公式は、いずれも(1.1)において、 $W(x)=1$ の最も簡単な場合で、それ以外の具体的な研究は、見受けられない。

これらの公式の問題点は、重み ω_i と分点 x_i を計算しておかなければならない点である。この計算は、通常、高次の代数方程式となる。このため、通常の精度(10進数で約15桁)で計算した場合、十分な精度が得られない場合や、計算精度不足のため、これらの計算が収束しないことがある。ガウス型数値積分公式が効率的であることがわかったとしても、具体的に、重み ω_i と分点 x_i が計算できないために、ガウス型積分公式を使えない場合や、使えたとしても、十分な次数の公式を使えない場合がよくある。

このような高次代数方程式の問題点に対して多くの研究が成されているが、高精度で計算するのが最も確実な方法と思われる。本研究では、C++言語で作成された高精度演算パッケージを使い、この問題を解決し、いろいろなガウス型数値積分公式を求めるプログラムを作成し、いろいろなガウス型積分公式を提供する。

2. ガウス型積分公式の計算

一般にガウス型数値積分公式と言うと、ガウス

ールジャンドル、ガウスラゲール、ガウスエルミートおよび上のラダー、ロバット型の積分公式が有名であるが、ここでは一般に、 m 点固定の n 次のガウス型数値積分公式を作る。

重み関数 $W(x)$ が、積分区間で、正（一定符号）であれば、ガウス型数値積分公式が存在することが知られているが、ここでは、このような仮定を設けなくて、重み関数が積分区間で符号を変えるような場合も適用できるようなアルゴリズムである代数的な方法で計算を行った。このため、解の存在が保証されていない場合も計算できる。

多くの数値解析の教科書では、直交多項式の応用として、ガウス型積分公式を導いていて、代数的方法で説明しているものが少ないので、ここで簡単に説明を行う。

$2n$ 個の関数 $f(x) = 1, x, x^2, \dots, x^{2n-1}$ に対して

$$\int_a^b \omega(x) f(x) dx = \sum_{i=1}^n \omega_i f(x_i) \quad (2.1)$$

が厳密に成り立つような $2n$ 個の数値 $\omega_1, \omega_2, \dots, \omega_n, x_1, x_2, \dots, x_n$ を求めることを考える。この式をモーメント

$$\int_a^b \omega(x) x^j dx = m_j \quad (2.2)$$

を使ってこの条件を書き下すと、次の方程式が得られる。

$$\begin{aligned} m_0 &= \omega_1 + \omega_2 + \dots + \omega_n \\ m_1 &= \omega_1 x_1 + \omega_2 x_2 + \dots + \omega_n x_n \\ m_2 &= \omega_1 x_1^2 + \omega_2 x_2^2 + \dots + \omega_n x_n^2 \\ &\vdots \\ m_{2n-1} &= \omega_1 x_1^{2n-1} + \omega_2 x_2^{2n-1} + \dots + \omega_n x_n^{2n-1} \end{aligned} \quad (2.3)$$

ここで

$$\begin{aligned} p(x) &= (x - x_1)(x - x_2) \dots (x - x_n) = \sum_{k=0}^n c_k x^k, \\ c_n &= 1 \end{aligned} \quad (2.4)$$

と定義しておく、さて、(2.3) 式から

$$\begin{aligned} m_0 c_0 &= \omega_1 c_0 + \omega_2 c_0 + \dots + \omega_n c_0 \\ m_1 c_1 &= \omega_1 c_1 x_1 + \omega_2 c_1 x_2 + \dots + \omega_n c_1 x_n \\ m_2 c_2 &= \omega_1 c_2 x_1^2 + \omega_2 c_2 x_2^2 + \dots + \omega_n c_2 x_n^2 \\ &\vdots \\ m_n c_n &= \omega_1 c_n x_1^n + \omega_2 c_n x_2^n + \dots + \omega_n c_n x_n^n \end{aligned} \quad (2.5)$$

