- Richardson Extrapolation.
- Romberg Integration.
- Gaussian quadrature.

Richardson Extrapolation

Richardson extrapolation is a sequence acceleration method, used to improve the rate of convergence of a sequence.

Richardson Extrapolation

Richardson extrapolation is a sequence acceleration method, used to improve the rate of convergence of a sequence.

Let $A(h)$ be an approximation of A that depends on a positive step size h with an error formula of the form

$$A - A(h) = a_0 h^{k_0} + a_1 h^{k_1} + a_2 h^{k_2} + \cdots$$

where the a_i are unknown constants and the k_i are known constants such that $h^{k_i} > h^{k_{i+1}}$.

Richardson Extrapolation

Richardson extrapolation is a sequence acceleration method, used to improve the rate of convergence of a sequence.

Let $A(h)$ be an approximation of A that depends on a positive step size h with an error formula of the form

$$A - A(h) = a_0 h^{k_0} + a_1 h^{k_1} + a_2 h^{k_2} + \cdots$$

where the a_i are unknown constants and the k_i are known constants such that $h^{k_i} > h^{k_{i+1}}$.

The exact value sought can be given by

$$\begin{aligned} A &= A(h) + a_0 h^{k_0} + a_1 h^{k_1} + a_2 h^{k_2} + \cdots \\ &= A(h) + a_0 h^{k_0} + \mathcal{O}(h^{k_1}) \end{aligned}$$

Richardson Extrapolation

Using the step sizes h and h/t for some t, the two formulas for A are:

$$\begin{aligned} A =& A(h) + a_0 h^{k_0} + \mathcal{O}(h^{k_1}) \\ A =& A\left(\frac{h}{t}\right) + a_0 \left(\frac{h}{t}\right)^{k_0} + \mathcal{O}(h^{k_1}). \end{aligned}$$

Richardson Extrapolation

Using the step sizes h and h/t for some t, the two formulas for A are:

$$
\begin{aligned}
A =& A(h) + a_0 h^{k_0} + \mathcal{O}(h^{k_1}) \\
A =& A\left(\frac{h}{t}\right) + a_0 \left(\frac{h}{t}\right)^{k_0} + \mathcal{O}(h^{k_1}).
\end{aligned}
$$

Multiplying the second equation by t^{k_0} and subtracting the first equation gives

$$
(t^{k_0} - 1)A = t^{k_0} A\left(\frac{h}{t}\right) - A(h) + \mathcal{O}(h^{k_1})
$$

Richardson Extrapolation

Using the step sizes h and h/t for some t, the two formulas for A are:

$$A = A(h) + a_0 h^{k_0} + \mathcal{O}(h^{k_1})$$
$$A = A\left(\frac{h}{t}\right) + a_0 \left(\frac{h}{t}\right)^{k_0} + \mathcal{O}(h^{k_1}).$$

Multiplying the second equation by t^{k_0} and subtracting the first equation gives

$$(t^{k_0} - 1)A = t^{k_0} A\left(\frac{h}{t}\right) - A(h) + \mathcal{O}(h^{k_1})$$

which can be solved for A to give

$$A = \frac{t^{k_0} A\left(\frac{h}{t}\right) - A(h)}{t^{k_0} - 1} + \mathcal{O}(h^{k_1})$$

By this process, we have achieved a better approximation of A by subtracting the largest term in the error which was $\mathcal{O}(h^{k_0})$. This process can be repeated to remove more error terms to get even better approximations.

By this process, we have achieved a better approximation of A by subtracting the largest term in the error which was $\mathcal{O}(h^{k_0})$. This process can be repeated to remove more error terms to get even better approximations.

A general recurrence relation beginning with $A_0 = A(h)$ can be defined for the approximations by

$$A_{i+1}(h) = \frac{t^{k_i} A_i\left(\frac{h}{t}\right) - A_i(h)}{t^{k_i} - 1}$$

where k_{i+1} satisfies

$$A = A_{i+1}(h) + \mathcal{O}(h^{k_{i+1}})$$

- Romberg's method is used to estimate the definite integral

$$I = \int_a^b f(x)\,dx$$

by applying Richardson extrapolation repeatedly on the trapezium rule.

- Romberg's method is used to estimate the definite integral

$$I = \int_a^b f(x)\,dx$$

  by applying Richardson extrapolation repeatedly on the trapezium rule.
- The estimates generate a triangular array.

- Romberg's method is used to estimate the definite integral

$$I = \int_a^b f(x)\,dx$$

  by applying Richardson extrapolation repeatedly on the trapezium rule.
- The estimates generate a triangular array.
- Romberg's method evaluates the integrand at equally spaced points.

Romberg Integration

As already discussed in previous lecture, trapezoidal rule:

$$I_n^{(0)} = h[\frac{1}{2}f_0 + f_1 + \ldots + f_{n-1} + \frac{1}{2}f_n]$$

where $h = \frac{b-a}{n}$, $x_i = x_0 + ih$, $x_0 = a$, $x_n = b$.

