acsefunctions

Release 0.1

Your Name

Apr 19, 2025

CONTENTS

acsefunctions.transcendental.**cosh**(*x*, *n_terms=20*)

Compute the hyperbolic cosine function cosh(x) using a Taylor series approximation.

Parameters

- **x** (*float or numpy.ndarray*) – Input value(s) for which to compute cosh(x).
- **n_terms** (*int, optional*) – Number of terms to use in the Taylor series (default is 20).

Returns

Computed cosh(x) for the input(s).

Return type

float or numpy.ndarray

Examples

```
>>> cosh(0)
1.0
>>> cosh(1)
1.5430806348152437
>>> cosh(np.array([0, 1, 2]))
array([1.        , 1.54308063, 3.76219569])
```

acsefunctions.transcendental.**exp**(*x*, *n_terms=20*)

Compute the exponential function e^x using a Taylor series approximation.

Parameters

- **x** (*float or numpy.ndarray*) – Input value(s) for which to compute e^x.
- **n_terms** (*int, optional*) – Number of terms to use in the Taylor series (default is 20).

Returns

Computed e^x for the input(s).

Return type

float or numpy.ndarray

Examples

```
>>> exp(0)
1.0
>>> exp(1)
2.718281828459045
>>> exp(np.array([0, 1, 2]))
array([1.        , 2.71828183, 7.3890561 ])
```

acsefunctions.transcendental.**sinh**(*x*, *n_terms=20*)

Compute the hyperbolic sine function sinh(x) using a Taylor series approximation.

Parameters

- **x** (*float or numpy.ndarray*) – Input value(s) for which to compute sinh(x).
- **n_terms** (*int, optional*) – Number of terms to use in the Taylor series (default is 20).

Returns

Computed sinh(x) for the input(s).

Return type
float or numpy.ndarray

Examples

```
>>> sinh(0)
0.0
>>> sinh(1)
1.1752011936438014
>>> sinh(np.array([0, 1, 2]))
array([0.        , 1.17520119, 3.62686041])
```

`acsefunctions.transcendental.tanh`(*x*, *n_terms=20*)

Compute the hyperbolic tangent function tanh(x) as sinh(x) / cosh(x).

Parameters

- **x** (*float or numpy.ndarray*) – Input value(s) for which to compute tanh(x).
- **n_terms** (*int, optional*) – Number of terms to use in the Taylor series for sinh and cosh (default is 20).

Returns
Computed tanh(x) for the input(s).

Return type
float or numpy.ndarray

Raises
ZeroDivisionError – If cosh(x) equals zero, which can occur in rare numerical edge cases.

Examples

```
>>> tanh(0)
0.0
>>> tanh(1)
0.7615941559557649
>>> tanh(np.array([0, 1, 2]))
array([0.        , 0.76159416, 0.96402758])
```

Special functions: factorial, gamma, and Bessel.

`acsefunctions.special.bessel`(*alpha*, *x*, *n_terms=20*)

Compute the Bessel function J_alpha(x) using its series expansion.

Parameters

- **alpha** (*float*) – Order of the Bessel function.
- **x** (*float or numpy.ndarray*) – Input value(s).
- **n_terms** (*int, optional*) – Number of terms in the series (default is 20).

Returns
Computed J_alpha(x).

Return type
float or numpy.ndarray

Examples

```
>>> bessel(0, 0)
1.0
>>> bessel(0, 1)  # Approximate value
0.7651976865579666
>>> bessel(0, np.array([0, 1]))
array([1.        , 0.76519769])
```

`acsefunctions.special.`**`factorial`**`(`*n*`)`

Compute the factorial n! for non-negative integers.

Parameters
n (*int or numpy.ndarray*) – Non-negative integer input(s).

Returns
Computed n!.

Return type
int or numpy.ndarray

Raises
ValueError – If n is negative.

Examples

```
>>> factorial(0)
1
>>> factorial(5)
120
>>> factorial(np.array([0, 1, 2]))
array([1, 1, 2])
```

`acsefunctions.special.`**`gamma`**`(`*z*, *T=100*, *M=1000*`)`

Compute the gamma function gamma(z) for z > 0 using numerical integration.

Uses trapezoidal rule on gamma(z) = _0^∞ t^(z-1) e^(-t) dt.

Parameters

- **z** (*float or numpy.ndarray*) – Input value(s), must be positive.
- **T** (*float, optional*) – Upper integration limit (default is 100).
- **M** (*int, optional*) – Number of integration points (default is 1000).

Returns
Computed gamma(z).

Return type
float or numpy.ndarray

Raises
ValueError – If z <= 0.

Examples

```
>>> gamma(1)
1.0
>>> gamma(0.5)  # Approximately sqrt(pi)
1.7724538209055159
>>> gamma(np.array([1, 2]))
array([1., 1.])
```

PYTHON MODULE INDEX

a

INDEX