Section 2

# Integration

Numerical integration of a function f(x) from x=a to x=b is written as

$$I = \int_a^b f(x)dx = \sum_{i=0}^{n-1} w_i f(x_i) \qquad (1)$$

where $w_i$ are the weights and $x_i$ are abscissas.

The integration schemes can be classified into two schemes:

Simple methods: Here we choose $x_i$ as evenly spaced *n* points, and then compute $w_i$. If error is beyond admissible limit, then the number of points is increased.

Complex methods: Here $x_i$ and $w_i$ are chosen in such a way the integral has minimum error.

**Newton-Cotes Formulas**

Newton-Cotes scheme belongs to the class of simple methods. Here we divide the interval $(a, b)$ into $(n-1)$ equal divisions, that is, $(b-a)/(n-1) = h$. The abscissas are located at $x_j = a + jh$ where $j = 0 : (n-1)$.

We approximate the function f(x) using the Lagrange interpolation:

$$P(x) = \sum L_j(x) f(x_j),$$

which is substituted in Eq. (1) that yields

$$I = \int_b^a f(x)dx \approx \int_b^a P(x)dx = \sum_{j=0}^{n-1} f(x_j) \int_a^b L_j(x)dx$$

$$= (b-a) \sum_{j=0}^{n-1} C_j^{(n)} f(x_j).$$

where

$$C_j^{(n)} = \frac{1}{b-a} \int_b^a L_j(x)dx.$$

Since $L_j(x)$ are independent of the data $f(x_j)$, we conclude that $C_j^n$'s are independent of $f(x_j)$. Using $f(x) = 1$, we conclude that

$$\sum_j C_j^{(n)} = 1.$$

Note that $(b-a)C_j^{(n)}$ are the weights.

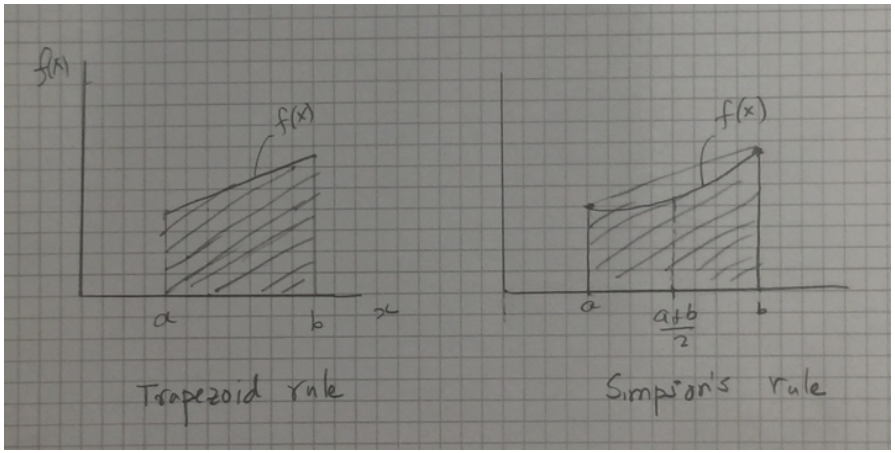
Now let us compute the $C_j^{(n)}$ for some of the Legendre polynomials. For two points, the polynomial is

$$P_2(x) = \frac{x-b}{a-b} f(a) + \frac{x-a}{b-a} f(b).$$

When we integrate the above, we obtain

$$I = \int_b^a P(x)dx = \frac{h}{2}(f(a) + f(b)).$$

This method, called trapezoid rule, is accurate for linear functions.




Note that the error in the above computation is

$$E = \int_a^b \frac{f''(\zeta)}{2!}(x-a)(x-b)dx = -\frac{1}{12}h^3 f''(\zeta).$$

When we use $f(x) = P_3(x)$ using the points x=a, (a+b)/2, b, we obtain

$$I = \int_b^a P(x)dx = \frac{h}{3}(f(a) + 4f((a+b)/2) + f(b)),$$

This method is called Simpson's rule, and it is accurate for polynomials up to third order.

