scipy.integrate.quad

scipy.integrate.quad(*func*, *a*, *b*, *args=()*, *full_output=0*, *epsabs=1.49e-08*, *epsrel=1.49e-08*, *limit=50*, *points=None*, *weight=None*, *wvar=None*, *wopts=None*, *maxp1=50*, *limlst=50***) [source] (https://github.com/scipy/scipy/blob/v1.1.0/scipy/integrate/quadpack.py#L44-L427)**

Compute a definite integral.

Integrate func from *a* to *b* (possibly infinite interval) using a technique from the Fortran library QUADPACK.

Parameters:

func : ***{function, scipy.LowLevelCallable}***

A Python function or method to integrate. If *func* takes many arguments, it is integrated along the axis corresponding to the first argument.

If the user desires improved integration performance, then *f* may be a **scipy.LowLevelCallable** (scipy.LowLevelCallable.html#scipy.LowLevelCallable) with one of the signatures:

```
double func(double x)
double func(double x, void *user_data)
double func(int n, double *xx)
double func(int n, double *xx, void *user_data)
```

The `user_data` is the data contained in the **scipy.LowLevelCallable** (scipy.LowLevelCallable.html#scipy.LowLevelCallable). In the call forms with `xx`, `n` is the length of the `xx` array which contains `xx[0] == x` and the rest of the items are numbers contained in the `args` argument of quad.

In addition, certain ctypes call signatures are supported for backward compatibility, but those should not be used in new code.

a : ***float***

Lower limit of integration (use -numpy.inf for -infinity).

b : ***float***

Upper limit of integration (use numpy.inf for +infinity).

args : ***tuple, optional***

Extra arguments to pass to *func*.

full_output : ***int, optional***

Non-zero to return a dictionary of integration information. If non-zero, warning messages are also suppressed and the message is appended to the output tuple.

Returns:

y : ***float***

The integral of func from *a* to *b*.

abserr : ***float***

An estimate of the absolute error in the result.

infodict : ***dict***

A dictionary containing additional information. Run scipy.integrate.quad_explain() for more information.

message

A convergence message.

explain

Appended only with 'cos' or 'sin' weighting and infinite integration limits, it contains an explanation of the codes in infodict['ierlst']

Other Parameters:

epsabs : *float or int, optional*
: Absolute error tolerance.

epsrel : *float or int, optional*
: Relative error tolerance.

limit : *float or int, optional*
: An upper bound on the number of subintervals used in the adaptive algorithm.

points : *(sequence of floats,ints), optional*
: A sequence of break points in the bounded integration interval where local difficulties of the integrand may occur (e.g., singularities, discontinuities). The sequence does not have to be sorted.

weight : *float or int, optional*
: String indicating weighting function. Full explanation for this and the remaining arguments can be found below.

wvar : *optional*
: Variables for use with weighting functions.

wopts : *optional*
: Optional input for reusing Chebyshev moments.

maxp1 : *float or int, optional*
: An upper bound on the number of Chebyshev moments.

limlst : *int, optional*
: Upper bound on the number of cycles (>=3) for use with a sinusoidal weighting and an infinite end-point.

See also:

dblquad (scipy.integrate.dblquad.html#scipy.integrate.dblquad) double integral
tplquad (scipy.integrate.tplquad.html#scipy.integrate.tplquad) triple integral
nquad (scipy.integrate.nquad.html#scipy.integrate.nquad) n-dimensional integrals (uses **quad** recursively)
fixed_quad (scipy.integrate.fixed_quad.html#scipy.integrate.fixed_quad) fixed-order Gaussian quadrature
quadrature (scipy.integrate.quadrature.html#scipy.integrate.quadrature) adaptive Gaussian quadrature
odeint (scipy.integrate.odeint.html#scipy.integrate.odeint) ODE integrator
ode (scipy.integrate.ode.html#scipy.integrate.ode) ODE integrator
simps (scipy.integrate.simps.html#scipy.integrate.simps) integrator for sampled data
romb (scipy.integrate.romb.html#scipy.integrate.romb) integrator for sampled data
scipy.special (../special.html#module-scipy.special) for coefficients and roots of orthogonal polynomials

Notes

Extra information for quad() inputs and outputs

If full_output is non-zero, then the third output argument (infodict) is a dictionary with entries as tabulated below. For infinite limits, the range is transformed to (0,1) and the optional outputs are given with respect to this transformed range. Let M be the input argument limit and let K be infodict['last']. The entries are:

'neval'
: The number of function evaluations.

'last'
: The number, K, of subintervals produced in the subdivision process.

