phuzzy Documentation

Release 0.7.4

Lepy

Dec 10, 2020

Contents:

CHAPTER 1

phuzzy

Fuzzy calculation in Python.

- Free software: MIT license
- Documentation: https://phuzzy.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install phuzzy, run this command in your terminal:

```
$ pip install phuzzy
```

This is the preferred method to install phuzzy, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for phuzzy can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/lepy/phuzzy
```

Or download the tarball:

```
$ curl  -OL https://github.com/lepy/phuzzy/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use phuzzy in a project:

```
import phuzzy
tn = phuzzy.TruncNorm(alpha0=[2, 3], alpha1=[], number_of_alpha_levels=15, name="t")
tri = phuzzy.Triangle(alpha0=[1, 4], alpha1=[2], number_of_alpha_levels=5)
f = tn + tri
print(f.df)
```

3.1 available shapes

3.1.1 Uniform

```
import phuzzy.mpl as phm
uni = phm.Uniform(alpha0=[1, 4], number_of_alpha_levels=5, name="x")
uni.plot(show=True, filepath="/tmp/uniform.png", title=True)
```

3.1.2 Triangle

```
import phuzzy.mpl as phm

tri = phm.Triangle(alpha0=[1, 4], alpha1=[2], number_of_alpha_levels=5)
tri.plot(show=False, filepath="/tmp/triangle.png", title=True)
```

3.1.3 Trapezoid

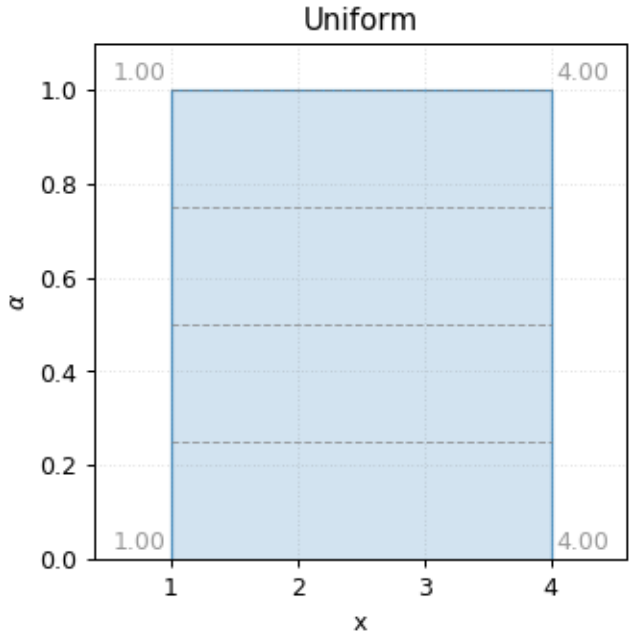

Fig. 1: Uniform fuzzy number (this is just an interval)

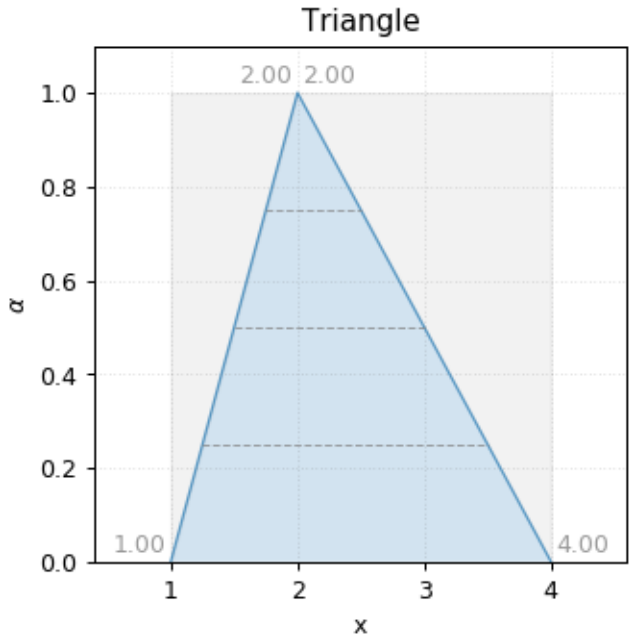

Fig. 2: Triangle fuzzy number

```
import phuzzy.mpl as phm
trap = phm.Trapezoid(alpha0=[1, 5], alpha1=[2, 3], number_of_alpha_levels=5)
trap.plot(show=False, filepath="/tmp/trapezoid.png", title=True)
```

