What is Gauss Quadrature?

The general Gauss quadrature formula is a numerical integration formula of the following form:

$$\int_a^b \mathrm{H}(x)\, \mathrm{f}(x)\, dx = \left(\sum_{i=1}^{n} w_i\, \mathrm{f}(x_i) \right) + R_n$$

where a and b are extended real numbers and H(x) is a non-negative weight function , whose integral over the interval [a,b] is positive. The set of n numbers x_i are called the nodes and the set of n numbers w_i are called the weights. The nodes and weights are chosen such the right hand side is exact for every polynomial f(x) of degree $2\,n-1$. The error term is R_n .

What is Gauss-Legendre Quadrature?

In the general Gauss quadrature formula choose H(x)=1 , a=-1 and b=1 and we have the Gauss-Legendre quadrature formula

$$\int_{-1}^{1} \mathrm{f}(x)\, dx = \left(\sum_{i=1}^{n} w_i\, \mathrm{f}(x_i) \right) + R_n$$

The Error term R_n

$$R_n = \frac{2^{(2\,n+1)}\,(n!)^4\,(\mathrm{D}^{(2\,n)})(f)(\xi)}{(2\,n+1)\,((2\,n)!)^3}$$

where $(\mathrm{D}^{(2\,n)})(f)(\xi)$ denotes the $2\,n$'th derivative of f evaluated at ξ , where ξ is a real number between -1 and 1 exclusive.

What are the nodes and weights?

The n nodes x_i are the roots of the n'th degree Legendre polynomial $P_n(x)$. The formula for the weights in terms of the nodes is given in Abramowitz and Stegun, on page 887, formula 25.4.29 as

$$w_i = \frac{2}{(1-x_i^2)\,\mathrm{D}(P_n)(x_i)^2}$$

where $\mathrm{D}(P_n)(x_i)$ denotes the derivative of the n'th degree Legendre polynomial evaluated at the i'th root.

What are Legendre polynomials?

The n'th order Legendre polynomial is a polynomial of degree n . The infinite set $P_i(x)$ for i from 0,1,2,3, ... forms an orthogonal basis for the square integratable real-valued functions defined on the interval [-1,1] . The inner product of P_i and P_j is defined by

$\int_{-1}^{1} P_i(x)\, P_j(x)\, dx$ which is 0 for i not equal to j. The Legendre polynomials are not orthonormal. They all satisfy the equation

$P_n(1) = 1$, for all n. The normalization condition is

$$\int_{-1}^{1} P_n(x)^2\,dx = \frac{2}{2\,n+1} \quad \text{for all n>=0 .}$$

What are the coefficients of the Legendre polynomials?

The formula for the explicit expression of the n'th order Legendre polynomial is

$$P_n(x) = \sum_{m=0}^{\mathrm{trunc}\left(\frac{n}{2}\right)} 2^{(-n)}\,(-1)^m\,\mathrm{binomial}(n, m)\,\mathrm{binomial}(2\,n-2\,m, n)\,x^{(n-2\,m)}$$

Use Maple's fsolve to compute the roots

The 17 point formula to 16 digits

```
> Digits := 16:
> p17 := orthopoly[P](17,x);
```

$$p17 := \frac{583401555}{32768}x^{17} - \frac{300540195}{4096}x^{15} + \frac{1017958725}{8192}x^{13} - \frac{456326325}{4096}x^{11} + \frac{929553625}{16384}x^{9} - \frac{66927861}{4096}x^{7} + \frac{20369349}{8192}x^{5} - \frac{692835}{4096}x^{3} + \frac{109395}{32768}x$$

```
> r17 := [fsolve(p17,x)];
```

$r17$:= [-.9905754753144173, -.9506755217687678, -.8802391537269859, -.7815140038968014, -.6576711592166908, -.5126905370864770, -.3512317634538763, -.1784841814958479, 0, .1784841814958479, .3512317634538763, .5126905370864770, .6576711592166908, .7815140038968014, .8802391537269859, .9506755217687678, .9905754753144173]

```
> p17d := diff(p17,x);
```

$$p17d := \frac{9917826435}{32768}x^{16} - \frac{4508102925}{4096}x^{14} + \frac{13233463425}{8192}x^{12} - \frac{5019589575}{4096}x^{10} + \frac{8365982625}{16384}x^{8} - \frac{468495027}{4096}x^{6} + \frac{101846745}{8192}x^{4} - \frac{2078505}{4096}x^{2} + \frac{109395}{32768}$$

```
> p17d := unapply(p17d,x):
> wf := x->2./(1-x^2)/p17d(x)^2:
> w17 := map(wf,r17);
```

