Extended Gaussian quadratures for functions with an end-point singularity of logarithmic type

K. Pachucki[a], M. Puchalski[b,a], V. A. Yerokhin[c,*]

[a] *Institute of Theoretical Physics, Warsaw University, Hoża 69, 00-681 Warsaw, Poland*
[b] *Faculty of Chemistry, Adam Mickiewicz University, Umultowska 89b, 61-614 Poznań, Poland*
[c] *Center for Advanced Studies, St. Petersburg State Polytechnical University, Polytekhnicheskaya 29, St. Petersburg 195251, Russia*

Abstract

The extended Gaussian quadrature rules are shown to be an efficient tool for numerical integration of wide class of functions with singularities of logarithmic type. The quadratures are exact for the functions

$$\mathrm{pol1}_{n-1}(x) + \ln x\, \mathrm{pol2}_{n-1}(x)\,,$$

where $\mathrm{pol1}_{n-1}(x)$ and $\mathrm{pol2}_{n-1}(x)$ are two arbitrary polynomials of degree $n-1$ and n is the order of the quadrature formula. We present an implementation of numerical algorithm that calculates the nodes and the weights of the quadrature formulas, provide Fortran code for numerical integration, and test the performance of different kinds of Gaussian quadratures for functions with logarithmic singularities.

Keywords: numerical integration; quadratures;

PROGRAM SUMMARY

Manuscript Title: Extended Gaussian quadratures for functions with an end-point singularity of logarithmic type.
Authors: K. Pachucki, M. Puchalski, and V. A. Yerokhin.
Program Title: GAUSEXT
Journal Reference:
Catalogue identifier:

*Corresponding author.
E-mail address: yerokhin@spbcas.ru

Licensing provisions: None.
Programming language: Mathematica, Fortran.
Computer: PCs or higher performance computers.
Operating system: Linux, Windows, MacOS.
RAM: Kilobytes.
Keywords: Quadratures, numerical integration, end-point singularity.
Classification: 4.11 Quadratures.
Nature of problem: Quadrature formulas for numerical integration, effective for a wide class of functions with end-point singularities of logarithmic type.
Solution method: The method of solution is based on the algorithm developed in Ref. [1] with some modifications.
Running time: milliseconds to minutes.

References:

[1] J. Ma, V. Rokhlin, S. Wandzura, Generalized gaussian quadrature rules for systems of arbitrary functions, Soc. Indust. Appl. Math. J. Numer. Anal. 33 (3) (1996) 971 – 996.

1. Introduction

Classical Gaussian quadrature formulas are widely used for performing numerical integrations in physics and chemistry. They are most efficient for functions that are well approximated by polynomials. When the function to be integrated is very different from polynomials, however, Gaussian quadratures do not perform well. In particular, this is the case when the integrand has an end-point integrable singularity. The standard approach for such cases is to apply a Gaussian quadrature rule with the suitable chosen weight function (Gauss-Chebyshev, Gauss-Jacobi, etc). However, the Gaussian quadratures with singular weight functions do not work well in many practical situations. More precisely, they perform very well only in those rare cases when the singularity can be *exactly* factored out in terms of the weight function. In a typical situation however, the functions to be integrated is quite complicated and its singularity cannot be factored out explicitly.

In the present paper we demonstrate that the extended Gaussian quadrature rules provide an efficient tool for numerical integration of wide class of functions with singularities of logarithmic type. The extended Gaussian quadratures considered in the present work are exact for the functions of the

type

$$\mathrm{pol1}_{n-1}(x) + \ln x\, \mathrm{pol2}_{n-1}(x)\,, \tag{1}$$

where $\mathrm{pol1}_{n-1}(x)$ and $\mathrm{pol2}_{n-1}(x)$ are two arbitrary polynomials of degree $n-1$ and n is the order of quadrature formula.