**Figure**

The higher-order terms get more complex. Here we list $C_i$ in Table 1.

| n | N | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 1 | 1 | | | | | |
| 3 | 6 | 1 | 4 | 1 | | | | |
| 4 | 8 | 1 | 3 | 3 | 1 | | | |
| 5 | 90 | 7 | 32 | 12 | 32 | 7 | | |
| 6 | 288 | 19 | 75 | 50 | 50 | 75 | 19 | |
| 7 | 840 | 41 | 216 | 27 | 272 | 27 | 216 | 41 |

## Error analysis for Newton-Cote's method

In an earlier chapter we derived that the error in Lagrange interpolation with *n* points is

$$E(x) = f(x) - P(x) = \frac{f^{(n)}(\zeta)}{n!} \prod (x - x_i)$$

Hence the error in the integration by Newton-Cotes scheme is

$$E = \int_a^b E(x)dx = \frac{f^{(n)}(\zeta)}{n!}\int_a^b dx \prod (x - x_i)$$

We observe that for even *n*, the above integration is proportional to $h^{n+1}$, but it vanishes for odd *n.* For odd *n,* the error comes from the next term of the expansion that passes through the points (a,a+h,a+2h,...,b,b+h). The error estimate is

$$E = \int_a^b E(x)dx = \frac{f^{(n+1)}(\zeta)}{(n+1)!}\int_a^b dx \left[(x - b - h)\prod (x - x_i)\right].$$

The above integrals yield the errors as

| *n* | Error |
|---|---|
| $P_2$ | $(h^3/12)f^{(2)}(\zeta)$ |
| $P_3$ | $(h^5/90)f'^{(4)}(\zeta)$ |
| $P_4$ | $(3h^5/80)f^{(4)}(\zeta)$ |
| $P_5$ | $(8h^7/945)f'^{(6)}(\zeta)$ |
| $P_6$ | $(275h^7/12096)f'^{(6)}(\zeta)$ |
| $P_7$ | $(9h^9/1400)f'^{(8)}(\zeta)$ |

We illustrate the above error estimates using trapezoid and Simpson methods. For the trapezoid method

$$E_2 = \frac{f^{(2)}(\zeta)}{2!}\int_a^b dx(x - a)(x - b).$$

We choose a=0 and b=1, hence h=1 that yields

$$E_2 = \frac{f^{(2)}(\zeta)}{12}.$$

For n=3, the leading-order Legendre expansion yields

$$E_2 = \frac{f^{(3)}(\zeta)}{3!}\int_0^1 dx x(x - 1/2)(x - 1) = 0.$$

Hence, for estimation, we pick the next polynomial using points (0,1/2,1,3/2). The error is

$$E_2 = \frac{f^{(4)}(\zeta)}{4!}\int_0^1 dx x(x - 1/2)(x - 1)(x - 3/2) = \frac{h^5}{90}f^{(4)}(\zeta),$$

where h=1/2. Since $f^4(x)=0$ for a third-order polynomial, Simplson's method is accurate for polynomials up to cubic order.

We can compute the coefficients of Table 1 as well as error formula using Sympy.

```
x,a,b,h = symbols(('x','a','b','h'))

n = 4

h = (b-a)/(n-1)

xarray = []

for i in range(n):

   xarray.append(a+i*h)

def Newton_cotes_coeff(xarray):

   n = len(xarray)
```

```
    coeff = []
    for j in range(n):
        numr = 1; denr = 1;
        for i in range(n):
            if (j != i):
                numr *= (x-xarray[i])
                denr *= (xarray[j]-xarray[i])
        Cj = simplify(integrate(numr/denr, (x,a,b))/(b-a))
        coeff.append(Cj)

    factor = 1
    for i in range(n):
        factor *= (x-xarray[i])

    if (n%2 == 1):
        error_factor = simplify(integrate(factor*(x-b-h), (x,a,b))/
(h**(n+2)*factorial(n+1)))
    else:
        error_factor = simplify(integrate(factor, (x,a,b))/
(h**(n+1)*factorial(n)))
    return coeff, error_factor
```