'alist'
: A rank-1 array of length M, the first K elements of which are the left end points of the subintervals in the partition of the integration range.

'blist'
: A rank-1 array of length M, the first K elements of which are the right end points of the subintervals.

'rlist'
: A rank-1 array of length M, the first K elements of which are the integral approximations on the subintervals.

'elist'
: A rank-1 array of length M, the first K elements of which are the moduli of the absolute error estimates on the subintervals.

'iord'
: A rank-1 integer array of length M, the first L elements of which are pointers to the error estimates over the subintervals with `L=K` if `K<=M/2+2` or `L=M+1-K` otherwise. Let I be the sequence `infodict['iord']` and let E be the sequence `infodict['elist']`. Then `E[I[1]], ..., E[I[L]]` forms a decreasing sequence.

If the input argument points is provided (i.e. it is not None), the following additional outputs are placed in the output dictionary. Assume the points sequence is of length P.

'pts'
: A rank-1 array of length P+2 containing the integration limits and the break points of the intervals in ascending order. This is an array giving the subintervals over which integration will occur.

'level'
: A rank-1 integer array of length M (=limit), containing the subdivision levels of the subintervals, i.e., if (aa,bb) is a subinterval of `(pts[1], pts[2])` where `pts[0]` and `pts[2]` are adjacent elements of `infodict['pts']`, then (aa,bb) has level l if `|bb-aa| = |pts[2]-pts[1]| * 2**(-l)`.

'ndin'
: A rank-1 integer array of length P+2. After the first integration over the intervals (pts[1], pts[2]), the error estimates over some of the intervals may have been increased artificially in order to put their subdivision forward. This array has ones in slots corresponding to the subintervals for which this happens.

Weighting the integrand

The input variables, *weight* and *wvar*, are used to weight the integrand by a select list of functions. Different integration methods are used to compute the integral with these weighting functions. The possible values of weight and the corresponding weighting functions are.

`weight`	Weight function used	`wvar`
'cos'	cos(w*x)	wvar = w
'sin'	sin(w*x)	wvar = w
'alg'	g(x) = ((x-a)**alpha)*((b-x)**beta)	wvar = (alpha, beta)

`weight`	Weight function used	`wvar`
'alg-loga'	g(x)*log(x-a)	wvar = (alpha, beta)
'alg-logb'	g(x)*log(b-x)	wvar = (alpha, beta)
'alg-log'	g(x)*log(x-a)*log(b-x)	wvar = (alpha, beta)
'cauchy'	1/(x-c)	wvar = c

wvar holds the parameter w, (alpha, beta), or c depending on the weight selected. In these expressions, a and b are the integration limits.

For the 'cos' and 'sin' weighting, additional inputs and outputs are available.

For finite integration limits, the integration is performed using a Clenshaw-Curtis method which uses Chebyshev moments. For repeated calculations, these moments are saved in the output dictionary:

'momcom'

The maximum level of Chebyshev moments that have been computed, i.e., if `M_c` is `infodict['momcom']` then the moments have been computed for intervals of length `|b-a| * 2**(-l)`, `l=0,1,...,M_c`.

'nnlog'

A rank-1 integer array of length M(=limit), containing the subdivision levels of the subintervals, i.e., an element of this array is equal to l if the corresponding subinterval is `|b-a|* 2**(-l)`.

'chebmo'

A rank-2 array of shape (25, maxp1) containing the computed Chebyshev moments. These can be passed on to an integration over the same interval by passing this array as the second element of the sequence wopts and passing infodict['momcom'] as the first element.

If one of the integration limits is infinite, then a Fourier integral is computed (assuming w neq 0). If full_output is 1 and a numerical error is encountered, besides the error message attached to the output tuple, a dictionary is also appended to the output tuple which translates the error codes in the array `info['ierlst']` to English messages. The output information dictionary contains the following entries instead of 'last', 'alist', 'blist', 'rlist', and 'elist':

'lst'

The number of subintervals needed for the integration (call it `K_f`).

'rslst'

A rank-1 array of length M_f=limlst, whose first `K_f` elements contain the integral contribution over the interval `(a+(k-1)c, a+kc)` where `c = (2*floor(|w|) + 1) * pi / |w|` and `k=1,2,...,K_f`.

'erlst'

A rank-1 array of length `M_f` containing the error estimate corresponding to the interval in the same position in `infodict['rslist']`.

'ierlst'

A rank-1 integer array of length `M_f` containing an error flag corresponding to the interval in the same position in `infodict['rslist']`. See the explanation dictionary (last entry in the output tuple) for the meaning of the codes.

Examples

Calculate $\int_0^4 x^2 dx$ and compare with an analytic result