The extended Gaussian quadrature rules were introduced many years ago by Karlin and Studden [1]. However, these quadratures have not received much attention from the scientific community so far. The main reason was the lack of efficient algorithms for the numerical evaluation of these quadrature rules. This is in contrast to the standard Gaussian quadratures, which can be easily evaluated for different weight functions, (see e.g., *Numerical Recipes* [2]).

An efficient numerical algorithm for the construction of extended Gaussian quadrature rules was presented in Ref. [3]. The authors proved the convergence of the method and showed some examples. However, the problem of calculation of the nodes and the weights of quadrature rules remained, as the method was rather complicated and required a dedicated implementation in each particular case.

In the present work, we improved the original algorithm of Ref. [3] and implemented it in a Mathematica code, which determines the nodes and the weights of the quadrature rules efficiently and with an arbitrary precision. We also provide a Fortran subroutine that returns the pre-stored nodes and weights in the standard double precision arithmetics. With several numerical integration examples in Sec. 3 we demonstrate that the extended Gaussian quadratures perform much better than the standard Gaussian quadratures with logarithmic weight function in many (actually, most) practical situations.

We have used the extended Gaussian quadrature rules for a decade in our calculations of various properties of few-electron systems [4, 5, 6] and found these quadratures to be very efficient integration method of functions with logarithmic singularities.

2. Formulation of the problem and the algorithm

We consider an integral of the form

$$\int_a^b \phi(x)\, dx\,, \tag{2}$$

where $\phi(x)$ is the function to be integrated. This integral is approximated by an n point quadrature rule

$$\sum_{k=1}^{n} w_k \, \phi(x_k) \, , \tag{3}$$

where w_k and x_k are the weights and the nodes of the quadrature formula, respectively. The quadrature rules are usually chosen to be exact for a certain class of functions. In particular, the standard Gauss-Legendre quadratures are exact for the set of $2n$ functions $\{\phi_1(x), \ldots \phi_{2n}(x)\}$, with

$$\begin{aligned} \phi_1(x) &= 1 \, , \\ \phi_2(x) &= x \, , \\ &\vdots \\ \phi_{2n}(x) &= x^{2n-1} \, . \end{aligned}$$

An *extended* n-point Gaussian quadrature rules are, by definition, exact for the general set of $2n$ functions $\{\phi_1(x), \ldots \phi_{2n}(x)\}$, which are not necessarily polynomials. The existence and uniqueness of extended Gaussian quadratures for a wide class of functions $\{\phi_1(x), \ldots \phi_{2n}(x)\}$ were proven in Ref. [1], with the condition that these functions constitute a Chebyshev system on the interval $[a, b]$. This important extension of the classical quadrature rules possesses most of desirable properties of the standard Gaussian integration formulas, such as positivity of the weights, fast convergence and mathematical elegance.

In the present work, we are interested in the extended Gaussian quadratures that are exact for the following set of functions

$$\{1, \, x, \, \ldots x^{n-1}, \, \ln x, \, x \ln x, \, \ldots x^{n-1} \ln x\}. \tag{4}$$

In other words, the n-point quadrature formula should evaluate exactly the following integrals

$$\int_0^1 dx \, [\mathrm{pol1}_{n-1}(x) + \ln x \, \mathrm{pol2}_{n-1}(x)] \, , \tag{5}$$

where $\mathrm{pol1}_{n-1}(x)$ and $\mathrm{pol2}_{n-1}(x)$ are two arbitrary polynomials of the maximal degree $n-1$.

In order to find the weights w_k and nodes x_k of the quadrature formula, one needs to solve a system of $2n$ nonlinear equations for $2n$ variables,

$$\int_0^1 dx\, x^{i-1} = \sum_{k=1}^{n} w_k\, x_k^{i-1}\,, \tag{6}$$

$$\int_0^1 dx\, x^{i-1}\, \ln x = \sum_{k=1}^{n} w_k\, x_k^{i-1}\, \ln x_k\,, \tag{7}$$

where $i = 1, \ldots, n$. The straightforward numerical solution of this system is problematic since the equations are nonlinear and the system is numerically ill-conditioned.