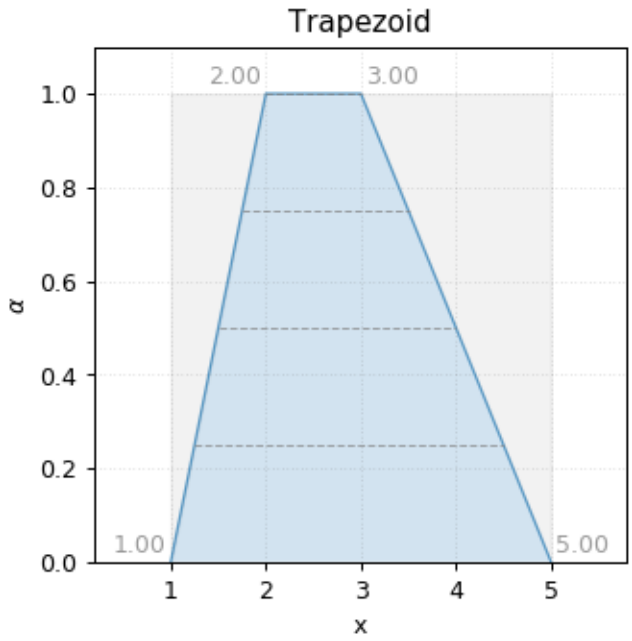

Fig. 3: Trapezoid fuzzy number

3.1.4 TruncNorm

```
import phuzzy.mpl as phm
tn = phm.TruncNorm(alpha0=[1, 3], number_of_alpha_levels=15, name="x")
tn.plot(show=False, filepath="/tmp/truncnorm.png", title=True)
```

3.1.5 TruncGenNorm

```
import phuzzy.mpl as phm
tgn = phm.TruncGenNorm(alpha0=[1, 4], alpha1=[2, 3], number_of_alpha_levels=15,␣
↪beta=3.)
tgn.plot(show=False, filepath="/tmp/truncgennorm.png", title=True)
```

3.1.6 Superellipse

```
import phuzzy.mpl as phm
se = phm.Superellipse(alpha0=[-1, 2.], alpha1=None, m=1.0, n=.5, number_of_alpha_
↪levels=17)
se.plot(show=True, filepath="/tmp/superellipse.png", title=True)
```