Example: $\int_0^{\pi/2} dx \sin x$

**Newton_cotes.py**

```
# assume a single interval
# Solve int_a^b f(x)dx using m-th order Newton-Cotes method
def Newton_cotes(m, f, a, b):
  h=(b-a)/(m-1)

  if (m==2):
    return (b-a)*(f(a)+f(b))/2
  elif (m==3):
    return (b-a)*(f(a)+4*f((a+b)/2)+f(b))/6
  elif (m==4):
    return (b-a)*(f(a)+3*(f(a+h)+f(a+2*h))+f(b))/8
  elif (m==5):
    return (b-a)*(7*(f(a)+f(b)) + 32*(f(a+h)+f(b-h)) + 12*f(a+2*h))/
90
  elif (m==6):
    return (b-a)*(19*(f(a)+f(b)) + 75*(f(a+h)+f(b-h)) +
50*(f(a+2*h)+f(a+3*h)))/288

# (b-a) divided into n-1 interval; given n points
# Trapezoid rule
def Newton_cotes2(f, a, b, n):
  h = (b-a)/(n-1)
  ans = 0
  for i in range(n):
    ans += f(a+i*h)

  ans -= (f(a)+f(b))/2
  return h*ans


def f(x):
  return np.sin(x)
```

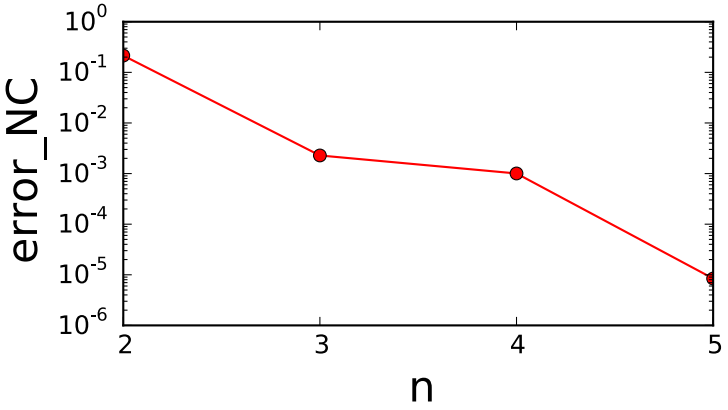
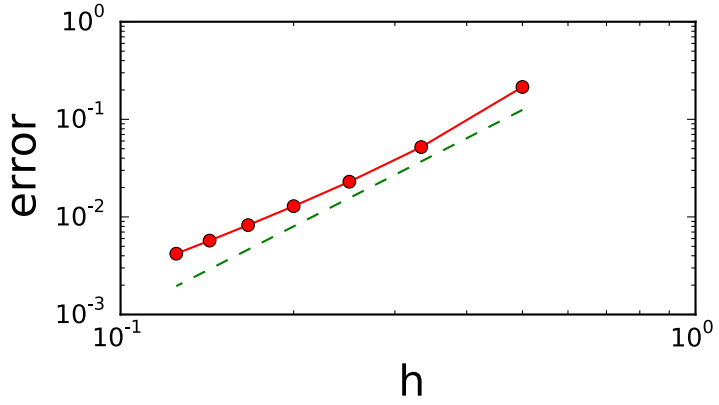



Fig: Error in Newton-Cotes method vs. degree of the polynomial, n, used.

The figure shows that error drops when we go from 2 to 3, but flattens from 3 to 4. See the error table for the reasons.

We can also integrate using smaller h, but summing over all the interval. We perform the above using trapezoid rule. The following figure illustrates how the error decreases with h as $h^3$, as h is decreased. The green dashed line represents $h^3$.



**Gaussian Quadrature**

In this method, we choose both $x_m$ and $w_m$ of

$$I = \int_a^b f(x)dx \approx \sum_{m=0}^{N-1} w_m f(x_m) \quad (1)$$

so that the integral is accurate. Thus we have 2*N* unknowns. If we demand exact quadrature for $f(x) = 1, x, x^2, \ldots, x^{2N-1}$, we will have 2*N* equations using which we can obtain the aforementioned 2*N* unknowns.

**Example**: Work out for *N*=2 for [-1,1].

**Solution**: We require that the integral is exact for $f(x) = 1, x, x^2, x^3$, which yields the following four equations:

$$2 = w_0 + w_1$$

$$0 = w_0 x_0 + w_1 x_1$$

$$\frac{2}{3} = w_0 x_0^2 + w_1 x_1^2$$

$$0 = w_0 x_0^3 + w_1 x_1^3,$$

whose solutions are $w_0 = w_1 = 1$, and $x_1 = -x_0 = 1/\sqrt{3}$.