This problem was solved in Ref. [3], where authors formulated a stable numerical algorithm for constructing the extended Gaussian quadrature formulas for a wide class of function. The algorithm consists of two parts. In the first part, an iterative procedure is formulated that allows one to determine the nodes x_i, provided that a good initial approximation is known. It is proven that the iterations converge quadratically. In the second part of the algorithm, a homotopy is constructed that ensures that one gets a good initial approximation for the nodes on each step of numerical procedure. Below we describe the numerical algorithm.

The set of $2\,n$ functions ϕ_i, for which the quadrature formula is exact is

$$\phi_i(x) = x^{i-1}\,, \tag{8}$$

$$\phi_{i+n}(x) = x^{i-1}\, \ln x\,, \tag{9}$$

for $i = 1 \ldots n$. We now construct another set of functions σ_i, η_i $(i = 1 \ldots n)$,

$$\sigma_i(x) = \sum_{j=1}^{2\,n} \alpha_{ij}\, \phi_j(x)\,, \tag{10}$$

$$\eta_i(x) = \sum_{j=1}^{2\,n} \beta_{ij}\, \phi_j(x)\,. \tag{11}$$

The linear coefficients α_{ij} and β_{ij} in the above equations are uniquely determined by the following set of conditions on σ_i and η_i

$$\begin{aligned} \sigma_i(x_k) &= 0\,, \\ \sigma_i'(x_k) &= \delta_{ik}\,, \\ \eta_i(x_k) &= \delta_{ik}\,, \\ \eta_i'(x_k) &= 0\,, \end{aligned} \tag{12}$$

where x_k ($k = 1 \ldots n$) are approximations for the true solutions of Eqs. (6,7), i.e., the nodes of the quadrature formula.

If x_k are the true solutions, then, for any $i = 1 \ldots n$,

$$\begin{aligned} \int_0^1 dx\, \sigma_i(x) &= 0\,, \\ \int_0^1 dx\, \eta_i(x) &= w_i\,. \end{aligned} \tag{13}$$

If x_k are not exactly the solutions but are sufficiently close, then the iteration $x_k \to \tilde{x}_k$

$$\tilde{x}_k = x_k + \frac{\displaystyle\int_0^1 dx\, \sigma_k(x)}{\displaystyle\int_0^1 dx\, \eta_k(x)}\,, \tag{14}$$

converges quadratically to the solutions of Eqs. (6,7) [3].

In order to get a good initial approximation for the nodes x_i, Ref. [3] proposes to construct an one-parameter family of systems (homotopy) $\phi_i^t(x)$ ($i = 1 \ldots 2n$) as follows

$$\phi_i^t(x) = (1-t)\, x^{i-1} + t\, \phi_i(x)\,. \tag{15}$$

Note that for $t = 0$, the system of functions $\{\phi_i^{t=0}\}$ coincides with the basic polynomials (which corresponds to the standard Gauss-Legendre quadratures), whereas for $t = 1$, the system of functions $\{\phi_i^{t=1}\}$ coincides with the system of our problem (8,9). So, the idea is to gradually change t from 0 to 1 and for each t determine the nodes by the iteration procedure formulated above. As a result, one gets the set of nodes for $t = 1$, which is the nodes of the required quadrature formula.

An improved choice of the homotopy for the set of functions (8,9) was proposed in Ref. [4], which was shown to yield better convergence than the original version (15). Let us change the initial set of functions $\{\phi_i^{t=0}\}$ in (15). Instead of $\{1, \ldots, x^{2n-1}\}$, we will use $\{x^{-1/2}, 1, x^{1/2}, x, \ldots, x^{n-3/2}, x^{n-1}\}$. The exact nodes of the quadrature formula for such set of functions are known; they are $x_k = y_k^2$, where y_k are the nodes of the standard Gauss-Legendre quadrature. For the purpose of the present work, we reorder this set and write is as $\{1, \ldots, x^{n-1}; x^{-1/2}, x^{-1/2}\, x, \ldots, x^{-1/2}\, x^{n-1}\}$. We note that the first half of the functions in this set coincides with the first half of system