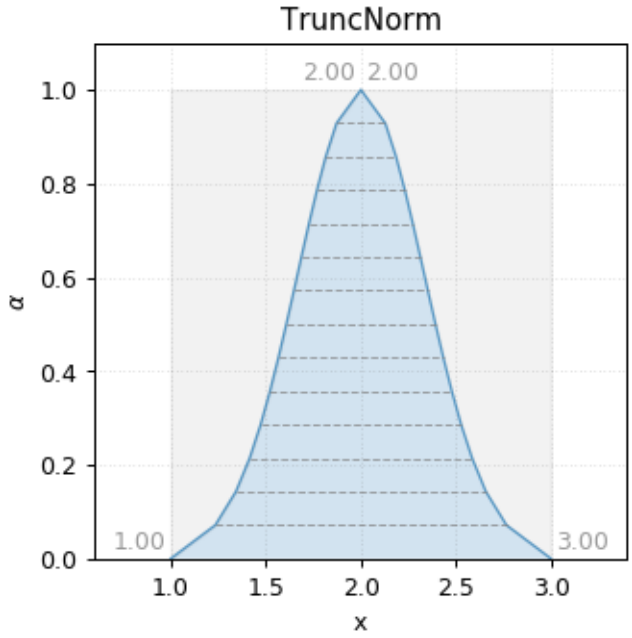

Fig. 4: TruncNorm fuzzy number

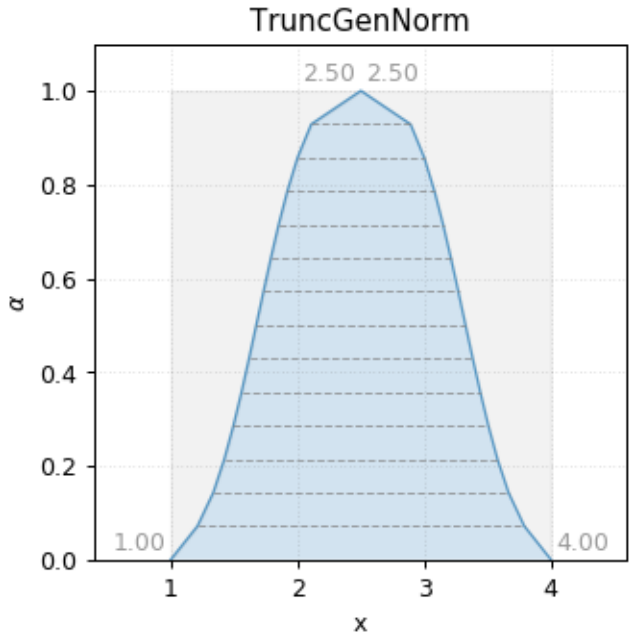

Fig. 5: TruncGenNorm fuzzy number

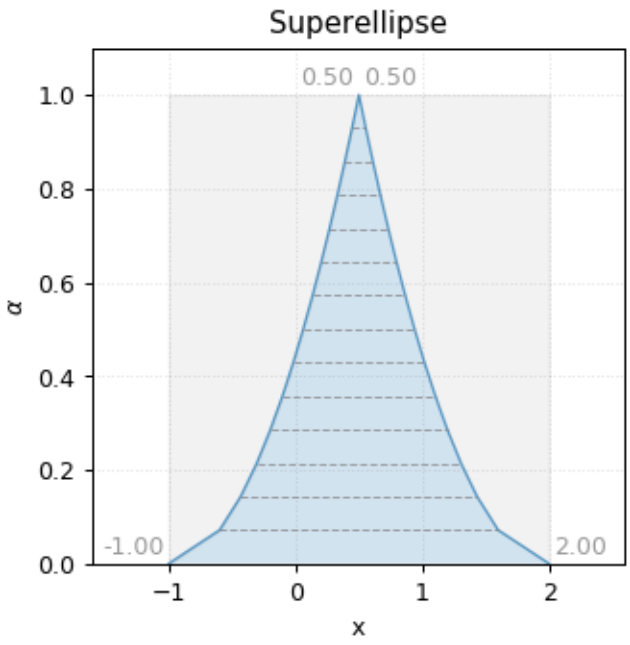

Fig. 6: Superellipse fuzzy number

3.2 basic operations

3.2.1 Addition

$$z = x + y$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
y = phuzzy.TruncNorm(alpha0=[1, 3], number_of_alpha_levels=15, name="y")
z = x + y
z.name = "x+y"
```

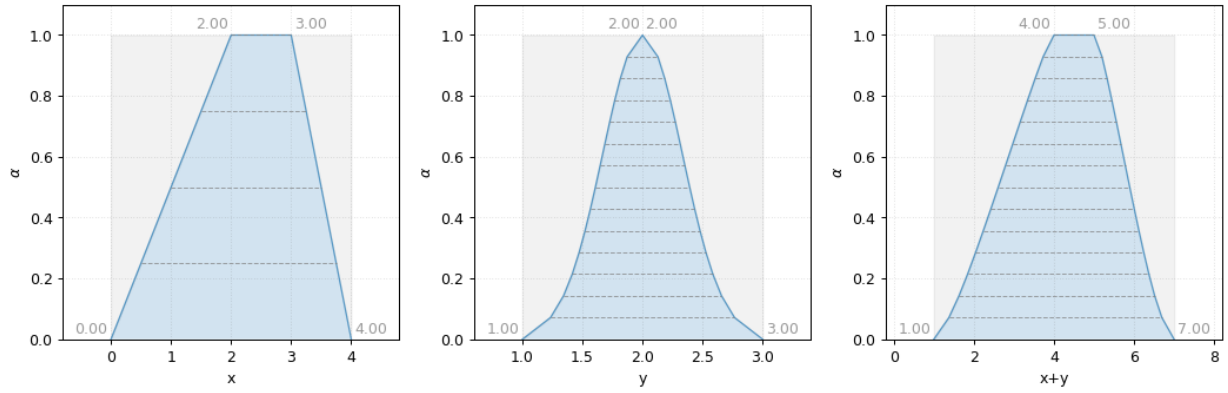

Fig. 7: Addition of fuzzy numbers

$$z = 3 + x$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
z = 3 + x
z = x + 3
```