As already discussed in previous lecture, trapezoidal rule:

$$I_n^{(0)} = h[\frac{1}{2}f_0 + f_1 + \ldots + f_{n-1} + \frac{1}{2}f_n]$$

where $h = \frac{b-a}{n}$, $x_i = x_0 + ih$, $x_0 = a$, $x_n = b$.

Error for this rule ($\mathcal{O}(h^2)$) only has even powers of h:

$$I = I_n^{(0)} + Ah^2 + Bh^4 + Ch^6 + \ldots$$

where A, B, C are related to derivatives of $f(x)$ at the end points and numerical weights. The exact expressions are called *Euler-Maclaurin formula*.

To obtain a more accurate estimate for I, we will eliminate the leading contribution to the error the term of order h^2, by taking n to be even and determining the trapezoidal rule for $\frac{n}{2}$ intervals as well as for n intervals.

To obtain a more accurate estimate for I, we will eliminate the leading contribution to the error the term of order h^2, by taking n to be even and determining the trapezoidal rule for $\frac{n}{2}$ intervals as well as for n intervals.

Since the width of one interval is now $2h$ we have

$$\begin{aligned} I^{(0)}_{\frac{n}{2}} =& 2h[\frac{1}{2}f_0 + f_1 + \ldots + f_{n-1} + \frac{1}{2}f_n] \\ I =& I^{(0)}_{\frac{n}{2}} + A(2h)^2 + B(2h)^4 + C(2h)^6 + \ldots \end{aligned}$$

Romberg Integration

To obtain a more accurate estimate for I, we will eliminate the leading contribution to the error the term of order h^2, by taking n to be even and determining the trapezoidal rule for $\frac{n}{2}$ intervals as well as for n intervals.

Since the width of one interval is now $2h$ we have

$$
\begin{aligned}
I^{(0)}_{\frac{n}{2}} =& 2h[\frac{1}{2}f_0 + f_1 + \ldots + f_{n-1} + \frac{1}{2}f_n] \\
I =& I^{(0)}_{\frac{n}{2}} + A(2h)^2 + B(2h)^4 + C(2h)^6 + \ldots
\end{aligned}
$$

Combining and eliminating the leading h^2 term:

$$
I = \frac{4I^{(0)}_n - I^{(0)}_{\frac{n}{2}}}{3} - 4Bh^4 - 20Ch^6 + \ldots
$$

As a result the next level of approximation becomes:

$$I_n^{(1)} = \frac{4I_n^{(0)} - I_{\frac{n}{2}}^{(0)}}{3}$$

As a result the next level of approximation becomes:

$$I_n^{(1)} = \frac{4I_n^{(0)} - I_{\frac{n}{2}}^{(0)}}{3}$$

The integral I can be written as:

$$I = I_n^{(1)} + B'h^4 + C'h^6 + \dots$$

with $B' = -4B$ and $C' = -20C$.

As a result the next level of approximation becomes:

$$I_n^{(1)} = \frac{4I_n^{(0)} - I_{\frac{n}{2}}^{(0)}}{3}$$

The integral I can be written as:

$$I = I_n^{(1)} + B'h^4 + C'h^6 + \ldots$$

with $B' = -4B$ and $C' = -20C$.

In terms of the weighted sum, this expression reduces to:

$$I_n^{(1)} = \frac{h}{3}[f_0 + 4f_1 + 2f_2 + \ldots + 2f_{n-1} + f_n]$$

which is the Simpson's rule!

Romberg Integration

One can keep repeating this to get the next approximation to I. Formulae differ from the Newton-Cotes. In general,

$$I_n^{(k)} = \frac{4^k I_n^{(k-1)} - I_{\frac{n}{2}}^{(k-1)}}{4^k - 1}$$

for $k = 1, 2, 3, \ldots$ which will have an error $\mathcal{O}(h^{2k+2})$.

One can keep repeating this to get the next approximation to I. Formulae differ from the Newton-Cotes. In general,

$$I_n^{(k)} = \frac{4^k I_n^{(k-1)} - I_{\frac{n}{2}}^{(k-1)}}{4^k - 1}$$

for $k = 1, 2, 3, \ldots$ which will have an error $\mathcal{O}(h^{2k+2})$.

As a result better approximations can be found by using the table:

n $\quad k \rightarrow$ $\downarrow$	0	1	2	3	$\ldots$
1	$I_1^{(0)}$				
2	$I_2^{(0)}$	$I_2^{(1)}$			
4	$I_4^{(0)}$	$I_4^{(1)}$	$I_4^{(2)}$		
8	$I_8^{(0)}$	$I_8^{(1)}$	$I_8^{(2)}$	$I_8^{(3)}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Romberg Integration

```python
import numpy as np

def romberg(a, b, eps, nmax, func):
    h = np.zeros(nmax)
    for i in range(0,nmax):
        h[i] = (b - a)/(2.**i)

    r = np.zeros((nmax, nmax))
    r[0,0] = (b - a)*(func(a) + func(b))/2.
    for j in range (1,nmax):
        subtotal = 0
        for i in range(0,2**(j-1)):
            subtotal = subtotal + func(a+(2*i+1)*h[j])
        r[j,0] = r[j-1,0]/2. + h[j]*subtotal
        for k in range(1,j+1):
            r[j,k] = (4**(k)*r[j,k-1]-r[j-1,k-1])/(4**(k)-1)
    return r

def f(x):
    return 1.0/(x*x)
```

n k	0	1	2	3	4	5
1	0.62500000000					
2	0.53472222222	0.50462962963				
4	0.50899376417	0.50041761149	0.50013681028			
8	0.50227085033	0.50002987904	0.50000403021	0.50000192259		
16	0.50056917013	0.50000194339	0.50000008102	0.50000001833	0.50000001086	
32	0.50014238459	0.50000012275	0.50000000137	0.50000000010	0.50000000003	0.50000000002