Hence

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

The above integral is exact as long as $f(x)$ is a polynomial of degree 3 or less. We verify for $f(x) = x^2 + \frac{1}{2}x^3$, for which the RHS of the above equation yields 2/3, which is the exact integral.

The above procedure however is quite tedious for a general function. In the following discussion, we provide a general formulation based on orthogonal polynomials. Note that our formulation should yield an exact quadrature for any polynomial of degree *2N*-1 or less. We denote this function by f(x).

Suppose that the orthogonal polynomials used in our method is $\phi_i(x)$, and they satisfy the following orthogonality relation:

$$\int_a^b w(x)\phi_i(x)\phi_j(x) = \delta_{ij}\gamma_i,$$

where $\gamma_i$ are a constants, $i = 0 : (N-1)$, and $w(x)$ is the weight function. Note that not all the polynomials need to be of (N-1)th order. However, we demand that at least one of them is of (N-1)-order. In this scheme the integral is

$$I = \sum_{j=0}^{N-1} w_j f(x_j),$$

where $x_j$'s are the roots of $\phi_N(x)$, and $w_j$'s are

$$w_j = -\frac{a_N \gamma_N}{\phi'_N(x_j)\phi_{N+1}(x_j)}$$

We provide the proof of the above scheme in two steps:

Step 1: We write the function $f(x)$ as

$$f(x) = q_{N-1}(x)\phi_N(x) + r_{N-1}(x), \qquad (2)$$

where $\phi_N(x)$ is the N-th order polynomial, $q_{N-1}$ is the quotient, and $r_{N-1}$ is the remainder. Both $q_{N-1}$ and $r_{N-1}$ are polynomials of order (N-1), hence

$$q_{N-1} = \sum_{j=0}^{N-1} a_j \phi_j(x),$$

where $a_j$'s are the coefficients. Integration of Eq. (2) yields

$$\int_a^b w(x)f(x)dx = \int_a^b w(x)q_{N-1}(x)\phi_N(x)dx + \int_a^b w(x)r_{N-1}(x)dx. \quad (3)$$

The integral

$\int_a^b w(x)q_{N-1}\phi_N(x)dx = \sum_j a_j \int_a^b w(x)\phi_j(x)\phi_N(x)dx = 0$ due to orthogonality property. Hence

$$\int_a^b w(x)f(x)dx = \int_a^b w(x)r_{N-1}(x)dx.$$

Step 2: The N-th order polynomial $\phi_N(x)$ has N roots. We choose these roots, $x_j (j = 0 : N)$, as the abscissa, hence $\phi_N(x_j) = 0$. The function $f(x)$ given at these $x_j$'s , i.e., $y_j = f(x_j)$. Using Eq. (2)

$$f(x_j) = q_{N-1}(x_j)\phi_N(x_j) + r_{N-1}(x_j) = r_{N-1}(x_j).$$

Expand $r_{N-1}(x)$ using Lagrange interpolation

$$r_{N-1}(x) = \sum_j r_{N-1}(x_j)L_j(x),$$

substitution of which in Eq. (3) yields

$$\int_a^b w(x)f(x)dx = \int_a^b w(x)r_{N-1}(x)dx = \sum_j r_{N-1}(x_j)\int_a^b w(x)L_j(x)dx$$

$$= \sum_j w_j r_{N-1}(x_j) = \sum_j w_j f(x_j),$$

where $w_j = \int_a^b w(x)L_j(x)dx$. Using

$$\prod_{i,i\neq j}(x - x_i) = \frac{\prod(x - x_i)}{x - x_j} = \frac{\phi_N(x)}{a_N(x - x_j)},$$

where $a_N$ is the coefficient of $x^N$ in $\phi_N(x)$. Using L'Hospital rule, we take the limit $x \to x_j$, which yields

$$\prod_{i,i\neq j}(x_j - x_i) = \frac{\phi'_N(x_j)}{a_N}.$$

Therefore,

$$w_j = \int_a^b dx\, w(x)\frac{\prod_{i,i\neq j}(x - x_i)}{\prod_{i,i\neq j}(x_j - x_i)} = \frac{1}{\phi'_N(x_j)}\int_a^b \frac{w(x)\phi_N(x)}{x - x_j}dx. \quad (4)$$