$w17$:= [.02414830286921128, .05545952937435178, .08503614831991218, .1118838471956181, .1351363684687945, .1540457610768659, .1680041021564411, .1765627053669926, .1794464703562065, .1765627053669926, .1680041021564411, .1540457610768659, .1351363684687945, .1118838471956181, .08503614831991218, .05545952937435178, .02414830286921128]

How do we use the formula?

We just define a function f(x) and do the summation.

```
> f := x->exp(x)+cos(3*x):
> g17 := add(w17[i]*f(r17[i]),i=1..17);
g17 := 2.44448239266819346602
```

How close is the above g17 to the exact answer?

```
> exF := int(f(x),x=-1..1);
exF := exp(1)+2/3*sin(3)-exp(-1)
> exF16 := evalf(exF);
exF16 := 2.44448239266084772849
> err17 := exF16-g17;
err17 := -.734573753e-11
```

Computer Hardware and Software

All the timings were generated in Maple V release 5, running on a Silicon Graphics computer. The hardware contains a 180 MHZ IP27 processor, the CPU is a MIPS R10000 and the FPU is a MIPS R10010 . The main memory size is 256 Mbytes.

What can go wrong with the above?

On page 917 of Abramowitz and Stegun they give the 48 point Gaussian quadrature formula to 21 digits to the right of the decimal point. Lets see what happens when we try to find them using the above Maple statements.

```
> Digits := 21:
> st:=time(): p48:=orthopoly[P](48,x): tp48:=time()-st;
tp48 := .186
> st:=time(): r48:=[fsolve(p48,x)]: tr48:=time()-st;
tr48 := 45.776
> st:=time(): p48d:=diff(p48,x): p48d:=unapply(p48d,x):
  wf48:=x->2/(1-x^2)/p48d(x)^2: tp48d:=time()-st;
tp48d := .3e-2
> st:=time(): w48:=map(wf48,r48): tw48:=time()-st;
tw48 := .354
```

Check a few values

```
> r48[25];
.323801709628693620333e-1
> r48[48];
.998771007252426118601
> w48[25];
.647376968126839225028e-1
> w48[48];
.315331830158760836920e-2
```

w48[48] should be 0.00315 33460 52305 838633

We lost about 16 digits in the weight formula so it looks like we need 16 guard digits. Also the fsolve took a lot of time.

Legendre polynomial as a Gauss hypergeometric function

On page 779, formula 22.5.49 of Abramowitz and Stegen we find

$$P_n(x) = \mathrm{hypergeom}\left([-n, n+1], [1], \frac{1-x}{2}\right).$$

Lets try out the above formula.

```
> Digits := 16;
```

$$Digits := 16$$

```
> Ph := (n,x)->`evalf/hypergeom/kernel`([-n,n+1],[1],(1-x)/2):
> Ph(96,0.25);
Error, (in Ph) hypergeom: lack of convergence
```

It looks like we will have to try something else.

Jenkins-Traub algorithm for roots of polynomials

Let's try *traubjen* from the share library.

```
> with(share):
See ?share and ?share,contents for information about the share library
> readlib(traubjen):
> p48 := orthopoly[P](48,x):
> Digits:=21+16:
> st:=time(): j48 := traubjen(p48): tj48 := time()-st;
tj48 := 21.811
```

The traubjen procedure takes less than 1/2 the time of fsolve.

```
> j48 := sort(j48):
> wj48 := map(wf48,j48):
> Digits := 21: j48 := evalf(j48): wj48 := evalf(wj48):
```

Check some values.

```
> j48[48];
.998771007252426118601
> wj48[48];
.315334605230583863268e-2
```

Asymptotic formula for roots

The asymptotic formula for the roots of the n'th Legendre polynomial is formula 22.16.6 on page 787 of Abramowitz and Stegun.

$$x_m = \cos\left(\frac{(4\,m-1)\,\pi}{4\,n+2} + \frac{1\cot\left(\frac{(4\,m-1)\,\pi}{4\,n+2}\right)}{8\,n^2}\right) + \mathrm{O}(n^{(-3)})$$

In the above formula, *m* is an integer between 1 and *n*. Since the error term is O(1/n^3) this will only yield a few digits of accuracy.