To reach close to machine accuracy with double precision, Romberg integration needs 64 intervals, while Simpson's rule would need about 1900 intervals, and the trapezium rule would need no less than 3.8×10^6 intervals

- Newton-Cotes Formulae

Gaussian Quadrature

- Newton-Cotes Formulae
  - Use evenly-spaced functional values.

- Newton-Cotes Formulae
  - Use evenly-spaced functional values.
  - Did not use the flexibility we have to select the quadrature points

- Newton-Cotes Formulae
  - Use evenly-spaced functional values.
  - Did not use the flexibility we have to select the quadrature points
- In fact a quadrature has several degrees of freedom.

$$I[f] = \sum_{i=1}^{m} c_i f(x_i)$$

A formula with m function evaluations requires $2m$ numbers to be specified, c_i and x_i

Gaussian Quadrature

- Select both these weights and locations so that a higher order polynomial can be integrated.

Gaussian Quadrature

- Select both these weights and locations so that a higher order polynomial can be integrated.
- Price: functional values must now be evaluated at nonuniformly distributed points to achieve higher accuracy.

Gaussian Quadrature

- Select both these weights and locations so that a higher order polynomial can be integrated.
- Price: functional values must now be evaluated at nonuniformly distributed points to achieve higher accuracy.
- Weights are no longer simple numbers.

Gaussian Quadrature

- Select both these weights and locations so that a higher order polynomial can be integrated.
- Price: functional values must now be evaluated at nonuniformly distributed points to achieve higher accuracy.
- Weights are no longer simple numbers.
- Usually derived for an interval such as [-1,1].

Gaussian Quadrature

- Select both these weights and locations so that a higher order polynomial can be integrated.
- Price: functional values must now be evaluated at nonuniformly distributed points to achieve higher accuracy.
- Weights are no longer simple numbers.
- Usually derived for an interval such as [-1,1].
- Other intervals [a,b] determined by mapping to [-1,1].

Gaussian Quadrature on [-1,1]

$$I[f] = \int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} c_i f(x_i) = c_1 f_1 + c_2 f_2 + \ldots + c_{n-1} f_{n-1} + c_n f_n$$

$$I[f] = \int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} c_i f(x_i) = c_1 f_1 + c_2 f_2 + \ldots + c_{n-1} f_{n-1} + c_n f_n$$

Two function evaluations: Choose (c_1, c_2, x_1, x_2) such that the method yields "exact integral" for $f(x) = x^0, x^1, x^2, x^3$

Gaussian Quadrature on [-1,1]

$$I[f] = \int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} c_i f(x_i) = c_1 f_1 + c_2 f_2 + \ldots + c_{n-1} f_{n-1} + c_n f_n$$

Two function evaluations: Choose (c_1, c_2, x_1, x_2) such that the method yields "exact integral" for $f(x) = x^0, x^1, x^2, x^3$
For $n = 2$, the method is accurate up to $2n - 1 = 3$ degree polynomial.

Finding quadrature nodes and weights

- One way is through the theory of orthogonal polynomials.

Finding quadrature nodes and weights

- One way is through the theory of orthogonal polynomials.
- Here we will do it via brute force.

Finding quadrature nodes and weights

- One way is through the theory of orthogonal polynomials.
- Here we will do it via brute force.
- Set up equations by requiring that the $2m$ points guarantee that a polynomial of degree $2m - 1$ is integrated exactly.

Finding quadrature nodes and weights

- One way is through the theory of orthogonal polynomials.
- Here we will do it via brute force.
- Set up equations by requiring that the $2m$ points guarantee that a polynomial of degree $2m-1$ is integrated exactly.
- In general process is non-linear:
  - Involves a polynomial function involving the unknown point and its product with unknown weight.

Finding quadrature nodes and weights

- One way is through the theory of orthogonal polynomials.
- Here we will do it via brute force.
- Set up equations by requiring that the $2m$ points guarantee that a polynomial of degree $2m - 1$ is integrated exactly.
- In general process is non-linear:
  - Involves a polynomial function involving the unknown point and its product with unknown weight.
  - Can be solved by using a multidimensional nonlinear solver

Finding quadrature nodes and weights

- One way is through the theory of orthogonal polynomials.
- Here we will do it via brute force.
- Set up equations by requiring that the $2m$ points guarantee that a polynomial of degree $2m-1$ is integrated exactly.
- In general process is non-linear:
  - Involves a polynomial function involving the unknown point and its product with unknown weight.
  - Can be solved by using a multidimensional nonlinear solver
  - Alternatively can sometimes be done step by step

