Computing the zeros and weights of Gauss-Laguerre and Gauss-Hermite quadratures: 2 Matlab files

Thomas Vallée

Computing the zeros and weights of Gauss-Laguerre and Gauss-Hermite **quadratures: 2 Matlab files**

Thomas Vallée*

2018/11

(*) LEMNA - Université de Nantes

Computing the zeros and weights of Gauss-Laguerre and Gauss-Hermite quadratures: 2 Matlab files

Thomas Vallée

LEMNA, Université de Nantes

IAE de Nantes - Institut d'Économie et de Management

Chemin de la Censive du Tertre, BP 52231

44322 Nantes Cedex 3, France.

thomas.vallee@univ-nantes.fr

Abstract

This article provides two simple Matlab files to compute the zeros and weights of the Gauss-Laguerre and Gauss-Hermite quadratures.

1 Introduction

Economic problems usually deal with integrals of the form $\int_{-\infty}^{\infty} e^{-x^2} f(x)dx$ or $\int_0^{\infty} e^{-x} f(x)dx$. The first one arises naturally in problem evolving normal random variables. The second one is simply a typical exponentially discounted sum. To evaluate (numerically) such integrals, one has to use appropriated methods, that is the Gauss-Hermite and the Gauss-Laguerre quadratures respectively.

These both procedures require to redefine these integrals as some special polynomials that are evaluated as the sum depending on some weights w_i and zeros x_i given the number of points to be used n. Although tables of these values exist [4], we provide two simple Matlab programs that calculate, given n, the optimal weights and zeros. Other methods to compute such weights and zeros [2] exist and are based on some recursive procedure. The method we used is based on the fact that the associated polynomial can be rewritten as some finite series.

2 Gauss-Laguerre quadrature

Evaluation of integral of the form $\int_0^{\infty} e^{-x} f(x)dx$ using Gauss-Laguerre quadrature involves to calculate $\sum_{i=1}^{n} w_i f(x_i)$, where x_i are zeros of the associated polynomial $L_n(x) = e^x \frac{d^n}{dx^n}(e^{-x}x^n)$ and where w_i are some weights defined by $w_i = \frac{(n!)^2}{x_i(L_n'(x_i))^2}$.

After some algebras, one can find that the polynomial $L_n(x)$ can be redefined as follows :

$$L_n(x) = e^x \frac{d^n}{dx^n}(e^{-x}x^n) \tag{2.1}$$
$$= \sum_{i=1}^{n+1} \left[(-1)^{n+1-i} [b_i \frac{n!}{(n+1-i)!} x^{n+1-i}] \right]$$

with $0! \equiv 1$ and where the b_i are some binomial coefficients corresponding to n and i, that is $b_i \equiv \frac{n!}{i!(n-i)!}$.

The program **gaussla.m**, described in appendix, is a Matlab function that gives as output the values of the zeros x_i and weights w_i given n as the unique input. It used the standard **roots.m** file that calculate the roots of a polynomial using eigenvalue method. It also used the **factorial** function that calculates the factorial value of a given number. For example with $n = 4$, **[x,w]=gaussla(4)** returns :

```
w = 9.39507091230114   x = 0.00053929470556
    4.53662029692113       0.03888790851501
    1.74576110115835       0.35741869243780
    0.32254768961939       0.60315410434163
```

3 Gauss-Hermite quadrature

Evaluation of integral of the form $\int_{-\infty}^{\infty} e^{-x^2} f(x)dx$ using Gauss-Hermite quadrature involves to calculate $\sum_{i=1}^{n} w_i f(x_i)$, where x_i are zeros of the associated polynomial $H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}(e^{-x^2})$ and where w_i are some weights defined by $w_i = \frac{2^{n+1}(n!)\sqrt{\pi}}{(H_n'(x_i))^2}$.

