Applications of Guassian Quadrature

Tao Luo, Wenchao Zhang
Jun 24 2013

Abstract

We finished three tasks of project 3, which discuss high dimensional integral by high dimensional Guassian quadrature. By the way, we apply these methods to our tasks and modify then by matlab.

1. Introduction

Roots of the Legendre polynomials are precisely the nodes used in Gaussian quadrature. Finding such roots in an efficient manner can greatly increase the speed of numerical integration. In this project, we explore the Glaser-Liu-Rokhlin algorithm [1] which yields the roots of Legendre polynomials in linear time. Previous methods include the Golub-Welsch method, which solves for the eigenvalues of a tridiagonal matrix. Known algorithms can find these eigenvalues in quadratic time, and so for a large number of nodes, this method becomes cumbersome. For a small number of nodes, the Golub-Welsch method is preferable, since quadratic algorithms tend to be tractable for smaller inputs. However, for a large number of nodes, a linear time algorithm becomes an invaluable tool!

Several specific methods for numerical evaluation of integrals over higher dimensional regions have been proposed. James Clerk-Maxwell [2] proposed the formulas for the rectangle and the rectangular parallelopipedon in 1877. Appell, Burnside, Ionescue, and Mineur have developed special formulas for planar regions. Tyler [3] recently gave formulas for rectangles, parabolic regions, cubes, and for the hypercubes. Others have developed formulas primarily for rectangular regions based on the formulas for the line.

A Basic Theorem for Numerical Integration. In this section we state a theorem which greatly extends the usefulness of particular integration formulas when they are available. We limit the discussion to integration formulas of the form

$$(1.1)\int_R \omega(x) f(x)dx \approx \sum a_j f(x_j)$$

where R is an n-dimensional region in Euclidean n-space, E_n, x is a point vector in E_n, a_1, …, a_m , am are numbers (real or complex) and evaluation points x_1, …, x_m are in the domain of f. In particular, R will be taken as the closure of a bounded open set in E_n and the weight function w(x) and f are continuous on R. That these conditions may be relaxed will not concern us. The dx indicates a volume element in E_n.

2. Guassian Quadrature

In numerical analysis, a quadrature rule is an approximation of the definite integral of a function, usually stated as a weighted sum of function values at specified points within the domain of integration. (See numerical integration for more on quadrature rules.) An n-point Gaussian

quadrature rule, named after Carl Friedrich Gauss, is a quadrature rule constructed to yield an exact result for polynomials of degree **2n−1** or less by a suitable choice of the points x_i and weights w_i for i = 1,...,n. The domain of integration for such a rule is conventionally taken as [−1,1], so the rule is stated as

$$(2.1)\ \int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} \omega_i f(x_i)$$

Let $f(x) = W(x)g(x)$, then we have following equation:

$$(2.2)\ \int_{-1}^{1} f(x)dx = \int_{-1}^{1} W(x)g(x)dx \approx \sum_{i=1}^{n} \omega'_i\, g(x_i)$$

For the simplest integration problem stated above, i.e. with W(x)=1, the associated polynomials are Legendre polynomials, $P_n(x)$, and the method is usually known as Gauss–Legendre quadrature. With the n^{th} polynomial normalized to give $P_n(1)=1$, the i^{th} Gauss node, x_i, is the i^{th} root of P_n; its weight is given by

$$(2.3)\ \omega_i = \frac{2}{\left(1-x_i^2\right)\left[P_n'(x_i)\right]^2}$$

3. Numerical Evaluation of Multiple Integrals

Continuing from (1.1.1), Let S be another region in E_n such that there exists a transformation J with continuous nonvanishing Jacobian which maps R onto S. We write y=Jx for x∈R. Then if g(y) is defined and continuous on S we may write

$$(2.4)\ \int_S \omega(y)g(y)dy = \int_R \omega(y)g(y)\left|J\right|dx$$

where in the integrand on the right we have y = Jx and |J| is the absolute value of the Jacobian which may also be indicated as ∂y/∂x.

Let R be the Cartesian product of two regions R1 and R2 in lower dimensional Euclidean spaces. Let us designate y∈R1, z∈R2 so that every x∈R can be written (y, z). Suppose there are numerical integration formulas for weight functions $w_1(y)$ and $w_2(z)$, defined on R1 and R2 respectively, so that the error functions may be written

$$(2.5)\quad \begin{aligned} E(R_1, f_1) &= \sum a_i f_1(y_i) - \int_{R_1} \omega_1(y) f_1(y)dy \\ E(R_2, f_2) &= \sum b_j f_2(z_j) - \int_{R_2} \omega_1(z) f_2(z)dz \end{aligned}$$

Then we ask for a numerical integration formula over $R = R_1 \times R_2$ with weight function $w_1(y)w_2(z)$. We have following theorem, by using (1.2.5),