Gaussian Quadrature on [-1,1]

For $n = 2$ $\quad \int_{-1}^{1} f(x)dx = c_1 f(x_1) + c_2 f(x_2)$

Gaussian Quadrature on [-1,1]

$$\text{For } n = 2 \quad \int_{-1}^{1} f(x)dx = c_1 f(x_1) + c_2 f(x_2)$$

Integral for $f(x) = x^0, x^1, x^2, x^3 \implies$ Four equations, four unknowns

Gaussian Quadrature on [-1,1]

$$\text{For } n = 2 \quad \int_{-1}^{1} f(x)dx = c_1 f(x_1) + c_2 f(x_2)$$

Integral for $f(x) = x^0, x^1, x^2, x^3 \implies$ Four equations, four unknowns

$$\left.\begin{aligned}
f = 1 &\implies \int_{-1}^{1} 1dx = 2 = c_1 + c_2 \\
f = x &\implies \int_{-1}^{1} xdx = 0 = c_1 x_1 + c_2 x_2 \\
f = x^2 &\implies \int_{-1}^{1} x^2 dx = \frac{2}{3} = c_1 x_1^2 + c_2 x_2^2 \\
f = x^3 &\implies \int_{-1}^{1} x^3 dx = 0 = c_1 x_1^3 + c_2 x_1^3
\end{aligned}\right\} \implies \begin{cases} c_1 = c_2 = 1 \\ x_1 = -x_2 = \frac{1}{\sqrt{3}} \end{cases}$$

Gaussian Quadrature on [-1,1]

$$\text{For } n = 2 \quad \int_{-1}^{1} f(x)dx = c_1 f(x_1) + c_2 f(x_2)$$

Integral for $f(x) = x^0, x^1, x^2, x^3 \implies$ Four equations, four unknowns

$$\left.\begin{array}{l} f = 1 \implies \int_{-1}^{1} 1dx = 2 = c_1 + c_2 \\ f = x \implies \int_{-1}^{1} xdx = 0 = c_1 x_1 + c_2 x_2 \\ f = x^2 \implies \int_{-1}^{1} x^2 dx = \dfrac{2}{3} = c_1 x_1^2 + c_2 x_2^2 \\ f = x^3 \implies \int_{-1}^{1} x^3 dx = 0 = c_1 x_1^3 + c_2 x_1^3 \end{array}\right\} \implies \begin{cases} c_1 = c_2 = 1 \\ x_1 = -x_2 = \frac{1}{\sqrt{3}} \end{cases}$$

$$I = \int_{-1}^{1} f(x)dx = f(\frac{1}{\sqrt{3}}) + f(-\frac{1}{\sqrt{3}})$$

Gaussian Quadrature on [-1,1]

For $n = 3$ $\quad \displaystyle\int_{-1}^{1} f(x)dx = c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3)$

Gaussian Quadrature on [-1,1]

For $n = 3$ $\quad \int_{-1}^{1} f(x)dx = c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3)$

$$
\left.\begin{aligned}
f = 1 &\implies \int_{-1}^{1} 1dx = 2 = c_1 + c_2 + c_3 \\
f = x &\implies \int_{-1}^{1} xdx = 0 = c_1 x_1 + c_2 x_2 + c_1 x_3 \\
f = x^2 &\implies \int_{-1}^{1} x^2 dx = \frac{2}{3} = c_1 x_1^2 + c_2 x_2^2 + c_3 x_3^2 \\
f = x^3 &\implies \int_{-1}^{1} x^3 dx = 0 = c_1 x_1^3 + c_2 x_2^3 + c_3 x_3^3 \\
f = x^4 &\implies \int_{-1}^{1} x^4 dx = \frac{2}{5} = c_1 x_1^4 + c_2 x_2^4 + c_3 x_3^4 \\
f = x^5 &\implies \int_{-1}^{1} x^5 dx = 0 = c_1 x_1^5 + c_2 x_2^5 + c_3 x_3^5
\end{aligned}\right\} \implies \begin{cases} c_1 = \frac{5}{9} \\ c_2 = \frac{8}{9} \\ c_2 = \frac{5}{9} \\ x_1 = \sqrt{\frac{3}{5}} \\ x_2 = 0 \\ x_3 = -\sqrt{\frac{3}{5}} \end{cases}
$$

Gaussian Quadrature on [-1,1]

$$I = \int_{-1}^{1} f(x)dx = \frac{5}{9}f(-\sqrt{\frac{3}{5}}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{\frac{3}{5}})$$

Gaussian Quadrature on [-1,1]

```
from numpy import ones,copy,cos,tan,pi,linspace

def gaussxw(N):

    # Initial approximation to roots of the Legendre polynomial
    a = linspace(3,4*N-1,N)/(4*N+2)
    x = cos(pi*a+1/(8*N*N*tan(a)))

    # Find roots using Newton's method
    epsilon = 1e-15
    delta = 1.0
    while delta>epsilon:
        p0 = ones(N,float)
        p1 = copy(x)
        for k in range(1,N):
            p0,p1 = p1,((2*k+1)*x*p1-k*p0)/(k+1)
        dp = (N+1)*(p0-x*p1)/(1-x*x)
        dx = p1/dp
        x -= dx
        delta = max(abs(dx))

    # Calculate the weights
    w = 2*(N+1)*(N+1)/(N*N*(1-x*x)*dp*dp)

    return x,w

def gaussxwab(N,a,b):
    x,w = gaussxw(N)
    return 0.5*(b-a)*x+0.5*(b+a),0.5*(b-a)*w

x,w = gaussxw(3)
print x
print w
```

Gaussian Quadrature on [a,b]

Define:

$$t = \frac{b-a}{2}x + \frac{b+a}{2}$$

At $x = -1$, $t = a$ and $x = 1$, $t = b$.