が成り立つから、(2.5)の両辺を加えると

$$\begin{aligned} m_0 c_0 + m_1 c_1 + m_2 c_2 + \dots + m_n c_n \\ &= \omega_1 (c_0 + c_1 x_1 + \dots + c_n x_1^n) \\ &\quad + \omega_2 (c_0 + c_1 x_2 + \dots + c_n x_2^n) + \dots \\ &\quad + \omega_n (c_0 + c_1 x_n + \dots + c_n x_n^n) \end{aligned} \quad (2.6)$$

となる。 i 番目の括弧の中は

$$\sum_{k=0}^n c_k x_i^k = p(x_i) = 0 \quad (2.7)$$

であるから

$$m_0 c_0 + m_1 c_1 + \dots + m_n c_n = 0 \quad (2.8)$$

が成立する事がわかる。同様の操作を (2.3) 式 2, 3, ..., $(n+1)$ 番目の方程式に対して実行すると

$$m_1 c_0 + m_2 c_1 + \dots + m_{n+1} c_n = 0$$

を得る。以下同じように計算を続けていけば、 $c_n = 1$ と与えることができるので、結局 c_0, \dots, c_{n-1} に関する n 元連立方程式を得られる。

$$\begin{aligned} m_0 c_0 + m_1 c_1 + \dots + m_{n-1} c_{n-1} &= -m_n \\ m_1 c_0 + m_2 c_1 + \dots + m_n c_{n-1} &= -m_{n+1} \\ m_2 c_0 + m_3 c_1 + \dots + m_{n+1} c_{n-1} &= -m_{n+2} \\ &\vdots \\ m_{n-1} c_0 + m_n c_1 + \dots + m_{2n-2} c_{n-1} &= -m_{2n-1} \end{aligned} \quad (2.9)$$

この方程式系の行列式は

$$d = |m_{i+j}| = \left| \int_a^b \omega(x) x^i \cdot x^j dx \right| \quad (2.10)$$

であるが、これは関数 $1, x, x^2, \dots, x^{n-1}$ のグラム行列式である。これらの関数が 1 次独立であることに注意すれば、 $d \neq 0$ であることがわかる。したがって、方程式系 (2.9) を解いて定数 c_0, \dots, c_{n-1} に対する一意解を得る事が出来る。そうしてから (2.7) 式を解けば、分点 x_1, \dots, x_n が求められる。重み関数 $W(x)$ が、積分区間で、正值であるならば、分点 x_1, \dots, x_n は実根であることが証明できるが、一般に複素数である。重み関数 $W(x)$ が、積分区間で、正值であることは、分点を実数であることの十分条件であるから、この条件を満たさない場合でも実数の分点を持つ有用な積分公式が計算できると期待できる。

分点が求められたならば、(2.3) 式から重み $\omega_1, \dots, \omega_n$ を得る事が出来る。この考え方を少し変えれば、1 点あるいは数点の指定した分点を持つガウス型公式を得るのに利用する事が出来る。

例えば、 $f(x) = 1, x, x^2, \dots, x^{2n-2}$ に対して

$$\int_a^b \omega(x) f(x) dx = \omega_1 f(x_1) + \sum_{k=2}^n \omega_k f(x_k) \quad (2.11)$$

が成立するように、 $2n-1$ 個の数値 $\omega_1, \omega_2, \dots, \omega_n, x_2, x_3, \dots, x_n$ を定める事を考える。 x_1 は $x_1 \geq b$ あるいは $x_1 \leq a$ を満足する指定した数であると仮定して、この場合の決定方程式は

$$m_k = \omega_1 x_1^k + \sum_{i=2}^n \omega_i x_i^k, \quad k = 0, 1, \dots, 2n-2 \quad (2.12)$$

である。方程式 (2.3) に x_i を乗じて、番号の一つ多い式から引けば

$$\mu_k = m_{k+1} - x_1 m_k = \sum_{i=2}^n \omega_i (x_i - x_1) x_i^k, \quad k = 0, 1, \dots, 2n-3 \quad (2.13)$$

を得る。ここで

$$p(x) = (x - x_2)(x - x_3) \cdots (x - x_n) = \sum_{k=0}^{n-1} c_k x^k \quad (2.14)$$