Weirstrass algorithm for roots (Durand-Kerner algorithm)

See "The Mathematical Basis and a Prototype Implementation of a New Polynomial Rootfinder with Quadratic Convergence", by T.E. Hull and R. Mathon, in ACM Transactions on Mathematical Software, Sept. 1996, Vol. 22, Number 3, pages 261-280. This algorithm is similar to Newton's method, except that it approximates all the roots at each iteration. It replaces the derivative in Newton's formula with the derivative of the polynomial whose roots are exactly the polynomial which is formed by the product of all approximate linear factors. This method is slower than fsolve, with its default starting values. However when we use the above asymptotic formula to compute starting values for the roots and only compute the positive roots, then it is faster than the Jenkins-Traub algorithm traubjen.

```
> read GLwgsrg;
> Digits := 21:
> time(GLwgsrg(48,'W48','R48'));
.853
```

Eigenvalues of a symmetric tri-diagonal matrix

See the article "Calculation of gaussian quadrature rules" by G.H. Golub and J.H. Welsch, in Mathematics of Computation, 23, (April 1969), pages 221-230, or see the fortran subroutine GAUSSQ from netlib. This routine uses the implicit QL method to compute the eigenvalues of a symmetric tri-diagonal matrix that is constructed from the coefficients in the recurrence relation for the Legendre polynomials. The eigenvalues are the roots of the Legendre polynomial (the nodes in the Gauss-Legendre quadrature formula). The first component of each eigenvector is also computed. This can be scaled to the weights in the Gauss-Legendre quadrature formula. This has been translated into Maple. Let's see how fast it is?

```
> read gaussqM;
> Digits := 21:
> time(gaussqM(48,'qw48','qr48'));
3.994
```

So it is faster than traubjen , but slower than the Weirstrass algorithm. This method has some advantages over the methods that are based on finding the roots of explicit Legendre polynomials. It is stable, so that it only requires a few guard digits. It is an O(n^2) time complexity algorithm and O(n) space . It only uses 3 arrays of length n. The above assumes that the precision is small and fixed (double or quadruple precision). One disadvantage is that it computes all roots , instead of using the symmetry to compute only the positive roots . For n large enough it should be faster than the Weirstrass algorithm.

```
> time(GLwgsrg(512,'ww','wr'));
577.305
> time(gaussqM(512,'gw','gr'));
432.160
>
```

Legendre polynomial recurrence relation

The recurrence relation for the Legendre polynomials is given on page 782 , 22.7.10, of Abramowitz and Stegun as

$$(n+1)\,P_{n+1}(x) = (2\,n+1)\,x\,P_n(x) - n\,P_{n-1}(x)\,.$$

The two starting conditions are

$$P_0(x) = 1,\ P_1(x) = x\ .$$

Use recurrence relation to compute the nodes and weights

We will use the recurrence relation to compute values of the Legendre polynomials and their derivatives. Then we can use a precision doubling Newton method to compute the Gauss-Legendre nodes and weights. We use the asymptotic formula for the roots to start the iteration. The advantage of this method over other the previous methods is that we can vary the precision, since the recurrence relation is stable.

The 96 point formula to 21 digits

From Abramowitz and Stegun page 919
w[92] is listed as 0.00501 42027 42927 517693
but the last digit should be 2 .

```
> Digits := 25:
> read gIw;
> time(gwn(96,-1,1,w,r));
2.817
> for i from 92 to 92 do
    printf("%2d   %23.21f  %23.21f\n",i,r[i],w[i]);
   od;
92    .988054126329623799481   .005014202742927517692
> w[92];
.5014202742927517692470195e-2
```

Timings for recurrence relation method

```
> read gIw;
> Digits := 21:
> time(gIw(48,'w','r'));
.492
> time(gIw(128,'w','r'));
3.249
> time(gIw(256,'w','r'));
12.677
> time(gIw(512,'w','r'));
52.639
```

Legendre Differential Equation

The n'th Legendre polynomial satisfies the following ordinary differential equation.

$(1-x^2)\left(\frac{\partial^2}{\partial x\,\partial x}\,y(x)\right)-2\,x\left(\frac{\partial}{\partial x}\,y(x)\right)+n\,(n+1)\,y(x)=0$, with initial conditions $y(1)=1$ and $D(y)(1)=\frac{n\,(n+1)}{2}$.