Gaussian Quadrature on [a,b]

Define:

$$t = \frac{b-a}{2}x + \frac{b+a}{2}$$

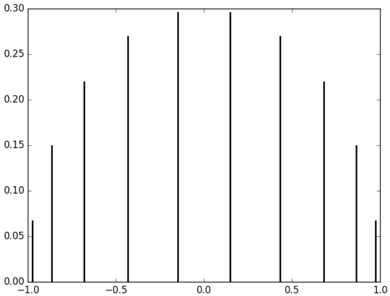
At $x = -1$, $t = a$ and $x = 1$, $t = b$.

$$I = \int_a^b = \int_{-1}^1 f(\frac{b-a}{2}x + \frac{b+a}{2})\frac{b-a}{2}dx = \int_{-1}^1 g(x)dx$$

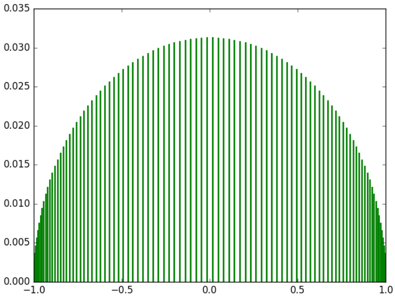
- Gaussian quadrature.
- Adaptive Integration.
- Special cases.
- Multiple integrals.

Gaussian quadrature

In general, in gaussian quadrature, the points are placed non-uniformly.



10 point quadrature



100 point quadrature

More points closer to the edges than in the middle.

Gaussian quadrature

```
from numpy import ones,copy,cos,tan,pi,linspace,sin

def gaussxw(N):
    a = linspace(3,4*N-1,N)/(4*N+2)
    x = cos(pi*a+1/(8*N*N*tan(a)))
    epsilon = 1e-15
    delta = 1.0
    while delta>epsilon:
        p0 = ones(N,float)
        p1 = copy(x)
        for k in range(1,N):
            p0,p1 = p1,((2*k+1)*x*p1-k*p0)/(k+1)
        dp = (N+1)*(p0-x*p1)/(1-x*x)
        dx = p1/dp
        x -= dx
        delta = max(abs(dx))
    w = 2*(N+1)*(N+1)/(N*N*(1-x*x)*dp*dp)
    return x,w

def gaussxwab(N,a,b):
    x,w = gaussxw(N)
    return 0.5*(b-a)*x+0.5*(b+a),0.5*(b-a)*w

def g(x):
   return 1./(x*x)

for i in range(1,15):
    x,w = gaussxwab(i,1.,2.)
    integ = 0.0
    for j in range(0,i):
        integ += g(x[j])*w[j]
    print " %2d  %18.16f"%(i, integ)
```

$$\int_1^2 \frac{1}{x^2} = 0.5$$

n	integral
1	0.4444444444444447
2	0.4970414201183431
3	0.4998740236835472
4	0.4999951475626201
5	0.4999998234768075
6	0.4999999938120432
7	0.4999999997886506
8	0.4999999999929189
9	0.4999999999997659
10	0.4999999999999911

$$\int_1^2 \frac{1}{x^2} = 0.5$$

n	integral
1	0.4444444444444447
2	0.4970414201183431
3	0.4998740236835472
4	0.4999951475626201
5	0.4999998234768075
6	0.4999999938120432
7	0.4999999997886506
8	0.4999999999929189
9	0.4999999999997659
10	0.4999999999999911

To reach close to machine accuracy with double precision, Romberg integration needs 64 intervals, while Simpson's rule would need about 1900 intervals, and the trapezium rule would need no less than 3.8×10^6

Gaussian quadrature needs 10 points.

- Non-adaptive quadrature: We continue to subdivide all subintervals, say by half, until overall error estimate falls below desired tolerance.

- Non-adaptive quadrature: We continue to subdivide all subintervals, say by half, until overall error estimate falls below desired tolerance.
- This method – although unbiased – may often be very inefficient if the function is not equally smooth throughout the domain of integration.