と置いて前と同様の計算を実行すると

$$\sum_{k=0}^{n-1} c_k \mu_{k+j} = \sum_{i=2}^n \omega_i (x_i - x_1) p(x) = 0, \quad j = 0, 1, \dots, n-2 \quad (2.15)$$

を得る。この方程式系の行列式 $d = |\mu_{k+j}|$ は次のようになる。

$$\det\left(\int_a^b \omega(x)(x - x_1)x^k \cdot x^j dx\right) = \det\left(\int_a^b \omega(x)(x - x_1)x^k \cdot x^j dx\right) \quad (2.16)$$

これは判定値の重みの関数 $\omega(x)(x - x_1)$ に関するベキ $1, x, x^2, \dots$ のグラム行列式である。こうして方程式系 (2.12) は解くことができ、前と同様の計算をすすめることができる。ガウス積分公式を定めるこのようなやり方は、関係する行列が特異に近くなってしまうために、通常の精度の数値計算の立場からはあまり利用されていない。

さらに、 n が大きい時には、 $\sum c_k x^k$ の形で多項式を数値計算すると条件が悪くなる傾向があり、零点を正確に定めることができなくなる。この難点をのぞくために、Sack and Donovan[5]は、変形したモーメント

$$\int_a^b \omega(x) q_j(x) dx = m_j \quad (2.17)$$

から出発して

$$\sum c_j q_j(x) \quad (2.18)$$

を使う方法を提案している。ここで $\{q_i(x)\}$ は適当な多項式の組で、ふつうは直交多項式が使われる。

上のアルゴリズムでプログラムを作成した場合、与えなければならないのは、モーメント M_j と固定点の座標 x_l である。モーメント M_j は

$$M_j = \int_a^b W(x) x^j dx \quad j = 0, 1, \dots, k \quad (2.19)$$

で与えられるものである。

$2N$ 個のモーメント M_j 、 m 個の固定点の座標 x_l を与えると、 $(m+N)$ 個の分点を持つ $(2N+m)$ 次のガウス型数値積分公式を計算するように作成できる。また、

$$\int_a^b W(x) f(x) dx = \sum_{i=1}^n \omega_i f(x_i) + E f^{(2n)}(\xi) \quad a \leq \xi \leq b \quad (2.20)$$

としたときの、係数 E も簡単に計算できる。

(2.20) の E を計算するには、(2.20) の式に $f(x) = x^{2n}$ を代入する。代入すると

$$M_{2n} = \sum_{i=1}^n \omega_i f(x_i) + (2n)! E \quad (2.21)$$

となるので、 $2n$ 次のモーメント M_{2n} がわかれば簡単に計算できる。

3. 代数方程式の解法

ガウス型数値積分公式を生成する上で問題になるのが、途中に出現する代数方程式の解法である。

重み関数が正であるならば、直交多項式が生成させ、直交多項式の漸化式を利用した計算方法を適用することによって、ある程度精度低下を防ぐことができる。しかし、次数が高くなるにつれて、この方法も十分な精度で計算することができなくなる。

ガウス型数値積分公式の分点や重みは、通常の代数方程式の解法とことなり、精度が要求されることである。倍精度のガウス型数値積分公式を作るならば、最後の桁まで正しい値が要求されることである。

倍精度で計算し、倍精度の精度で解ける計算法は存在しないことが知られている。このことは次のような方程式を考察することによって容易にわかる。

$$(x-1.23)^6 = 0 \quad (3.1)$$

この方程式の解は、明らかに $x=1.23$ であるが、精度が15桁で良いとするならば、 $x=1.231$ も解である。解の計算精度は3桁となる。このように、次数が高くなると代数方程式は、精度よく計算するのは、非常に困難である。

計算精度を最低でも15桁確保するために、高精度計算ルーチン[4]を使った。

この計算で解かなければならない代数方程式は、すべての根が実数と期待できるので、単純なニュートン法を使って解いているが、収束しない場合があるので、収束が保証されている平野の方法[6]も併用している。もし、根が複素数である場合には、計算量が増えて、ガウス型数値積分公式の利点を失うので、このような計算は行っていない。

根は、絶対値が大きい方から求めている。このように計算すると減次のとき、誤差が大きくなるということが知られているので、係数を逆に並べた逆数を根にもつ方程式を使って、その方程式の減次を