Using Christoffel Darboux identity we can simplify it further to

$$\sum_{i=0}^{N} \frac{\phi_i(x)\phi_i(y)}{\gamma_i} = \frac{\phi_{N+1}(x)\phi_N(y) - \phi_N(x)\phi_{N+1}(y)}{a_N\gamma_N(x - y)} \quad (5)$$

where $\int_a^b w(x)\phi_i(x)\phi_j(x) = \delta_{ij}\gamma_i$ and

$$a_m = \frac{A_{m+1}}{A_m}$$

with $A_m$ as the coefficient of $x^m$ in $\phi_m(x)$. Choosing $y = x_j$, a zero of $\phi_N(x)$, substitution of which in Eq. (5) yields

$$\sum_{i=0}^{N} \frac{\phi_i(x)\phi_i(x_j)}{\gamma_i} = -\frac{\phi_N(x)\phi_{N+1}(x_j)}{a_N\gamma_N(x - x_j)}.$$

Integration of the above wrt $\int_a^b dx\, w(x)\phi_0(x)\ldots$ and using the fact that $\phi_0(x) = \text{const}$ yields

$$\phi_0(x_j) = -\frac{\phi_{N+1}(x_j)}{a_N\gamma_N}\phi_0(x)\int_a^b dx\,\frac{w(x)\phi_N(x)}{(x - x_j)}.$$

Hence

$$\int_a^b dx\,\frac{w(x)\phi_N(x)}{(x - x_j)} = -\frac{a_N\gamma_N}{\phi_{N+1}(x_j)}.$$

Hence

$$w_j = -\frac{a_N\gamma_N}{\phi'_N(x_j)\phi_{N+1}(x_j)}. \qquad (6)$$

We state without proof that the error in Gaussian quadrature is

$$E_{\text{int}} = \frac{f^{(2n)}(\zeta)}{(2n)!}\int_a^b w(x)W(x)dx \qquad (7)$$

[NOT] **Alternate proof:**

We can also use Hermite's interpolation to derive the formula for Gaussian quadrature:

$$f(x) \approx P(x) = \sum_j U_j(x)y_j + \sum_j V_j(x)y'_j \quad (1)$$

with

$$U_j(x) = [1 - 2L'_j(x_j)(x - x_j)][L_j(x)]^2;$$

$$V_j(x) = (x - x_j)[L_j(x)]^2$$

The integral is

$$\int_a^b w(x)f(x)dx \approx \int_a^b w(x)P(x)dx$$

$$= \sum_j y_j\int_a^b w(x)U_j(x)dx + \sum_j y'_j\int_a^b w(x)V_j(x)dx$$

We choose the abscissa in such a way that the second integral is zero, i.e.,

$$\int_a^b w(x)V_j(x)dx = \int_a^b w(x)(x - x_j)[L_j(x)]^2dx = 0.$$

Therefore,

$$\int_a^b w(x)P(x)dx = \sum_j y_j\int_a^b w(x)U_j(x)dx$$

$$= \sum_j y_j \int_a^b w(x)[L_j(x)]^2 dx = \sum_j y_j w_j$$

Thus the weight is

$$w_j = \int_a^b w(x)[L_j(x)]^2 dx$$

Compare it with the other formulas derived in the previous derivation.

**Error:**

We derived in the Hermite's polynomial that the error is

$$E = f(x) - P(x) = \frac{f^{(2n)}(\zeta)}{(2n)!} W(x),$$

where

$$W(x) = \prod_{i=0}^{n-1} (x - x_i)^2$$

Hence the error in the quadrature would be

$$E_{\text{int}} = \frac{f^{(2n)}(\zeta)}{(2n)!} \int_a^b w(x) W(x) dx$$

For integration of smooth functions within a limit, it is best to use **Legendre polynomials**, for which $w(x) = 1$, $a = -1$, and $b = 1$. We will show later how the integration scheme can be generalized to arbitrary a and b.