Again, after some algebras it turns out that the polynomial $H_n(x)$ can be rewritten by :

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}(e^{-x^2}) \tag{3.2}$$
$$= (-1)^n \sum_{i=1}^{\tilde{n}} b_i x^{n-2(i-1)}$$

where $\tilde{n} \equiv 1 + \frac{n}{2}$ if n is an even number, and $\tilde{n} \equiv \frac{n+1}{2}$ if n is an odd number. The b_i are more complicated to find and require some specific procedures. Indeed, the coefficients b_i are set as follows :

n						
0	1					
1	-2					
2	4	-2				
3	-8	12				
4	16	-48	12			
5	-32	160	-120			
6	64	-480	720	-120		
7	-128	1344	-3360	-1680		
8	256	-3584	13440	-13440	1680	
9	-512	9216	-48384	80640	-30240	
10	1024	-23040	161280	-403200	302400	-30240
$\vdots$						

Hence, with $n = 1$ we have $\tilde{n} = 1$ and $H_1(x) = -b_1 x = 2x$. With $n = 2$, we have $\tilde{n} = 2$ and $H_2(x) = (-1)^2(b_1 x^2 + b_2) = 4x^2 - 2$.

At a first glance, an explicit computational relationships between each raw and column of the above values does not seem to exist. By reversing the order of the columns at each raw, we get :

n						
0	1					
1	-2					
2	-2	4				
3	12	-8				
4	12	-48	16			
5	-120	160	-32			
6	-120	720	-480	64		
7	1680	-3360	1344	-128		
8	1680	-13440	13440	-3584	256	
9	-30240	80640	-48384	9216	-512	
10	-30240	302400	-403200	161280	-23040	1024
$\vdots$						

The way to compute the different b_i^n given n is still not straightforward but there do exist now some relations.

Last coefficient of each raw: $b(n, l_n)$

$n > 0$ and even

The last coefficient of each raw $b(n, l_n)$, where l_n is the $\tilde{n}$ value corresponding to n, is given by $b(n, l_n) \equiv 4^{l_n - 1}$. For example, with $n = 4$, we have $l_n = \tilde{n} = 3$ and $b(4, 3) = 4^2 = 16$.

$n > 0$ and odd

We have $b(n, l_n) \equiv (-2)4^{l_n - 1}$. For example, with $n = 7$, we have $l_n = \tilde{n} = 4$ and $b(7, 4) = (-2)4^3 = -128$.

First coefficient of each raw: $b(n,1)$

$n > 2$ and even

Given the initial value $b(2,1) = -2$, we have the following rule that can be iterated forward : $b(n,1) = (-1)^n(2 + 4(l_n - 2))|b(n-2,1)|$.

For example, with $n = 8$, and so $l_8 = 5$, knowing that $b(6,1) = -120$ we have $b(8,1) = (-1)^8(2 + 4(5-2))|b(6,1)| = 14 * 120 = 1680$.

$n > 1$ and odd

Given the initial value $b(1,1) = -2$, we have the following rule that can be iterated forward : $b(n,1) = (-1)^n(2 + 4(l_n - 1))|b(n-2,1)|$. For example, with $n = 3$ we have $l_3 = 2$ and $b(3,1) = (-1)^3(2 + 4(2-1))|b(1,1)| = -6(-2) = 12$. Then with $n = 5$ we get $b(5,1) = (-1)^5(2 + 4(3-1))|b(3,1)| = -10(12) = -120$, and so on.

Intermediate column values

We use a forward recursive to compute each intermediate column values. Notice that the number of columns is given by $\tilde{n}$. Hence, the first new column (second column) appears at $\tilde{n} = 2$, that is $n = 2$ or $n = 3$.

n even

The rule to calculate the $b(n,2)$, with $\tilde{n} \geq 3$ is as follows. First one has to compute the *division value*:

$$d_2 = |b(2,2)/b(2,1)| = |4/-2| = 2$$

The successive values for the column 2 are given by the rule :

$$b(n,2) = (-1)b(n,1)(l_n d_2 - 2)$$

For example, with $n = 4$ we have $l_n = 3$ and $b(4,2) = (-1)b(4,1)(3*2-2) = -48$. Then, with $n = 6$ and $l_n = 4$, we have $b(6,2) = (-1)b(6,1)(4*2-2) = 720$. And so on. The same procedure has to be repeated at each new column. That is, calculate first

$$d_{n'} = |b(n,l_n)/b(n,l_n - 1)|$$

with $n^{'} = 3$ and with $n = 2n^{'} - 2$. Then, for highest value of n, use the forward recursive rule :