行っている。

平野法では、計算の途中で、方程式の一部を取り2項方程式として解き、解の絶対値が最も小さい場合を選び出す演算がある。この計算は、選び出す段階では、あまり精度を必要としないから、この段階では、精度を落とし計算し、選び出して次の段階に進むとき精度を上げることによって、計算を速度上げることができる。このようにすることによって、30次程度の方程式でも約半分の時間で解くことができるようになった。次数が高くなれば、さらにその効果が現れると思われる。

4. プログラムの構成

ガウス型数値積分公式の分点と重み計算するプログラムは、C++言語を利用して作成した。このプログラムは、次のような形で呼び出すことができる。

```
void gauss_int( int n, long_float *moment,
               long_float *xx, long_float *ww,
               int nfix, long_float *xxf,
               long_float *wwf );
```

ここで、

入力データ

n : 非固定点の分点数
moment : モーメント ((2*n+nfix)個)
nfix : 固定分点数
xxf : 固定分点の座標 (nfix個)

出力データ

xx : 分点の座標 (非固定点) (n個)
ww : 分点の重み () (n個)
wwf : 分点の重み (固定点) (nfix個)

入力データとして、ユーザが与える分点の数nfixとその座標xxf、計算で求める分点の数n、0次から(2n+nfix-1)次のモーメントを与える。計算結果として、分点の座標(n個の配列)、分点の重み(n個の配列)、固定点の分点の重み(nfix個の配列)を返す。

この関数を利用して、自動的にC/C++言語用のプログラムを自動的に生成する関数も作成した。これを利用すれば、簡単にガウス型数値積分の関数を簡単に作成できる。

上記の関数を利用するには、モーメントを与えないといけない。解析的に容易に計算できる場合ならば問題ないが、数値的にしか計算できないような問題も多いと考えられる。

このような問題に適用するために、高精度で計算できる二重指数関数型数値公式を準備した。二重指数関数型数値公式は、被積分関数が弱い特異性をもっている場合でも、任意の精度で計算できる公式である。この積分ルーチンを利用すれば、任意の重み関数を持つガウス型積分公式を容易に

計算できることになる。

5. 数値例

多くのガウス型数値積分公式を生成できるが、ここでは、典型的な積分を以下に例を示す。

(1) ガウスラゲールの積分公式

$$\int_0^{\infty} e^{-x} f(x) dx \approx \sum_{i=1}^n \omega_i f(x_i) \quad (5.1)$$

この場合、モーメント

$$M_j = j! \quad (5.2)$$

である。n=20のとき、付録の表1のように精度80桁の数値を簡単に求められる。

この種のガウス型積分公式の計算は、nが大きくなると、重み (Weight Factors) が非常に小さくなり、アンダーフローが起こり計算が困難になる問題がある。N=20程度の次数では、 10^{-28} 程度に数値しか現れないが、50次、100次と次数が上がると、それぞれ 6.05×10^{-78} 、 3.25×10^{-162} となり、途中の計算を考えるとアンダーフローが簡単に起こることがわかる。

(2) 対数関数を重み関数とするガウス型の積分公式 (固定点がない場合)

$$\int_0^1 \log\left(\frac{1}{x}\right) f(x) dx = \sum_{i=1}^n \omega_i f(x_i) + E f^{(2n)}(\xi) \quad (5.3)$$

この場合、モーメント

$$M_j = \frac{1}{(j+1)^2} \quad (5.4)$$

である。n=5のときの数値を付録のTable 2に示す。この数値は、境界要素法で現れる重要な積分[3]を効率的に計算するために必要な数値である。

(1)の例題とことなり、この重み関数から生成される直交多項式、すなわち、このガウス型数値積分の分点の位置を根とする方程式は、あまり研究されていない。このため、この方程式の根を数値的に求めることを非常に困難にしている。

