Trivial Numerical Integration

MJR

February 2014

There follows a discussion of the numerical integration of a Gaussian over a semi-infinite interval. The integrands chosen are

$$\exp(-x^2/\sigma^2) \text{ and } \exp(-x^2/\sigma^2) + 10^{-6}$$

with $\sigma \leq 1$. The range is zero to 100, which, for any reasonable precision, is equivalant to zero to infinity.

This integrations are evaluated a number of times. After each evaluation, σ is halved. The analytic results are effectively $0.5\sigma\sqrt{\pi}$, and the same plus 10^{-4}.

The difficulty for numerical integration is although the exponential is non-zero at one end-point (it is one at the origin), for small σ it decays quite rapidly to zero. Integrators tend not to sample the end point, so may end up sampling exclusively at points where the exponential is zero (less than smallest non-zero representable number). In this case the integrator will falsely conclude that the result is zero for the first integrand, and 10^{-4} for the second, and that there is no interesting region in which a higher sampling rate is necessary.

The default integration routines from GSL, NAG, Matlab and Mathematica were used, and the results and codes are discussed below.

1 Results

1.1 First Integrand

NAG (`d01ahf`) incorrectly gave zero as the answer for $\sigma = 0.0625$ and below. It gave no warning. NAG (`d01ajf`) did better, incorrectly giving zero for $\sigma = 0.015625$ and below. The documentation for `d01ahf` states 'an attempt is made to detect sharp end point peaks and singularities. . . '

GSL (`gsl_integration_qag` with `GSL_INTEG_GAUSS15`) incorrectly gave zero as the answer for $\sigma = 0.015625$ and below. It gave no warning.

Mathematica (`NIntegrate`) incorrectly gave zero as the answer for $\sigma = 0.015625$ and below. It did give a warning that the integral and error estimates were all zero.

Matlab (`quadgk`) gave incorrect, tiny, non-zero answers for $\sigma = 7.62939e-06$ and $3.8147e-06$, followed by zero for smaller σ. It gave no warning.

1.2 Second Integrand

NAG (`d01ahf`) gives incorrect answers (i.e. 10^{-4}) for $\sigma = 0.25$ and below, and NAG (`d01ajf`) remains as good as GSL.

GSL gives incorrect answers for $\sigma = 0.0625$ and below.

With $\sigma = 0.125$ and below, Mathematica believes that this integrates to 0.0001. The correct answer is just over 0.11. No warnings are given.

Matlab is still correct until σ reaches $7.62939e-06$.

Octave 3.6 gives similar results to Matlab for both integrals.

2 Conclusions

In all cases wrong answers with no warnings were readily triggered.

Assuming the integrator first attempts a 7 point Gauss rule with a 15 point Kronrod rule, and compares the answers, the closest point to the origin sampled will be approximately 0.42725. For the first integral, there will be confusion if the integrand evaluates to zero here, which is approximately the condition that

$$\begin{aligned} \exp(-x^2/\sigma^2) &< 10^{-310} \\ -x^2/\sigma^2 &< -714 \\ x/\sigma &> 26.7 \\ \sigma &< 0.016 \end{aligned}$$

For the second integral, confusion would occur if the integrand does not evaluate to something distinguishable from 10^{-6}, which is approximately

$$\begin{aligned} \exp(-x^2/\sigma^2) &< 10^{-21} \\ \sigma &< 0.061 \end{aligned}$$

This reflects what GSL seems to do. NAG seems worse, and Matlab remarkably good. None is perfect though.

3 Programs

3.1 NAG

```
module const
  double precision :: sigma
  contains
  function gauss(x)
    double precision gauss
    double precision, intent(in):: x
    gauss=exp(-x*x/(sigma*sigma))
    return
  end function
end module

program integrate
  use const
  use nag_library
  integer i,ifail,npts
  double precision x,err,pi

  pi=3.14159265358979d0
  sigma=1
  ifail=0

  do i=1,10
    x=d01ahf(0d0,100d0,1d-8,npts,err,gauss,100000,ifail)
    write(*,*)'sigma=',sigma
    write(*,*)'integral=',x,'err=',err*x,'ifail=',ifail
    write(*,*)'actual error',abs(x-0.5*sigma*sqrt(pi))
    sigma=sigma*0.5
  enddo
end
```

3.2 GSL

```
/* Compile with
    gcc  -I/usr/include/gsl int_gsl.c -lgsl -lgslcblas
*/

#include<stdio.h>
#include<math.h>

#include<gsl/gsl_errno.h>
#include<gsl_roots.h>
#include<gsl/gsl_integration.h>

double sigma;
double gauss(double x, void *p){
  return exp(-x*x/(sigma*sigma));
}

int main(){
  int i;

  double x,err,p,pi=3.14159265358979;
  int n;
  gsl_integration_workspace *work;
  gsl_function F;

  work=gsl_integration_workspace_alloc(100000);
  F.function=gauss;
  F.params=NULL;
  gsl_set_error_handler_off();

  sigma=1;

  for(i=0;i<10;i++){
    gsl_integration_qag(&F,0,100,0,1e-8,100000,GSL_INTEG_GAUSS15,
                        work,&x,&err);
    printf("sigma=%g\n integral=%g claimed error %g actual error %g\n",
           sigma,x,err,fabs(x-0.5*sigma*sqrt(pi)));
    sigma*=0.5;
  }

  gsl_integration_workspace_free(work);

  return 0;
}
```

3.3 Matlab

```
sigma=1;

for i=1:20

  gauss=@(x) exp(-(x*x)/(sigma*sigma));

  x=quadgk(@(xx)arrayfun(gauss,xx),0,100,'AbsTol', eps, ...
        'MaxIntervalCount', 100000);

  fprintf('sigma=%g\n integral=%g claimed error %g actual error %g\n', ...
        sigma,x,eps,abs(x-0.5*sigma*sqrt(pi)));

  sigma=0.5*sigma;

end
```

3.4 Mathematica

```
sigma = 1; Do[y = NIntegrate[Exp[-x*x/(sigma*sigma)], {x, 0, 100}];
 Print [sigma, " ", y]; Print [Abs[y - 0.5*sigma*N[Sqrt[Pi]]]];
 sigma = 0.5*sigma, {i, 10}]
```