- 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:

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

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

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

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 Integration

- 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 Integration

- 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 Integration

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

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

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

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

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

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

Combining and eliminating the leading h^2 term:

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

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:

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

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!

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 $\downarrow$ \ $k \rightarrow$	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

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

Gaussian Quadrature

- Newton-Cotes Formulae

Gaussian Quadrature

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

Gaussian Quadrature

- 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.

- 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.

- 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]

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

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 $f(x) = x^0, x^1, x^2, x^3$

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

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]

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]

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

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

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

Gaussian Quadrature on [-1,1]

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

Gaussian Quadrature on [-1,1]

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

$$\left.\begin{aligned}
f = 1 &\implies \int_{-1}^{1} 1dx = 2 = c_1 + c_2 + c_3 \\
f = x &\implies \int_{-1}^{1} xdx = 0 = c_1x_1 + c_2x_2 + c_1x_3 \\
f = x^2 &\implies \int_{-1}^{1} x^2dx = \frac{2}{3} = c_1x_1^2 + c_2x_2^2 + c_3x_3^2 \\
f = x^3 &\implies \int_{-1}^{1} x^3dx = 0 = c_1x_1^3 + c_2x_2^3 + c_3x_3^3 \\
f = x^4 &\implies \int_{-1}^{1} x^4dx = \frac{2}{5} = c_1x_1^4 + c_2x_2^4 + c_3x_3^4 \\
f = x^5 &\implies \int_{-1}^{1} x^5dx = 0 = c_1x_1^5 + c_2x_2^5 + c_3x_3^5
\end{aligned}\right\} \implies \begin{cases} c_1 = \frac{5}{9} \\ c_2 = \frac{8}{9} \\ c_2 = \frac{5}{9} \\ x_1 = \sqrt{\frac{3}{5}} \\ x_2 = 0 \\ x_3 = -\sqrt{\frac{3}{5}} \end{cases}$$

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]

```python
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 $x = 1$, $t = b$.

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