$f(x) = x^{10}$ を(5.3)式に代入することによって定数Eを計算することができる。

(3) コーシーの主値積分のガウス型積分公式

$$\int_{-1}^1 \frac{f(x)}{x} dx = \sum_{i=1}^n \omega_i f(x_i) \quad (5.5)$$

この積分は通常の意味では存在しない積分であるが、コーシーの主値を積分値と考えれば、モーメントは

$$M_j = \begin{cases} j = \text{even} & 0 \\ j = \text{odd} & \frac{2}{j} \end{cases} \quad (5.6)$$

となる。n=8 ときの数値を付録のTable 3に示す。

この公式は、重み関数が一定符号でないだけでなく、特異性を持つ。このため、ガウス型積分公式が存在する保証はない。実際、計算してみると、偶数次のガウス型公式は存在するが、奇数次のガウス型数値積分公式は存在しない。

(4) 対数関数を重み関数とするガウス型の積分公式 (固定点がある場合)

$$\int_0^1 \log\left(\frac{1}{x}\right) f(x) dx = \omega_c f(x_c) + \sum_{i=1}^n \omega_i f(x_i) \quad (5.7)$$

この場合、モーメント

$$M_j = \frac{1}{(j+1)^2} \quad (5.8)$$

である。固定点 $x_c = \frac{3}{10}$ 、n=7のときの数値を付録

のTable 4に示す。固定点を持つ積分としては、ロバット積分やラダー積分などがあるが、いずれの場合でも固定点は積分区間の端点にあり、上の例題のように、積分区間の内部にあるガウス型数値積分は、これまであまり研究されていないものである。

分点および重みの最初の数値がそれぞれ固定点の位置、重みである。例では、分点の位置は区間内にあり、重みはすべて正数であるが、このような計算は多くの場合、積分区間外の分点が出現し、重みが負になる。

5. まとめ

高精度計算プログラムを利用して、一般的なガウス型積分公式を生成するプログラムを作成した。これを使うと、高精度で能率的な積分計算を行うことができる。計算は、およそ10進数で100桁で計算したが、30次程度の方方程式でも十分な精度とはいえない状況が見受けられた。

このプログラムを使うことによって、問題に合った適切なガウス型積分公式が作成することができるので、効率的に、高精度で計算できるものと期待できる。

参考文献

- [1] Abramowitz and Stegun: HANDBOOK OF MATHEMATICAL FUNCTIONS With formulas, Graphics, and Mathematical Tables, Dover, 1966
- [2] P.J.Davis P.Rabinowitz(森 正武訳)：計算機によ

る数値積分法、日本コンピュータ協会、1981

[3] 榎園 正人：マイコンによる境界要素解析、培風館、1982

[4] 平山 弘，" C++言語による高精度計算パッケージの開発"，日本応用数理学会，Vol. 5，No.3，pp. 123-134，1995

[5] Sack, R.A. and Donovan, A.F.: An algorithm for Gaussian quadrature given modified moments, Numer. Math. 18(1972) pp.465-478