Theorem 1. Let f(x)=f(y,z) be a function defined and continuous on $R = R_1 \times R_2$, then if we define

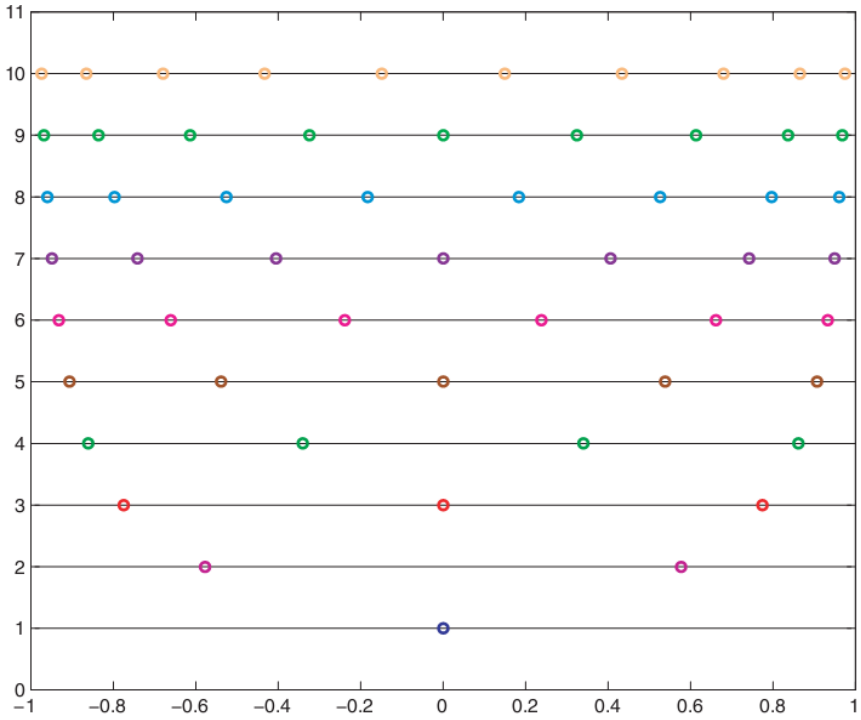
$$(2.6)\ E(R, f) = \sum\sum a_i b_j f(y_i, z_j) - \int_{R_1}\int_{R_2} \omega_1(y)\omega_2(z) f(y, z)dydz$$

Then we have

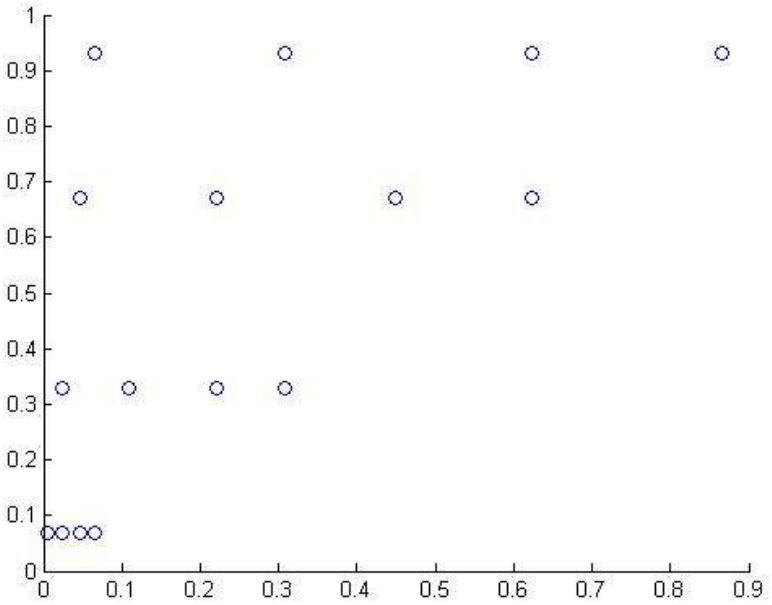
$$E(R, f) = \sum b_j E[R_1, f(y, z_j)] + \int_{R_1} \omega_1(y) E[R_2, f(y, z)] dy$$
$$= \sum a_i E[R_2, f(y_i, z)] + \int_{R_2} \omega_2(z) E[R_1, f(y, z)] dz \quad (2.7)$$