Adaptive integration

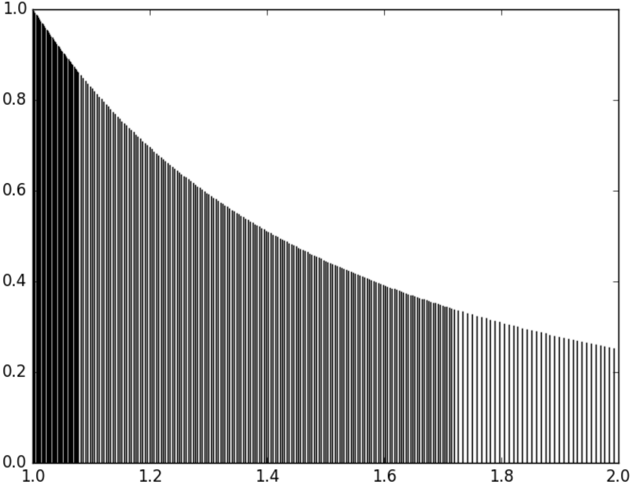
- Non-adaptive quadrature: We continue to subdivide all subintervals, say by half, until overall error estimate falls below desired tolerance.
- This method – although unbiased – may often be very inefficient if the function is not equally smooth throughout the domain of integration.
- Adaptive quadrature: The domain of integration is selectively refined. This reflects the behavior of particular integrand function on a specific subinterval

Adaptive Simpsons quadrature

```
def simpsons_rule(f,a,b):
    c = (a+b) / 2.0
    h3 = abs(b-a) / 6.0
    return h3*(f(a) + 4.0*f(c) + f(b))

def recursive_asr(f,a,b,eps,whole):
    c = (a+b) / 2.0
    left = simpsons_rule(f,a,c)
    right = simpsons_rule(f,c,b)
    if abs(left + right - whole) <= 15*eps:
        return left + right + (left + right - whole)/15.0
    return recursive_asr(f,a,c,eps/2.0,left) + \
               recursive_asr(f,c,b,eps/2.0,right)

def adaptive_simpsons_rule(f,a,b,eps):
    return recursive_asr(f,a,b,eps,simpsons_rule(f,a,b))

def f(x):
    return 1./(x*x)
print adaptive_simpsons_rule(f,1.,2.,1e-10)
```

Adaptive integration

Integrand is sampled densely in regions where it is difficult to integrate and sparsely in regions where it is easy.



Adaptive integration

- It is a very practical way to approach the problem of having an integrand which has some "interesting" behaviour.

Adaptive integration

- It is a very practical way to approach the problem of having an integrand which has some "interesting" behaviour.
- Can be quite efficient – uses 1384 sampling points.

Adaptive integration

- It is a very practical way to approach the problem of having an integrand which has some "interesting" behaviour.
- Can be quite efficient – uses 1384 sampling points.
- However, if interval of integration is very wide but "interesting" behavior of integrand is confined to narrow range, sampling by automatic routine may miss interesting part of integrand behavior, and resulting value for integral may be completely wrong.

Adaptive integration

- It is a very practical way to approach the problem of having an integrand which has some "interesting" behaviour.
- Can be quite efficient – uses 1384 sampling points.
- However, if interval of integration is very wide but "interesting" behavior of integrand is confined to narrow range, sampling by automatic routine may miss interesting part of integrand behavior, and resulting value for integral may be completely wrong.
- Plot to see the interesting part..

Special cases

Integrals with oscillating integrands:

$$\int_a^b f(x)\cos^n(\omega x)dx$$

Special cases

Integrals with oscillating integrands:

$$\int_a^b f(x) \cos^n(\omega x) dx$$

Use methods or programs specially designed to calculate integrals with oscillating functions:

- Filon's method
- Clenshaw-Curtis method

Special Cases: Improper Integrals

Improper integrals integrals whose integrand is unbounded in the range of integration.

- Change of variable
- Elimination of the singularity
- Ignoring the singularity
- Truncation of the interval
- Numerical evaluation of the Cauchy Principal Value

Change of variable

Sometimes it is possible to find a change of variable that eliminates the singularity.

Change of variable

Sometimes it is possible to find a change of variable that eliminates the singularity.

$$I = \int_0^1 x^{-\frac{1}{n}} g(x) dx$$

substituting $x = t^n$, the integral becomes:

$$I = n \int_0^1 t^{n-2} g(t^n) dt$$

which is proper!

Change of variable

Sometimes it is possible to find a change of variable that eliminates the singularity.

$$I = \int_0^1 x^{-\frac{1}{n}} g(x) dx$$

substituting $x = t^n$, the integral becomes:

$$I = n \int_0^1 t^{n-2} g(t^n) dt$$

which is proper!
But one has to be careful to not trade one problem for another:

$$I = \int_0^1 \log(x) f(x) dx$$

substituting $t = -\log(x)$,

$$I = -\int_0^\infty t e^{-t} f(e^{-t}) dt$$

Elimination of the singularity

General idea: Subtract from the singular integrand $f(x)$ a function, $g(x)$.

- $g(x)$ integral is known in closed form.
- $f(x) - g(x)$ is no longer singular.

This means that $g(x)$ has to mimic the behaviour of closely to its singular point.

Elimination of the singularity

General idea: Subtract from the singular integrand $f(x)$ a function, $g(x)$.

- $g(x)$ integral is known in closed form.
- $f(x) - g(x)$ is no longer singular.