of our problem (8,9) and only the second half is different. So, we define the homotopy as the the one-parameter family of $2n$ systems of the following form

$$\begin{aligned} \phi_i^t(x) &= x^{i-1}\,, \\ \phi_{i+n}^t(x) &= [(1-t)/\sqrt{x} + t\,\ln(x)]\,x^{i-1}\,, \end{aligned} \tag{16}$$

where $i = 1 \ldots n$. By slowly increasing t from 0 and applying at each step the iteration procedure, one finds the solution at $t = 1$. In the actual calculations we found that this choice of the initial set of function $\{\phi_i^{t=0}\}$ yields such a good approximation for the nodes of the final quadrature formula that the iteration procedure (14) converges without homotopy, i.e., the trivial one-step homotopy is sufficient in this case.

3. Results

The main subject of the present work are the extended Gaussian quadrature rules for the set of functions (8,9), which will be referred to as the extended Gauss-log quadratures. The n-point quadrature formula of this type is exact for the functions

$$\mathrm{pol1}_{n-1}(x) + \ln x\,\mathrm{pol2}_{n-1}(x)\,, \tag{17}$$

where $\mathrm{pol1}_{n-1}(x)$ and $\mathrm{pol2}_{n-1}(x)$ are two arbitrary polynomial of maximal degree $n-1$. In this work we present a Mathematica program `gausext.m`, which calculates the nodes and the weights of the extended Gaussian quadrature rules with an arbitrary accuracy. The code was tested for the maximal maximal order of quadrature $n = 128$ and the maximal accuracy of 64 significant digits, but it should work for higher n and higher accuracy as well. It can be easily adapted for generating extended Gaussian quadratures for different sets of functions $\phi_i(x)$. The Mathematica code is very compact; its listing is presented in Appendix.

The Mathematica code has been used to generate the extended Gauss-log quadrature formulas in octuple-precision arithmetics (about 64 significant digits), which was required in our lithium and hydrogen molecule calculations [4, 5, 6]. For most physical problems, however, the standard Fortran double precision arithmetics is sufficient. So, we provide a Fortran subroutine `gausext.f` with pre-stored extended Gauss-log quadratures in the double

precision arithmetics. The sequence of the pre-stored quadrature orders n is $\{4, 5, 6, 8, 10, 12, 14, 16, 20, 24, 32, 48, 64\}$.

The obtained quadrature rules are verified and tested on three examples. In the first example we evaluate the integral

$$I_1 = \int_0^1 dx \, [\sin x + \ln x \, \cos x] \,, \tag{18}$$

with the numerical results listed in Table 1. This table is produced by the example test file `test1.f`. We observe that for $n > 10$ the quadrature formulas evaluate the integral to machine precision. We might have anticipated this because the function under the integral is very well approximated by the class of functions (17) for which the quadrature formula is exact. So, Table 1 shows mainly that the quadrature rules are evaluated correctly and have the required properties.

In most practical applications, the function to be integrated is rather complicated. It is also often the case that little is known about the function, except for the type of the singularity and its location. We would like to demonstrate now that the extended Gauss-log quadratures perform very well for wide class of different functions that have an end-point singularity of the logarithmic type. In order to test the performance, we will compare the n-point extended Gauss-log quadratures with (i) the standard n-point Gauss-Legendre quadratures that are exact for the functions

$$\mathrm{pol}_{2n-1}(x) \,, \tag{19}$$

and (ii) the standard n-point Gauss-log quadratures that are exact for the functions

$$\ln x \, \mathrm{pol}_{2n-1}(x) \,. \tag{20}$$

We now compare the performance of the three different quadratures for the following integral

$$I_2 = \int_0^1 dx \, \sqrt{1 - x^3} \, \ln \left[1 - \sqrt{1 - \sqrt{x}} \right] \,. \tag{21}$$

