# acsefunctions

*Release 0.10*

Paulina Boadiwaa Mensah

Oct 18, 2024

# CONTENTS:

CHAPTER

ONE

# ACSEFUNCTIONS

This package consists of 2 modules: taylors_series and bessel_group used for trigonometric and Bessel functions respectively, for use on `scalar and numpy array` objects. The trigonometric functions were created using Taylors series.[1] The Bessel function was implemented based on Euler's definition.[2]

For examples see *`acsefunctions.taylors_series.trig_functions.exp()`* and *`acsefunctions.taylors_series.trig_functions.sinh()`* for more information.

## 1.1 Functions in the taylors_series module:

`acsefunctions.taylors_series.trig_functions.`**`cosh`**`(`*x*, *terms=20*`)`

Compute cosh(x) using Taylor series expansion.

**Parameters**

- **`np.array`** (*`int or float or list of integers or`*) – x: The input value for cosh(x). Can be a scalar or a numpy array.
- **`int`** –

  **terms: Number of terms in the Taylor series to consider**
  (higher = more accurate).

**Returns**

Approximate value of cosh(x). Can be a scalar or a numpy array.

**Return type**

np.float or np.array

### Examples

```
>>> cosh(1)
array(1.54308063)
```

```
>>> cosh([1.0, 2.0, 3.0, 4.0, 5.0])
array([ 1.54308063,  3.76219569, 10.067662  , 27.30823284, 74.20994852])
```

`acsefunctions.taylors_series.trig_functions.`**`exp`**`(`*x*, *terms=20*`)`

Compute e^x using Taylor series expansion.

[1] https://mathworld.wolfram.com/TaylorSeries.html

[2] https://mathworld.wolfram.com/BesselFunctionoftheFirstKind.html

**Parameters**

**np.array** (*int or float or list of integers or*) – x: The exponent value for e^x. Can be a scalar or a numpy array. terms: Number of terms in the Taylor series to consider

(higher = more accurate).

**Returns**

Approximate value of e^x. Can be a scalar or a numpy array.

**Return type**

np.float or np.array

### Examples

```
>>> exp(2.0)
array(7.3890561)
```

```
>>> exp([1.0, 2.0, 3.0])
array([ 2.71828183,  7.3890561 , 20.08553692])
```

acsefunctions.taylors_series.trig_functions.**sinh**(*x*, *terms=20*)

Compute sinh(x) using Taylor series expansion.

**Parameters**

- **np.array** (*int or float or list of integers or*) – x: The input value for sinh(x). Can be a scalar or a numpy array.
- **int** –

  **terms: Number of terms in the Taylor series to consider**
  (higher = more accurate).

**Returns**

Approximate value of sinh(x). Can be a scalar or a numpy array.

**Return type**

np.float or np.array

### Examples

```
>>> sinh(5,terms = 30)
array(74.20321058)
```

```
>>> sinh([1.0, 2.0, 3.0, 4.0, 5.0])
array([ 1.17520119,  3.62686041, 10.01787493, 27.2899172 , 74.20321058])
```

acsefunctions.taylors_series.trig_functions.**tanh**(*x*, *terms=20*)

Compute tanh(x) using Taylor series expansion by dividing sinh(x) by cosh(x).

**Parameters**

- **np.array** (*int or float or list of integers or*) – x: The input value for tanh(x). Can be a scalar or a numpy array.
- **int** –

  **terms: Number of terms in the Taylor series for sinh(x) and cosh(x)**
  to consider (higher = more accurate).

**Returns**
Approximate value of tanh(x). Can be a scalar or a numpy array.

**Return type**
np.float or np.array

### Examples

```
>>> tanh(1)
np.float64(0.7615941559557649)
```

```
>>> tanh([1.0, 2.0, 3.0, 4.0, 5.0])
array([0.76159416, 0.96402758, 0.99505475, 0.9993293 , 0.9999092 ])
```

## 1.2 Functions in the bessel_group module:

`acsefunctions.bessel_group.bessel_function.`**`bessel`**(*alpha*, *x*, *terms=50*)

Compute the bessel function for a scalar or numpy array.

**Parameters**

- **`np.array`** (*`int or float or list of integers or`*) – alpha: Order of the Bessel function.
- **`np.array`** – x: Input value(s) at which to evaluate the Bessel function.

**Returns**
Approximation of the Bessel function.

**Return type**
np.float or np.array

### Example

```
>>> bessel(1,[0, 1, 2, 3, 4, 5])
array([ 0.        ,  0.4404682 ,  0.57755987,  0.34031109, -0.06437481,
       -0.32549533])
```

`acsefunctions.bessel_group.bessel_function.`**`factorial`**(*n*)

Compute factorial for a scalar or numpy array. :param int or float or list of integers or np.array: n: Input value(s) for which to compute the factorial.

**Returns**
Factorial of the input(s).

**Return type**
np.float or np.array

### Example

```
>>> factorial([0, 1, 2, 3, 4, 5])
array([  1.,   1.,   2.,   6.,  24., 120.])
```

`acsefunctions.bessel_group.bessel_function.`**`gamma_function`**(*z*)

Compute Gamma function for a scalar or numpy array.

Parameters

**Returns**

Gamma function value of the input(s).

**Return type**

np.float or np.array

### Example

```
>>> gamma_function([1, 1.5, 2, 2.5, 3, 4])
array([1.05088491, 0.87972523, 0.99916542, 1.32925696, 1.99999916,
       6.00000083])
```

# PYTHON MODULE INDEX

# INDEX