The leading Legendre polynomials are

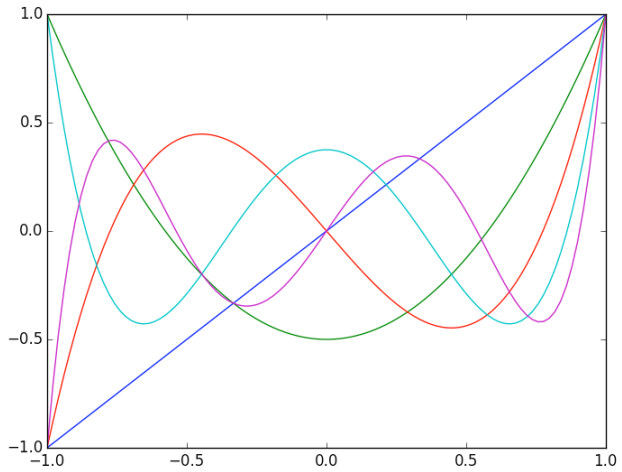
$$\phi_0(x) = 1$$

$$\phi_1(x) = x$$

$$\phi_2(x) = \frac{3x^2 - 1}{2}$$

$$\phi_3(x) = \frac{5x^3 - 3x}{2}$$

PLOTS OF LEGENDRE POL



Also,

$$\gamma_j = \frac{2}{2j+1}$$

$$A_j = \frac{(2j)!}{2^j (j!)^2},\ a_N = \frac{2}{N+1}$$

Hence, using Eq. (6) we obtain

$$w_j = -\frac{2}{(j+1)\phi'_N(x_j)\phi_{N+1}(x_j)}$$

Using an identity

$$(1-x^2)\phi'_j(x) = (j+1)x\phi_j(x) - (j+1)\phi_{j+1}(x)$$

and substituting $j = N; x = x_j$ we obtain

$$w_j = \frac{2}{(1-x_j^2)[\phi'_N(x_j)]^2}$$

N=2: $\phi_2(x) = \frac{3x^2-1}{2}$, $\phi'_2(x) = 3x$

The zeros of the polynomials at $x = -1/\sqrt{3}, 1/\sqrt{3}$. Also, $\phi'_2(\pm(1/\sqrt{3})) = \pm\sqrt{3}$. Therefore, $w_0 = w_1 = 1$.

N=3, $\phi_2(x) = \frac{5x^3-3x}{2}$.

The zeros of the polynomials at $x = -\sqrt{3/5}, 0, \sqrt{3/5}$, substitution of which yields:

$w_0 = w_2 = 5/9, w_1 = 8/9$.

We list the abscissa and weights in the following table:

| n | $x_i$ | $w_i$ |
|---|---|---|
| 2 | ±0.5773503 | 1 |
| 3 | 0 | 0.888889 |
| | ±0.774579 | 0.555556 |
| 4 | ±0.339981 | 0.652145 |
| | ±0.861136 | 0.347855 |
| 5 | 0 | 0.568889 |
| | ±0.538469 | 0.478629 |
| | ±0.90618 | 0.236927 |

**Example code**

```
# gaussian_quad.py

# sum(w_i f(x_i))

def my_gauss_quadrature(warray, xarray, f):
    yarray = np.zeros(len(xarray))
    for i in range(len(xarray)):
            yarray[i] = f(xarray[i])
    return sum(warray*yarray)
```

```
xarray = np.array([-1/np.sqrt(3),1/np.sqrt(3)])

warray = np.array([1,1])
```

**Example 1:** $\int_{-1}^{1} x^2 dx = 0.6666667$, exact value with $n = 2$.

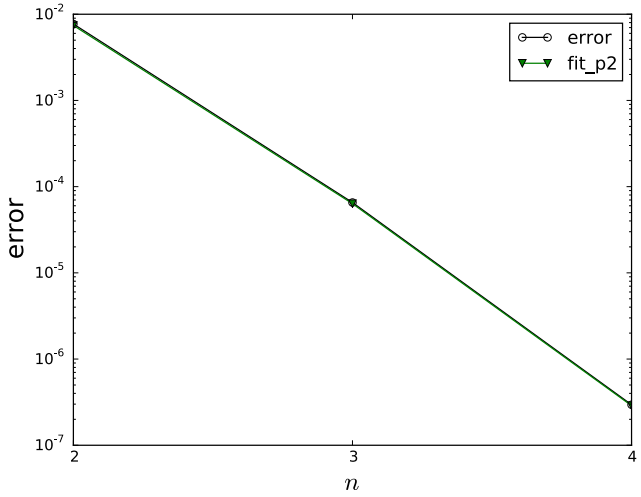
**Example 2:** $\int_{-1}^{1} \exp(x) dx$ for $n = 2,3,4$