The integrand in this example is not very physical, but it represents a function with logarithmic singularity at $x = 0$ which is clearly outside of the class of functions for which the quadratures are obtained. Radicals are not well approximated by polynomials, so we can expect slow convergence for all three

quadrature rules. The results of the calculation are presented in Table 2. The table is produced by the example test file `test2.f`. In the Fortran code, the Gauss-Legendre qudratures and the standard Gauss-log quadratures are calculated by the standard codes from *Numerical Recipes* [2]. We note that the standard Gauss-Legendre quadratures do not perform well in this case, which is not surprising because it does not account for the logarithmic end-point singularity of the integrand. More importantly, we observe that the standard Gauss-log quadratures do not provide much of advantage as compared to the Gauss-Legendre quadratures. The rate of convergence is the same for both quadratures, the Gauss-log ones providing just one additional correct digit. This is a typical situation, which we frequently observed practical calculations. On the contrary, the extended Gauss-log quadratures demonstrate a higher rate of convergence as compared to the other two quadratures. For $n = 64$, their result is 4 digits more accurate than the Gauss-log result and 5 digit more accurate than the Gauss-Legendre result.

We now turn to an even more difficult test case of a 3-dimensional integral of the form

$$I_3 = \int_0^1 dx \int_0^1 dy \int_0^1 dz \, \frac{1}{(1-xyz)^2} = \frac{\pi^2}{6} \,. \tag{22}$$

The integrand in the above expression has an integrable singularity when all three integration variables approach unity, which can be shown to be of a logarithmic type. Expressions of this kind are commonly encountered in numerical evaluations of Feynman parameters integrals in multi-loops QED diagrams [7]. In the cases when these expression can be evaluated analytically, partial results are expressed in terms of logarithmic, dilogarithmic, and polylogarithmic functions, which usually possess logarithmic end-point singularities.

The results of the calculation are presented in Table 3. The table is produced by the example test file `test3.f`. The 3-dimensional integral was evaluated by applying the quadrature formula to each integration variable. In order to bring the singularity to the point $x = y = z = 0$, we made the substitution of variables $x \to 1 - x$, $y \to 1 - y$, $z \to 1 - z$. We observe that the convergence of quadrature formulas in this case is even slower than in the previous example. The 64-point Gauss-Legendre quadrature gives the relative accuracy of about 10^{-4} only. The Gauss-log quadrature rules yield the accuracy even worse than that of the Gauss-Legendre formula. On the contrary, the extended Gauss-log quadratures demonstrate a higher rate of convergence and a higher accuracy than the Gauss-Legendre formula.

Table 1: Numerical integration of Eq. (18) by the extended Gauss-log quadratures. $S(n)$ is the result of the n-point quadrature formula. $\delta S(n)$ is the increment with respect to the previous line.

n	$S(n)$	$\delta S(n)$
4	−0.486 394 220 959 086	
5	−0.486 385 279 839 337	0.000 008 941 119 750
6	−0.486 385 374 818 375	−0.000 000 094 979 038
8	−0.486 385 376 235 414	−0.000 000 001 417 039
10	−0.486 385 376 235 323	0.000 000 000 000 092
12	−0.486 385 376 235 323	0.000 000 000 000 000
14	−0.486 385 376 235 323	0.000 000 000 000 000
16	−0.486 385 376 235 323	0.000 000 000 000 000
20	−0.486 385 376 235 323	0.000 000 000 000 000
24	−0.486 385 376 235 323	0.000 000 000 000 000
32	−0.486 385 376 235 323	0.000 000 000 000 000
48	−0.486 385 376 235 323	0.000 000 000 000 000
64	−0.486 385 376 235 322	0.000 000 000 000 000
exact	−0.486 385 376 235 323	

4. Conclusion

In the paper, we have studied the extended Gaussian quadrature rules for functions with an end-point singularity of logarithmical type. The quadratures are exact for the functions $\mathrm{pol1}_{n-1}(x) + \ln x\, \mathrm{pol2}_{n-1}(x)$. They were shown to be very efficient in numerical integrations of wide class of functions with approximate logarithmic singularity. We present a Mathematica code that implements an efficient numerical algorithm of determination of the weights and the nodes of the extended Gaussian quadrature formulas and a Fortran subroutine that returns the pre-stored values of the weights and the nodes. Several numerical examples demonstrate how to use the quadrature rules in practical calculations in Fortran.