3.2.2 Substraction

$$z = x - y$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
y = phuzzy.TruncNorm(alpha0=[1, 3], number_of_alpha_levels=15, name="y")
z = x - y
z.name = "x-y"
```

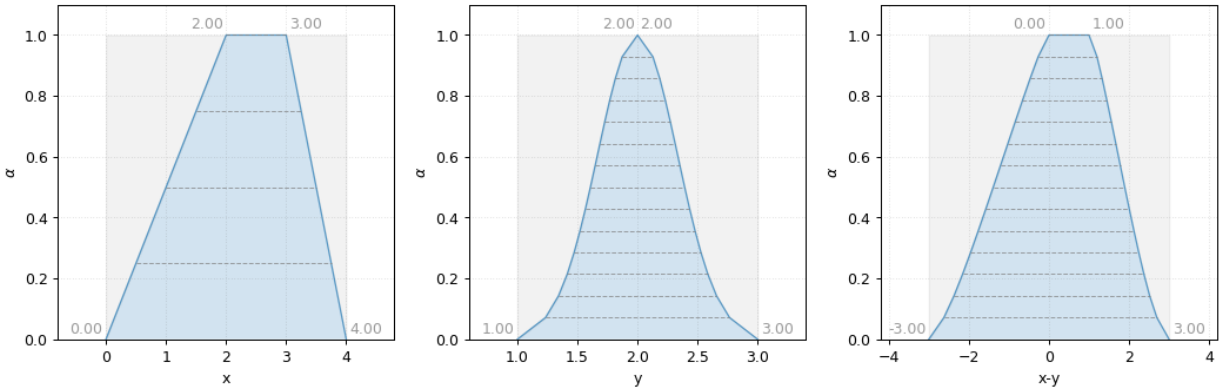

Fig. 8: Substraction of fuzzy numbers

$$y = 3 - x$$
$$z = x - 3$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
y = 3 - x
z = x - 3
```

3.2.3 Multiplication

$$z = xy$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
y = phuzzy.TruncNorm(alpha0=[1, 3], number_of_alpha_levels=15, name="y")
z = x * y
z.name = "x*y"
```

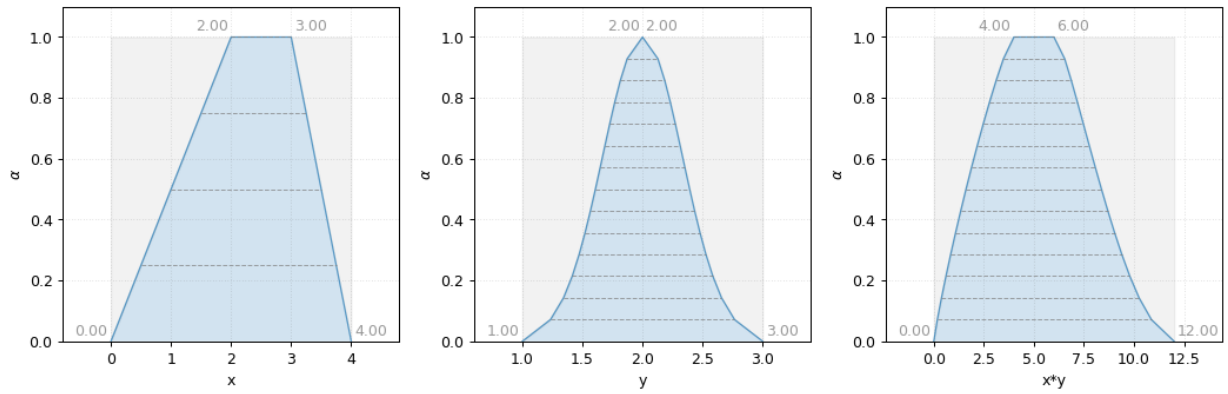

Fig. 9: Multiplication of fuzzy numbers

$$z = 3x$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
z = 3 * x
z = x * 3
```

3.2.4 Division

$$z = \frac{x}{y}$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
y = phuzzy.TruncNorm(alpha0=[1, 3], number_of_alpha_levels=15, name="y")
z = x / y
z.name = "x/y"
```

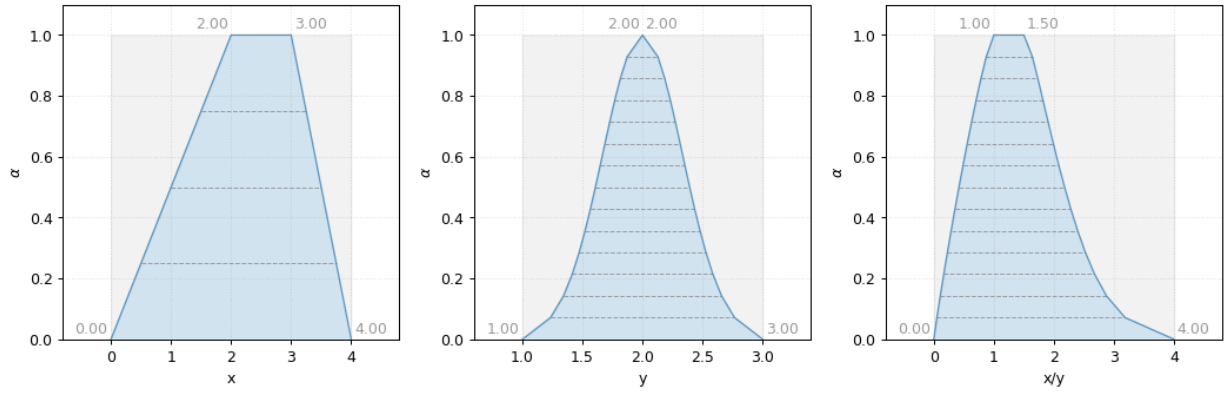

Fig. 10: Division of fuzzy numbers

$$y = 3/x$$
$$z = x/3$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
z = 3 / x
z = x / 3
```