$$b(n,n^{'}) = (-1)b(n,n^{'} - 1)((l_n - 1)d_{n'} - d_{n'})$$

For example, value of the fourth column is calculated as follows. First we set $d_4 = |b(6,4)/b(6,3)| = 2/15$ and then calculate $b(8,4) = (-1)b(8,3)(3*(2/15) - 2/15) = -1 * (13440) * (8/15) = -7168$. For the fifth column we calculate $d_6 = 1/14$ and we can obtain $b(10,6) = (-1)b(10,5) * (3(1/14) - 1/14) = -1 * (161280) * (2/14) = -23040$.

n odd

The procedure is the same: calculate first the *division value* of the new column $d_{n'} = |b(n,l_n)/b(n,l_n-1)|$ (with $n = 2n^{'} - 1$), then for highest value of n use the forward recursive rule : $b(n,n^{'}) = (-1)b(n,n^{'}-1)((l_n-1)d_{n'} - d_{n'})$.

For example, values of the third column (i.e. $n^{'} = 3$ and $n = 5$) are given by:

$$\begin{aligned} d_3 &= |b(5,3)/b(5,2)| = 0.2 \\ b(7,3) &= (-1)b(7,2)(3*(0.2)-0.2)) = -1*(-3360)*0.4 = 1344. \\ b(9,3) &= (-1)b(9,2)(4*(0.2)-0.2)) = -1*(80640)*0.6 = -48384. \end{aligned}$$

and so on...

The program

The program **gaussshe.m**, described in appendix, is also a Matlab function that gives as output the values of the zeros x_i and weights w_i given n as the unique input. With $n = 4$, **[x,w]=gausshe(4)** returns :

$$\begin{array}{rr} w = 1.65068012388578 & x = 0.08131283544724 \\ -1.65068012388578 & 0.08131283544724 \\ 0.52464762327529 & 0.80491409000551 \\ -0.52464762327529 & 0.80491409000551 \end{array}$$

4 Evaluation in Matlab

The evaluation of one specific function $f(x)$ may now easily be done by creating appropriate Matlab function that calls as subroutine the **gaussla.m** or **gausshe.m** files. Hence, for example a typical matlab function, let call this file **numintla.m**, that evaluates an integral using Gauss-Laguerre quadrature will be defined as follows :

```
function [sol]=numintla(fun,nb)
[xx,w]=gaussla(nb); % Call the gaussla.m file
sx=size(xx,1);
sol=0;
for i=1:sx,
    x=xx(i);
    fx=eval(fun);
    sol=sol+w(i)*fx;
end;
```

sol is the value of the evaluation, **n** the number of points used and **fun** the function $f(x)$ to be evaluated (**fun** must be a string). For example, to evaluate $\int_0^{\infty} e^{-x} sin(x)dx$ using 6 points ($n = 6$), just call: **sol=numintla(śin(x);6)**. The returned value is **0.50004947479768** while the exact value is 0.5.

References

[1] Judd, K., (1998), *Numerical Methods in Economics*, The MIT Press.

[2] Press, H. S. Teukolsky, W. Vetterling and B. Flannery, (1993), *Numerical Recipes in C: The Art of Scientific Computing*, 2$^{\text{nd}}$ edition, Cambridge University Press.

[3] Scheod, F., (1988), *Numerical Analysis*, 2$^{\text{nd}}$ edition, Schaum's Outline Series, McGraww-Hill.

[4] Stroud, A. and D. Secrest, (1966), *Gaussian Quadrature Formulas*, Prentice Hall.