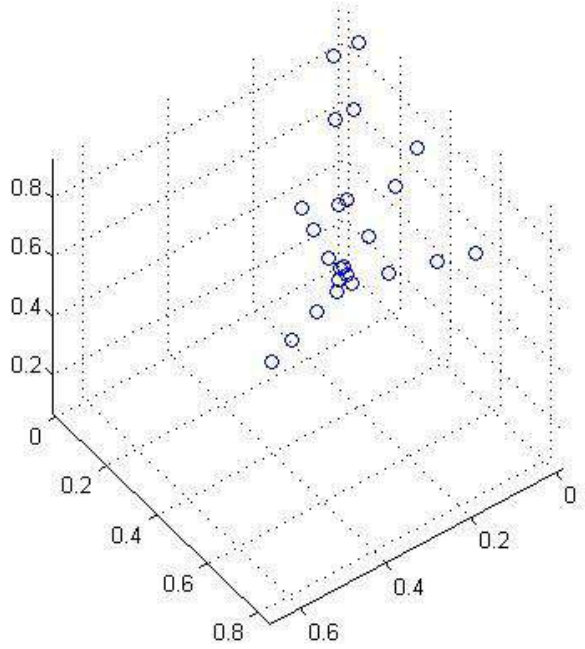
4. Our tasks

Task 1. Below is a figure showing Gaussian points in 1D. Please try to find Gaussian points in triangles in 2D and tetrahedron in 3D. And Plot those points on a reference triangle or tetrahedron.



Using the method before, we have get our results for 2D and 3D:





Task 2 &3.

We have three important function gldata2D gldata3D and gldata 4D. gldata2D returns a matrix of three columns, the first two columns are the xy sample point value, the third column is the weight. This function calls the matlab built-in function gldata (), which returns the n-order one-dimensional Gaussian points and weights.

gldata3D returns a matrix of four, the first three columns are the sample point of xyz values, and the fourth column is the right value. This function calls the function above gldata2D.

For gldata 2D, which is the basic one, we achieved it by from following code:

```
function m = gldata2D(a,b)
%return the data points in right triangular region%
% x >0; y>0; x+y <1;
n = a*b;
xgl = feval(symengine,'numeric::gldata',a,digits);
ygl  = feval(symengine,'numeric::gldata',b,digits);
wx = xgl(1);
wy = ygl(1);
x = xgl(2);
y = ygl(2);
w = zeros(n,1);
p = zeros(n,2);
for i=1:a
    for j=1:b
       p(b*(i-1)+j,:) =   [x(i)*y(j),y(j)];
       w(b*(i-1)+j) =  wx(i)*wy(j);
    end
end
```

```
 scatter(p(:,1),p(:,2));
 m = [p,w];
end
```

In task 2, our region is (1)x>0 0<y<1 y>x, the most important goal of the process is not in the area (1) of that part is a function of the value of the integration region Be symmetrical about the line y = x symmetry mapping, so that in the target region, so the question becomes the two times integral of the target area .

In task 2, our small program:

```
syms x y f1 f2
f1(x,y) = exp(x-y)*sin(x+y);
f2(x,y) = exp(y-x)*sin(x+y);
nx = 6;
ny = 7;
n = nx*ny;
evalf1 = zeros(n,1);
evalf2 = zeros(n,1);
 pw = gldata2D(nx,ny);
pnt =[pw(:,1),pw(:,1)];
wt = pw(:,3);
for i = 1:n
evalf1(i) = subs(f1,{x,y},{pnt(i,:)});
evalf2(i) = subs(f2,{x,y},{pnt(i,:)});
end
intg = (evalf1+evalf2)'*wt
```

In task 3 is seeking four-dimensional integration, even though we have obtained a four dimensional point, but since this is the theme of regional integration is not required when the sample target area. We cannot think of a simple way split, we think this case has split the light is not enough, and then should need to do our target area Topological Transformation job. We used a silly way, not very good, cannot guarantee accuracy. Process is equidistant sampling in each dimension, took a total of 6 * 5 * 5 * 5 = 750 points, and then find the function value, and finally add and quadrature. This may use Cartesian product, which have showed in our code rar. Finally we show our code for task 3.

```
syms x1 x2 x3 x4 f
k = pi /2;
f(x1,x2,x3,x4) =
k*cos(x1*x2*x3*x4)-7*k*x1*x2*x3*x4*sin(x1*x2*x3*x4)-6*k*((x1*x2*x3*x4)^2)*cos(x1*x2*x3
*x4)+k*((x1*x2*x3*x4)^3)*sin(x1*x2*x3*x4);

i = (1:6)';
j = (1:5)';
pnt = cartesianproduct(cartesianproduct(cartesianproduct(i,j),j),j);
intg = 0;
```

```
for i = 1:750
   intg = intg+subs(f,[x1,x2,x3,x4],{pnt(i,:)})/750;
end
intg
```

5. Conclusions

We study some knowledge about numerical methods for multiple dimensional integral and do some experiments for them by matlab with finishing our three task.

Reference:

[1]A.Glaser, X. Liu and V. Rokhlin, A fast algorithm for the calculation of the roots of special functions", SIAM Journal on Scientific Computing,29(2007), 1420-1438.

[2]J.CLERK-MAXWELL, "On approximate multiple integration between limits of summation," Cambridge Phil. Soc., Proc., v. 3, 1877, p. 39-47.

[3] G. W. TYLER, "Numerical integration of functions of several variables," Canadian Jn. Math., v. 5, 1953, p. 393-412