[6] 室田一雄，" 平野の変形Newton法の大域的収束性"，情報処理学会論文誌，Vol. 21(1980) pp. 469-474

[7] 森 正武：数値解析と複素関数論、筑摩書房、1975

付録

Table 1. Abscissas and Weight Factors for Gaussian Integration

$$\int_0^{\infty} e^{-x} f(x) dx \approx \sum_{i=1}^n \omega_i f(x_i)$$

n = 20 Abscissas = x_i						
7.0539889691	9887533666	8900458421	5095869360	6298353100	6537693247	574441750e-00002
3.7212681800	1611443794	2413887611	4663667402	8210156184	1234158265	441997834e-00001
9.1658210248	3273564667	7162770741	8318720560	4198042966	2805750011	627793207e-00001
1.7073065310	2834388068	7689667413	0507061879	3994821525	4602096586	773453957e+00000
2.7491992553	0943212964	5030460494	8133842756	1691740576	4408472293	537208636e+00000
4.0489253138	5088692237	4953369133	3321965066	5244703755	6152848232	092824781e+00000
5.6151749708	6161651410	4539885651	8923479177	5687664442	7164507338	893016019e+00000
7.4590174536	7106330976	8860218371	8175953844	7712017393	6316328059	273404811e+00000
9.5943928695	8109677247	3672734282	7983781506	3789421437	9574452792	003962310e+00000
1.2038802546	9643163096	2340929886	5515868227	5538736752	9595903753	637637246e+00001
1.4814293442	6307399785	1267971004	7975674250	7571056443	4748932939	077390734e+00001
1.7948895520	5193760173	6579099261	2509648678	0044291993	8949076178	347089484e+00001
2.1478788240	2850109757	3517036959	4669216209	7556971095	5144878168	409356164e+00001
2.5451702793	1869055035	1867748464	1541838163	5112236459	0133056602	941354213e+00001
2.9932554631	7006120067	1365613516	5823251660	5603505978	5534373873	209595003e+00001
3.5013434240	4790000062	8493590668	8139581765	9287325973	3056371930	358204481e+00001
4.0833057056	7285710620	2956770780	7552671707	6597491990	0008886119	011069218e+00001
4.7619994047	3465021399	4162715285	1121113143	9703425749	2874479289	596122956e+00001
5.5810795750	0638988907	5077344449	7235628385	3112483335	3939332085	467291837e+00001
6.6524416525	6157538186	4031879146	0665979634	9130056970	8145901753	669229260e+00001
Weight Factors = w_i						
1.6874680185	1113862149	2238996894	8079260392	7970557464	5163680349	929542070e-00001
2.9125436200	6068281716	7953238122	2649010672	6819466688	9449708476	233443710e-00001
2.6668610286	7001288549	5208689978	8224113897	4898758070	2294339990	673771918e-00001
1.6600245326	9506840031	4691278159	3693467646	0803226130	1840228940	228660942e-00001
7.4826064668	7923705400	6246396150	1698808812	0313000177	2627162037	971119630e-00002
2.4964417309	2832210728	2273832343	4186540843	8835233421	2763849816	480561465e-00002
6.2025508445	7223684744	7547853946	6705003810	1380546257	4277903655	246607291e-00003
1.1449623864	7690824203	9553569689	5561934993	1461386432	2094887414	453547796e-00003
1.5574177302	7811974779	8095132141	3489534193	5675738905	3970938109	443800792e-00004
1.5401440865	2249156893	8067140480	9120107480	5887090975	4832888872	025599568e-00005
1.0864863665	1798235147	9700044389	2190230009	1503368624	0994436792	943168642e-00006
5.3301209095	5671475092	7802443054	3401628159	9883255670	9528350380	617114528e-00008
1.7579811790	5058200357	7876378404	4478578733	8281260163	2685681948	320183304e-00009
3.7255024025	1232087262	9245853379	4420377466	4804506289	7967308550	705417098e-00011
4.7675292515	7819052449	4880716128	4488525265	5339155873	2511906562	040765904e-00013
3.3728442433	6243841236	5060649913	8969575096	3200227471	8530963489	186842779e-00015
1.1550143395	0039883096	3962471808	3030921175	6693784325	8190073798	415029117e-00017
1.5395221405	8234355346	3833196674	0244446312	9686997276	3889628013	021223726e-00020
5.2864427255	6915782880	2735876827	9932945003	7262751298	0720749476	004613717e-00024
1.6564566124	9902329590	7819085291	0559845009	2072076320	2192396558	672987564e-00028

Table 2. Abscissas and Weight Factors for Gaussian Integration

$$\int_0^1 \log\left(\frac{1}{x}\right) f(x) dx = \sum_{i=1}^n \omega_i f(x_i) + E f^{(2n)}(\xi)$$

n = 5

Abscissas = x_i

0.0291344721 5197205330 3726762115 4416923961 0793130238 2446684650 5773601
0.1739772133 2089762870 1139710828 5651744303 9224068336 5511959418 0932522
0.4117025202 8490204317 4931924645 6586476161 5211948171 3129100855 4415325
0.6773141745 8282038070 1802667998 0061802828 9928763756 3967397051 4995032
0.8947713610 3100828363 8886204455 1724608950 8556524206 4779090429 9594718