Acknowledgements

The authors acknowledge support from NCN Grants 2012/04/A/ST2/00105.

Table 2: Numerical integration of Eq. (21) by different quadratures.

	Extended Gauss-log		Gauss-log		Gauss-Legendre	
n	$S(n)$	$\delta S(n)$	$S(n)$	$\delta S(n)$	$S(n)$	$\delta S(n)$
4	−0.861 196 754 27		−0.863 562 1		−0.845 247	
5	−0.861 255 319 18	−0.000 058 564 92	−0.862 788 4	0.000 773 6	−0.850 626	−0.005 379
6	−0.861 275 807 84	−0.000 020 488 66	−0.862 350 4	0.000 437 9	−0.853 688	−0.003 061
8	−0.861 287 016 19	−0.000 011 208 35	−0.861 898 4	0.000 452 0	−0.856 867	−0.003 178
10	−0.861 289 607 49	−0.000 002 591 30	−0.861 682 4	0.000 216 0	−0.858 400	−0.001 532
12	−0.861 290 418 69	−0.000 000 811 20	−0.861 563 0	0.000 119 3	−0.859 254	−0.000 854
14	−0.861 290 727 45	−0.000 000 308 76	−0.861 490 4	0.000 072 6	−0.859 779	−0.000 524
16	−0.861 290 862 31	−0.000 000 134 86	−0.861 443 1	0.000 047 3	−0.860 124	−0.000 345
20	−0.861 290 961 77	−0.000 000 099 46	−0.861 387 4	0.000 055 7	−0.860 536	−0.000 411
24	−0.861 290 992 11	−0.000 000 030 33	−0.861 357 2	0.000 030 1	−0.860 763	−0.000 226
32	−0.861 291 008 27	−0.000 000 016 16	−0.861 327 4	0.000 029 7	−0.860 991	−0.000 228
48	−0.861 291 012 84	−0.000 000 004 57	−0.861 306 6	0.000 020 8	−0.861 156	−0.000 165
64	−0.861 291 013 39	−0.000 000 000 55	−0.861 299 5	0.000 007 0	−0.861 215	−0.000 058

Table 3: Numerical integration of Eq. (22) by the different quadratures.

	Extended Gauss-log		Gauss-log		Gauss-Legendre	
n	$S(n)$	$\delta S(n)$	$S(n)$	$\delta S(n)$	$S(n)$	$\delta S(n)$
4	1.656 330 61		1.547 20		1.601 08	
5	1.650 602 92	−0.005 727 70	1.577 35	0.030 15	1.615 38	0.014 29
6	1.648 049 71	−0.002 553 21	1.595 56	0.018 20	1.623 69	0.008 31
8	1.646 094 31	−0.001 955 39	1.615 40	0.019 83	1.632 46	0.008 76
10	1.645 457 28	−0.000 637 04	1.625 35	0.009 95	1.636 74	0.004 28
12	1.645 202 75	−0.000 254 52	1.631 02	0.005 67	1.639 14	0.002 40
14	1.645 085 63	−0.000 117 13	1.634 55	0.003 53	1.640 63	0.001 48
16	1.645 025 84	−0.000 059 79	1.636 89	0.002 34	1.641 61	0.000 97
20	1.644 973 37	−0.000 052 47	1.639 71	0.002 81	1.642 77	0.001 16
24	1.644 953 58	−0.000 019 79	1.641 27	0.001 56	1.643 42	0.000 64
32	1.644 940 46	−0.000 013 12	1.642 85	0.001 57	1.644 07	0.000 65
48	1.644 935 37	−0.000 005 09	1.644 00	0.001 14	1.644 54	0.000 47
64	1.644 934 49	−0.000 000 89	1.644 40	0.000 40	1.644 71	0.000 16
exact	1.644 934 07					

[1] S. Karlin, W. Studden, Tschebucheff Systems with Applications in Analysis and Statistics, John Whiley (Interscience), New York, 1966.