We can numerically solve the above to compute values of the n'th Legendre polynomial and its derivative and then use Newton's method to find the roots of the Legendre polynomial. For a given small precision (double to quadruple) this method runs in linear time in n .

```
> read gde;
> Digits := 21:
> time(gde(128,'dw','dr'));
4.502
> time(gde(256,'dw','dr'));
8.946
> time(gde(512,'dw','dr'));
17.498
```

But at higher precisions it is slower than the recurrence relation method because it must use high precision numbers at every step, and cannot vary the precision as does the recurrence relation method, which can double the precision after each Newton iteration.

```
> Digits := 200:
> time(gIw(101,'w','r'));
11.757
> time(gde(101,'dw','dr'));
111.656
>
```

Numerical Integration Accuracy Comparison

We see how many digits of π that we can compute by limiting ourselves to 101 function evaluations of $f(x)=\frac{4}{1+x^2}$, We integrate $f(x)$ numerically over the interval [0,1] . Recall that

$$\pi=\int_0^1 \frac{4}{1+x^2}\,dx .$$

digits	Method:
5	trapezoidal rule
14	Simpson's rule
16	Romberg (97 points)
23	multiple Newton-Cotes (10 intervals of 10 panels each, 101 points, each interval uses 11 point closed formula)
28	Double Exponential (101 points with step size = 0.0775)
54	Newton-Cotes (101 point closed)
71	Chebyshev nodes (interpolate at the 101 roots of the degree 101 Chebyshev polynomial)
134	Gauss-Legendre quadrature

Application: truncated Fourier series

Suppose we want to numerically find the first few fourier series coefficients of an analytic periodic function. In this case we can approximate the integrals with the trapezoidal rule and expect high accuracy. We can also use the fast Fourier transform algorithm to compute O(n) fourier series coefficients in O(n ln(n)) steps. But if our function is analytic but not periodic then the error in this method O(1/n^2) for an n-point trapezoidal rule approximation. Since f(x) is analytic we expect that Gauss-Legendre quadrature could compute the first few fourier series coefficients faster than the FFT based algorithm could. The gaussian quadrature method would require O(n^2) time for O(n) coefficients at double (or quadruple precision). The FFT based algorithm would need O(1/sqrt(error)) points and this can be much larger than O(n^2) for 16 or more digits of accuracy.

```
> read gde; read scaleto; read trigmoments;
> Digits := 32+4:
> time(gde(2048,'w','r'));
152.545
> time(scaleto(2048,w,r,0,1,'sw','sr'));
.276
> f := proc(x) exp(-x) end:
> time(trigmoments(1024,'ca','sa',f,0,1,2048,sw,sr));
1342.989
```

Check the last cos and sin terms with the exact answer

```
> ec1024 := 2*int(f(x)*cos(2*Pi*1024*x),x=0..1); c1024 :=
  evalf(ec1024);
ec1024 := -2*(exp(-1)-1)/(1+4194304*Pi^2)
c1024 := .30540085638993576269567115633833816 8e-7
> es1024 := 2*int(f(x)*sin(2*Pi*1024*x),x=0..1); s1024 :=
  evalf(es1024);
es1024 := -4096*Pi*(exp(-1)-1)/(1+4194304*Pi^2)
s1024 := .19649435378373706861489292681328937 7e-3
> ca[1024]; c1024-ca[1024];
.30540085638993576269567115611470862e-7
.223629548e-34
> sa[1024]; s1024-sa[1024];
.19649435378373706861489292681326540 9e-3
.23968e-34
```

Compare the above errors with the following FFT based erorrs in the cos(2*Pi*2^10*x) term.

```
     n       time      error
   4096     1.619   -.713723495780095e-8
   8192     4.344   -.161951902639322e-8
  16384    10.338   -.39551691187104e-9
  32768    22.771   -.9830744261320e-10
  65536    49.153   -.2454132919258e-10
 131072   105.841   -.613311551335e-11
```

See "On the Gibbs Phenomenom and its resolution" by David Gottlieb and Chi-Wang Shu in SIAM Review, Vol. 39, No. 4, pages 644-668, Dec. 1997 , for an example where they require a few fourier series coefficients to high accuracy.