Weight Factors = w_i

2.9789347178 2894457272 2578778879 3820241638 4126309288 7522186802 2577679e-01
3.4977622651 3224180375 0718703073 0988562653 9184965038 1263406036 7168586e-01
2.3448829004 4052418886 9068579425 9698649663 7927295106 2288596794 6294803e-01
9.8930459516 6331469761 8071144039 5543862072 2363956020 8977546562 3020884e-02
1.8911552143 1957964895 8268242175 9381598366 5250349648 0280557101 6568429e-02

E =

1.8151588908 2490819283 1808603735 3810612374 5609032510 4612407618 0468701e-13

Table 3. Abscissas and Weight Factors for Gaussian Integration

$$\int_{-1}^1 \frac{f(x)}{x} dx = \sum_{i=1}^n \omega_i f(x_i)$$

n = 8

Abscissas = x_i

-0.9602898564 9753623168 3560868569 4729904282 3523430145 2038271639 7773724
-0.7966664774 1362673959 1553936475 8304368371 7173161596 4832070170 2950392
-0.5255324099 1632898581 7739049189 2463490419 6424312039 2857750857 0992725
-0.1834346424 9564980493 9476142360 1839806667 5781291297 3782317188 4736992
0.1834346424 9564980493 9476142360 1839806667 5781291297 3782317188 4736992
0.5255324099 1632898581 7739049189 2463490419 6424312039 2857750857 0992725
0.7966664774 1362673959 1553936475 8304368371 7173161596 4832070170 2950392
0.9602898564 9753623168 3560868569 4729904282 3523430145 2038271639 7773724

Weight Factors = w_i

-1.0541456374 3895982946 6602497839 8913832293 2594761095 4610649877 5208905e-01
-2.7913944010 2128125624 0192923871 7329389456 2388548999 4656481673 4798169e-01
-5.9693111206 5634770044 7984355589 2120489430 4953560034 3831242134 9524310e-01
-1.9771825999 9314531302 5560852073 5077095013 5850419318 5749874140 1661330e+00
1.9771825999 9314531302 5560852073 5077095013 5850419318 5749874140 1661330e+00
5.9693111206 5634770044 7984355589 2120489430 4953560034 3831242134 9524310e-01
2.7913944010 2128125624 0192923871 7329389456 2388548999 4656481673 4798169e-01
1.0541456374 3895982946 6602497839 8913832293 2594761095 4610649877 5208905e-01

Table 4. Abscissas and Weight Factors for Gaussian Integration

$$\int_0^1 \log\left(\frac{1}{x}\right) f(x) dx = \omega_c f(x_c) + \sum_{i=1}^n \omega_i f(x_i)$$

n = 7 Abscissas = x_i

c	0.3000000000	0000000000	0000000000	0000000000	0000000000	0000000000	00000000
1	0.0000903853	6467977222	3371202489	2232819682	4450282475	0596494859	1047901
2	0.0415692073	7737008675	7468170497	7584522602	3150520124	2678066343	2727469
3	0.1459917131	4346717064	6206816898	8887488831	6522817733	8498231388	8313734
4	0.4827511628	0799830262	8863677874	1391463312	5683014159	7828526103	3537790
5	0.6686676300	4700496657	8720568740	0339859987	5991651980	0325398879	6343584
6	0.8313440243	8100330859	3639326659	0962189942	2059681296	9988568331	1565671
7	0.9474860113	7383926986	8558661119	6241195178	9504487544	4147034192	1391941

Weight Factors = w_i

c	2.0736447667	5058303544	5860984435	4805214527	9946969491	2998834676	5331547e-01
1	6.1845552088	0237927119	6791045151	6429200094	8833063090	1139484402	1035799e-02
2	2.3569043657	2431283866	7274538560	2598150219	0630112109	6832097823	5532628e-01
3	2.5463046819	3659771132	7604779017	3966318764	9741695814	7881744545	6011315e-01
4	1.3756768197	0803041954	6207100873	6043838690	2748183426	5517180986	6966867e-01
5	7.1836022012	3985538135	9230481955	9047942299	8372024971	4299526793	0211031e-02
6	2.6396252914	6959751765	1965879509	4824397277	6435417855	0848140466	0514154e-02
7	4.6691095729	2927779922	5385645155	5632383045	6898856601	4142680149	8154413e-03