[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes: The Art of Scientific Computing, 3rd Edition, Cambridge University Press, NY, 2007.

[3] J. Ma, V. Rokhlin, S. Wandzura, Generalized gaussian quadrature rules for systems of arbitrary functions, Soc. Indust. Appl. Math. J. Numer. Anal. 33 (3) (1996) 971 – 996.

[4] M. Puchalski, K. Pachucki, Ground-state wave function and energy of the lithium atom, Phys. Rev. A 73 (2) (2006) 022503.

[5] M. Puchalski, K. Pachucki, Ground state hyperfine splitting in ^{6,7}Li atoms and the nuclear structure, Phys. Rev. Lett. 111 (2013) 243001.

[6] K. Pachucki, V. A. Yerokhin, Application of the fully correlated basis of exponential functions for molecular hydrogen, Phys. Rev. A 87 (2013) 062508.

[7] V.A. Smirnov, Feynman Integral Calculus, Springer, Heidelberg, 2010.

Appendix A. Mathematica code

Here we present the Mathematica code for the calculation of the nodes and the weights of the extended Gauss-log quadrature formula.

```
gausext[n_,outprecision_] :=
Module[{f,g,precision,gauss,switch,m,v,dx,it,ff,xxi,wwi,test},
  precision  = IntegerPart[(2+ n/20) Max[outprecision,n/2]];
  f[x_,j_] = If[j<=n, x^(j-1)      , -Log[x] x^(j-n-1)];
  g[x_,j_] = If[j<=n, (j-1) x^(j-2), -((j-n-1) Log[x]+1) x^(j-n-2)];
  xi =  N[Solve[LegendreP[n, x] == 0, x], precision];
  xi = x/.xi;
  xi = N[(xi+1)/2,precision];
  xi = Sort[xi];
  xi = xi^2;
  xi = SetPrecision[xi,precision];
  switch = True;
  it=0;
  While[switch,(
   m = Table[Join[Table[f[xi[[i]],j],{i,1,n}],
                                  Table[g[xi[[i]],j],{i,1,n}]],{j,1,2n}];
      v = Join[Table[1/i,{i,1,n}],Table[1/i^2,{i,1,n}]];
      v =  LinearSolve[m,v];
      dx = Table[v[[n+i]]/v[[i]],{i,1,n}];
      dx = SetPrecision[dx,precision];
      xi = xi+dx;
      it =it+1;
      Print["iteration = ",it];
      If[Max[Abs[dx]]<10^(-outprecision-16), switch=False, switch=True])];
  wi = Table[N[v[[i]],outprecision],{i,1,n}];
  xi = N[xi,outprecision];
  Print[" "];
  Do[Print["      x(",i,")=",ScientificForm[xi[[i]],
                    NumberFormat->(SequenceForm[#1,"e",#3]&)]],{i,1,n}];
  Print[" "];
  Do[Print["      w(",i,")=",ScientificForm[wi[[i]],
                    NumberFormat->(SequenceForm[#1,"e",#3]&)]],{i,1,n}];
  Print[" "];
  Print["testing ..."];
```

```
Print["integration of Sum[x^i ((i+1) + (i+1)^2 Log[x]),{i,0,n-1}]"];
ff[x_] = Sum[x^i ((i+1) + (i+1)^2 Log[x]),{i,0,n-1}];
xxi = SetPrecision[xi,outprecision+64];
wwi = SetPrecision[wi,outprecision+64];
test = N[Sum[wwi[[i]] ff[xxi[[i]]],{i,1,n}],3];
Print["gausext integration = ",test];
Print["exact value of the integral = ",0]];
```