Appendix

gaussla.m

```
    function [xx,ww]=gaussla(n)
%Calculation of the zeros and weights of Gauss-Laguerre quadrature
% Thomas Vallee March 2009
% thomas.vallee@univ-nantes.fr
    format long;
    b(1)=1;
    for i=1:n, % Calculation of the binomial coefficients
       b(i+1)=(factorial(n))/(factorial(i)*(factorial(n+1-i)));
    end;
   for i=1:n+1, % The polynomial coefficients
       poly(i)=((-1)^(n+1-i))*b(i)*(factorial(n)/(factorial(n+1-i)));
   end;
   xx=roots(poly);% The polynomial roots
   for i=1:n, % Coefficients of the first derivative of the polynomial
      polycd(i)=poly(i)*(n+1-i);
    end;
   for i=1:n, % Evaluation
       x=xx(i);
       solde=0;
       for k=1:n,
           solde=solde+polycd(k)*(x^(n-k));
       end;
       ww(i,1)=(factorial(n)^2)/(xx(i)*(solde^2));
    end;
```

gausshe.m

```
function [xx,ww]=gausshe(n)
%Calculation of the zeros and weights of Gauss-Hermite quadrature
% Thomas Vallee March 2009
% thomas.vallee@univ-nantes.fr
format long; n=nk;
if rem(n,2)==0, % FIRST CASE: n even
    nn=(n/2)+1; % Calculation of the adjusted n
    b=zeros(nn-1,nn-1);
    % Calculation of the first raw values : b(n,1)
    b(1,1)=-2; % starting value
    for i=2:nn-1, %
       b(i,1)=((-1)^(i))*(2+4*(i-1))*abs(b(i-1,1));
    end;
    % Calculation of the last raw values : b(n,n)
    b(1,2)=4; % starting value
    for i=2:nn-1,
       b(i,i+1)=4*b(i-1,i);
    end;
    % calculation of intermediate column values
    for j=2:nn-1,
        valmm=abs(b(j-1,j)/b(j-1,j-1));
        valm=valmm+valmm;
        for i=j:nn-1,
            b(i,j)=(-1)*b(i,j-1)*valm;
        valm=valm+valmm;
       end;
    end;
% Calculation of the zeros and weights
    kk=1;
    for i=1:nn,
       poly=b(nn-1,nn-(i-1));% inversion of the polynomial coefficients
       polyc(kk)=poly;
        polyc(kk+1)=0;
        kk=kk+2;
    end;
    ssp=size(polyc,2);
    polycc=polyc(1:ssp-1)*((-1)^n);
    xx=roots(polycc); % calculation of the zeros
    for i=1:n, % Coefficients of the first derivative of the polynomial
      polyd(i)=polycc(i)*(n+1-i);
    end;
    for i=1:n, % calculation of the weights
        x=xx(i);
        solde=0;
        for k=1:n,
           solde=solde+polyd(k)*(x^(n-k));
        end;
        ww(i,1)=((2^(n+1))*factorial(n)*(pi^(.5)))/(solde^2);
    end;
end;
```

```
if rem(n,2)==1, % SECOND CASE: n odd
    nn=(n+1)/2; % Calculation of the adjusted n
    b=zeros(nn,nn);
    % Calculation of the first raw values : b(n,1)
    b(1,1)=-2; % starting values
    for i=2:nn,
       b(i,1)=((-1)^(i))*(2+4*(i-1))*abs(b(i-1,1));
    end;
   % Calculation of the last raw values : b(n,n)
    b(2,2)=-8; % starting values
    for i=3:nn,
        b(i,i)=4*b(i-1,i-1);
    end;
    % calculation of intermediate column values
    for j=2:nn-1,
        valmm=abs(b(j,j)/b(j,j-1));
        valm=valmm+valmm;
        for i=j+1:nn,
            b(i,j)=(-1)*b(i,j-1)*valm;
            valm=valm+valmm;
        end;
    end;
% calculation of the zeros and weights
    kk=1;
    for i=1:nn,
        poly=b(nn,nn-(i-1));% inversion of the polynomial coefficients
        polyc(kk)=poly;
        polyc(kk+1)=0;
        kk=kk+2;
    end;
    polycc=polyc*((-1)^n);
    xx=roots(polycc); % calculation of the zeros
    for i=1:n, % Coefficients of the first derivative of the polynomial
      polyd(i)=polycc(i)*(n+1-i);
    end;
    for i=1:n, % calculation of the weights
        x=xx(i);
        solde=0;
        for k=1:n,
           solde=solde+polyd(k)*(x^(n-k));
        end;
        ww(i,1)=((2^(n+1))*factorial(n)*(pi^(.5)))/(solde^2);
    end;
end;
```