This means that $g(x)$ has to mimic the behaviour of closely to its singular point.

$$\begin{aligned}\int_0^1 \frac{\cos x}{\sqrt{x}} dx = & \int_0^1 \frac{1}{\sqrt{x}} dx + \int_0^1 \frac{\cos(x) - 1}{\sqrt{x}} dx \\ = & 2 + \int_0^1 \frac{\cos(x) - 1}{\sqrt{x}} dx\end{aligned}$$

Elimination of the singularity

General idea: Subtract from the singular integrand $f(x)$ a function, $g(x)$.

- $g(x)$ integral is known in closed form.
- $f(x) - g(x)$ is no longer singular.

This means that $g(x)$ has to mimic the behaviour of closely to its singular point.

$$\begin{aligned}\int_0^1 \frac{\cos x}{\sqrt{x}}dx = &\int_0^1 \frac{1}{\sqrt{x}}dx + \int_0^1 \frac{\cos(x)-1}{\sqrt{x}}dx \\ =&2 + \int_0^1 \frac{\cos(x)-1}{\sqrt{x}}dx\end{aligned}$$

But $\cos(x) - 1 \approx -\frac{x^2}{2}$ near $x = 0$ making the new integrand proper that can be integrated easily.

Ignoring the singularity

- It is also possible to avoid the integrand singularities and apply the standard quadrature formulae. If we want to compute:

$$\int_0^1 f(x)dx$$

where $f(x)$ is unbounded in the neighbourhood of $x = 0$

Ignoring the singularity

- It is also possible to avoid the integrand singularities and apply the standard quadrature formulae. If we want to compute:
$$\int_0^1 f(x)dx$$
where $f(x)$ is unbounded in the neighbourhood of $x = 0$
- Then we set $f(0) = 0$ (or any other value) and use any sequence of quadrature methods.

Ignoring the singularity

- It is also possible to avoid the integrand singularities and apply the standard quadrature formulae. If we want to compute:

$$\int_0^1 f(x)dx$$

  where $f(x)$ is unbounded in the neighbourhood of $x = 0$
- Then we set $f(0) = 0$ (or any other value) and use any sequence of quadrature methods.
- Another option: use a sequence of quadrature methods that do not involve the value of $f(x)$ at $x = 0$.

Proceeding to the limit

$1 > r_1 > r_2 > \ldots$ is a sequence of points that converges to 0
(For e.g. if $r_n = 2^{-n}$, then

$$\int_0^1 f(x)dx = \int_{r_1}^1 f(x)dx + \int_{r_2}^{r_1} f(x)dx + \int_{r_3}^{r_2} f(x)dx + \ldots$$

Each of the above integrals is a proper integral.
The evalulation can be terminated if

$$\left| \int_{r_n}^{r_{n+1}} f(x)dx \right| \leq \epsilon$$

Truncation of the interval

$$\int_0^1 f(x)dx = \int_0^r f(x)dx + \int_r^1 f(x)dx$$

then if

$$\left| \int_0^r f(x)dx \right| \leq \epsilon,$$

we can simply evaluate

$$\int_r^1 f(x)dx$$

Truncation of the interval

$$\int_0^1 f(x)dx = \int_0^r f(x)dx + \int_r^1 f(x)dx$$

then if

$$\left| \int_0^r f(x)dx \right| \leq \epsilon,$$

we can simply evaluate

$$\int_r^1 f(x)dx$$

For example, assume $|g(x)| < 1 \forall x \in (0, 1]$, then

$$\left| \frac{g(x)}{x^{\frac{1}{2}} + x^{\frac{1}{3}}} \right| \leq \frac{1}{2x^{\frac{1}{2}}} \implies \left| \int_0^r \frac{g(x)}{x^{\frac{1}{2}} + x^{\frac{1}{3}}} dx \right| \leq \int_0^r \frac{dx}{2x^{\frac{1}{2}}} = r^{\frac{1}{2}}$$

If we chose $r \leq 10^{-8}$. we get an accuracy of 10^{-4}.

Reduction of the CPV to one-sided improper integral is possible.
Consider $f(x)$ unbounded in $x = c$ with $a < c < b$.
Suppose that $\mathsf{P} \int_a^b f(x)dx$ exists:

$$\mathsf{P} \int_a^b f(x)dx = \lim_{r \to 0} \left[\int_a^{c-r} f(x)dx + \int_{c+r}^b f(x)dx \right]$$

Reduction of the CPV to one-sided improper integral is possible.
Consider $f(x)$ unbounded in $x = c$ with $a < c < b$.
Suppose that P $\int_a^b f(x)dx$ exists:

$$\mathrm{P}\int_a^b f(x)dx = \lim_{r\to 0}\left[\int_a^{c-r} f(x)dx + \int_{c+r}^b f(x)dx\right]$$