The integral converges quickly with N. In the following Table, we list the errors for N=2,3,4, and the builtin function. The error is plotted in Figure that shows that the error varies as

$$E_n = \exp(-0.318N^2 - 3.16N + 2.7)$$

Thus the convergence is exponential (for small N).

| n | I | error |
|---|---|---|
| 2 | 2.342 | 0.00770 |
| 3 | 2.35033 | 6.55E-05 |
| 4 | 2.3504021 | 2.87E-07 |
| from Python Gauss quad | 2.3504023872 | 8.23E-10 |




For a variable *x'* in an arbitrary interval [a,b], we make a change of variable $x' = \alpha x + \beta$, with $(x' = -1, x = a)$ and $(x' = 1, x = b)$.. Hence $a = -\alpha + \beta$, and $b = \alpha + \beta$, leading to $\alpha = (b-a)/2$, $\beta = (b+a)/2$. Hence,

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

## Laguerre-Gauss quadrature

We often encounter integral of the form

$$\int_0^\infty e^{-x} f(x) dx,$$

$w(x) = \exp(-x)$ is the weight function. One of the example is Hydrogen atom. Note that the integration of the above function with Newton-Cote's scheme will be very expensive because we have to take many intervals to reach x = ∞.

we use Laguerre polynomials whose orthogonality relation is

$$\int_0^\infty e^{-x} L_n(x) L_m(x) dx = \delta_{m,n}$$

$$\int_0^\infty e^{-x} f(x) dx = \sum_{j=0}^{N-1} w_j f(x_j),$$

where $x_i$ are the roots of $L_N(x)$. Also,

$\gamma_j = 1$

$$A_j = \frac{(-1)^j}{j!}, \quad a_N = -\frac{1}{N}$$

Hence

$$w_j = -\frac{1}{(N+1)\phi'_N(x_j)\phi_{N+1}(x_j)} = -\frac{1}{N\phi_{N-1}(x_j)\phi'_N(x_j)}$$

Using identities we obtain

$$w_j = \frac{1}{x_j[\phi'_N(x_j)]^2} = \frac{x_j}{(N+1)^2[L_{N+1}(x_j)]^2}$$

[REF: mathworld.wolfram.com]

Laguerre-Gauss quadrature

| $n$ | $x_i$ | $w_i$ |
|---|---|---|
| 2 | 0.585786 | 0.853553 |
| | 3.41421 | 0.146447 |
| 3 | 0.415775 | 0.711093 |
| | 2.29428 | 0.278518 |
| | 6.28995 | 0.0103893 |
| 4 | 0.322548 | 0.603154 |
| | 1.74576 | 0.357419 |
| | 4.53662 | 0.0388879 |
| | 9.39507 | 0.000539295 |
| 5 | 0.26356 | 0.521756 |
| | 1.4134 | 0.398667 |
| | 3.59643 | 0.0759424 |
| | 7.08581 | 0.00361176 |
| | 12.6408 | 0.00002337 |

## Hermit-Gauss quadrature

We also encounter integral of the form

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx = \sum_j w_j f(x_j).$$

Here $w(x) = \exp(-x^2)$ is the weight function. For such integrals we employ Hermite's polynomials. Here $x_i$ are the roots of $H_N(x)$, and wi's are given below. Recall the wavefunction of the harmonic oscillator.

Hermite-Gauss quadrature

| n | $x_i$ | $w_i$ |
|---|---|---|
| 2 | ±0.707107 | 0.886227 |
| 3 | 0 | 1.18164 |
| | ±1.22474 | 0.295409 |
| 4 | ±0.524648 | 0.804914 |
| | ±1.65068 | 0.0813128 |
| 5 | 0 | 0.945309 |
| | ±0.958572 | 0.393619 |
| | ±2.02018 | 0.0199532 |