Consider $c = 0$ and $b = -a$
Decompose $f(x)$ into its odd and even parts:

$$g(x) = \frac{1}{2}[f(x) - f(-x)] \quad h(x) = \frac{1}{2}[f(x) + f(-x)]$$

$$\int_{-a}^{-r} f(x)dx + \int_{+r}^{a} f(x)dx =$$

$$\int_{-a}^{-r} g(x)dx + \int_{+r}^{a} g(x)dx + \int_{-a}^{-r} h(x)dx + \int_{+r}^{a} h(x)dx =$$

$$2\int_{+r}^{a} h(x)dx$$

Therefore:

$$\mathsf{P}\int_{-a}^{a} f(x)dx = 2\lim_{r\to 0^+}\int_{r}^{a} h(x)dx$$

Numerical Evaluation of the Cauchy Principal Value

$$\mathsf{P}\int_{-1}^{1} \frac{1}{x} dx = 0$$

$$\mathsf{P}\int_{-1}^{1} \frac{e^x}{x} dx = 2\int_{0}^{1} \frac{\sinh(x)}{x} dx$$

Numerical Evaluation of the Cauchy Principal Value

The method of subtracting the singularity may also be used.

$$I(x) = \mathsf{P}\int_a^b \frac{f(t)}{t-x}dt \quad a < x < b$$

$$\begin{aligned} I(x) &= \int_a^b \frac{f(t)-f(x)}{t-x}dt + f(x)\mathsf{P}\int_a^b \frac{dt}{t-x} \\ &= \int_a^b \frac{f(t)-f(x)}{t-x}dt + f(x)\log\frac{b-x}{x-a} \end{aligned}$$

Numerical Evaluation of the Cauchy Principal Value

The method of subtracting the singularity may also be used.

$$I(x) = \mathsf{P}\int_a^b \frac{f(t)}{t-x}dt \quad a < x < b$$

$$\begin{aligned} I(x) &= \int_a^b \frac{f(t)-f(x)}{t-x}dt + f(x)\mathsf{P}\int_a^b \frac{dt}{t-x} \\ &= \int_a^b \frac{f(t)-f(x)}{t-x}dt + f(x)\log\frac{b-x}{x-a} \end{aligned}$$

Consider the function:

$$\begin{aligned} \phi(t,x) &= \frac{f(t)-f(x)}{t-x} \quad t \neq x \\ \phi(x,x) &= f'(x) \quad t = x \end{aligned}$$

and solve

$$\int_a^b \phi(t,x)dt$$

It maybe useful to consider:

$$\int_{x-h}^{x+h} \phi(t,x)dt = \int_{-h}^{h} \frac{f(t+x)-f(x)}{t}dt$$

If $f(t+x)$ can be expanded in a Taylor series:

$$\begin{aligned}\int_{x-h}^{x+h} \phi(t,x)dt =& \int_{-h}^{h} \left(f'(x) + \frac{tf''(x)}{2!} + \frac{t^2 f'''(x)}{3!} + \dots \right) \\ =& 2hf'(x) + \frac{h^3 f'''(x)}{9} + \dots\end{aligned}$$

Special cases: Indefinite integrals

$$\int_a^\infty f(x)dx \quad \int_{-\infty}^\infty f(x)dx$$

- Change of variables (common one is):

$$z = \frac{x-a}{1+x-a}$$

then

$$\int_a^\infty f(x)dx = \int_0^1 \frac{1}{(1-z)^2} f\big(\frac{z}{1-z} + a\big)dz$$

- For $\int_{-\infty}^{\infty}$ use

$$x = \frac{z}{1-z^2} \quad \text{or} \quad x = \tan z$$

Special cases: Indefinite integrals

$$\int_a^\infty f(x)dx \quad \int_{-\infty}^\infty f(x)dx$$

- Replace infinite limits of integration by carefully chosen finite values. Use asymptotic behaviour to evaluate "tail" contribution! (For $a \gg 1$):

$$\begin{aligned}\int_0^\infty \frac{\sqrt{x}}{x^2+1}dx &= \int_0^a \frac{\sqrt{x}}{x^2+1}dx + \int_a^\infty \frac{\sqrt{x}}{x^2+1}dx \\ &\approx \int_0^a \frac{\sqrt{x}}{x^2+1}dx + \int_a^\infty \frac{1}{x^{3/2}}dx \\ &= \int_0^a \frac{\sqrt{x}}{x^2+1}dx + \frac{2}{\sqrt{a}}\end{aligned}$$

- Use nonlinear quadrature rules designed for infinite range intervals.

Multiple Integrals

- Use automatic one-dimensional quadrature routine for each dimension, one for outer integral and another for inner integral.
- Monte-Carlo method (effective for large dimensions)

$$\int_0^1 dx_1 \int_0^1 dx_2 \cdots \int_0^1 dx_7 (x_1 + x_2 + \ldots + x_7)^2$$

Conclusions

- Analyze the integrand and plot it to see any potential pitfalls.

Conclusions

- Analyze the integrand and plot it to see any potential pitfalls.
- For smooth functions all methods work well.

Conclusions

- Analyze the integrand and plot it to see any potential pitfalls.
- For smooth functions all methods work well.
- For oscillating functions, functions with singularities, functions with high and narrow peaks, etc., one should use special methods and programs.

Conclusions

- Analyze the integrand and plot it to see any potential pitfalls.
- For smooth functions all methods work well.
- For oscillating functions, functions with singularities, functions with high and narrow peaks, etc., one should use special methods and programs.
- Very good set of quadrature methods available through SciPy called QUADPACK. For your projects, use these whenever possible.