3.2.5 Exponentiation

$$z = x^y$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
y = phuzzy.TruncNorm(alpha0=[1, 3], number_of_alpha_levels=15, name="y")
z = x ** y
z.name = "x^y"
```

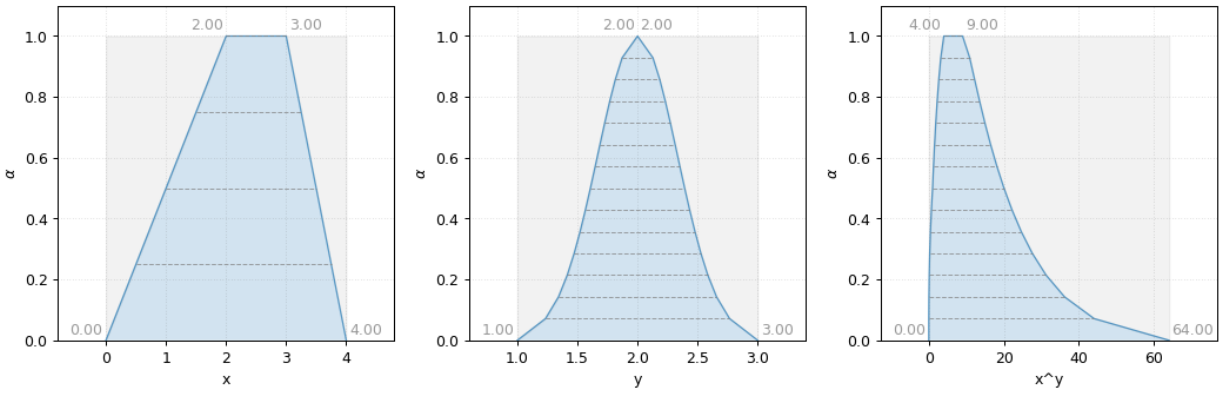

Fig. 11: Power operation with fuzzy numbers

$$z = x^3$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
z = x**3
```

3.2.6 Negation

$$z = -x$$

```
x = phuzzy.Trapezoid(alpha0=[0, 4], alpha1=[2, 3], number_of_alpha_levels=5)
z = -x
```

3.2.7 Absolute value

$$z = |x|$$

```
x = phuzzy.Trapezoid(alpha0=[-1, 4], alpha1=[2, 3], number_of_alpha_levels=5)
z = abs(x)
z = x.abs()
```

CHAPTER 4

Shapes

4.1 Uniform

```
import phuzzy.mpl as phm
uni = phm.Uniform(alpha0=[1, 4], number_of_alpha_levels=5, name="x")
uni.plot(show=True, filepath="/tmp/uniform.png", title=True)
```

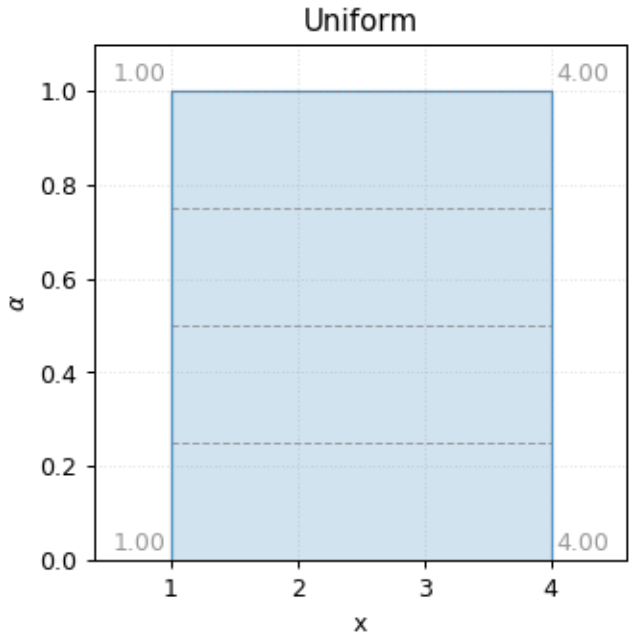

Fig. 1: Uniform fuzzy number

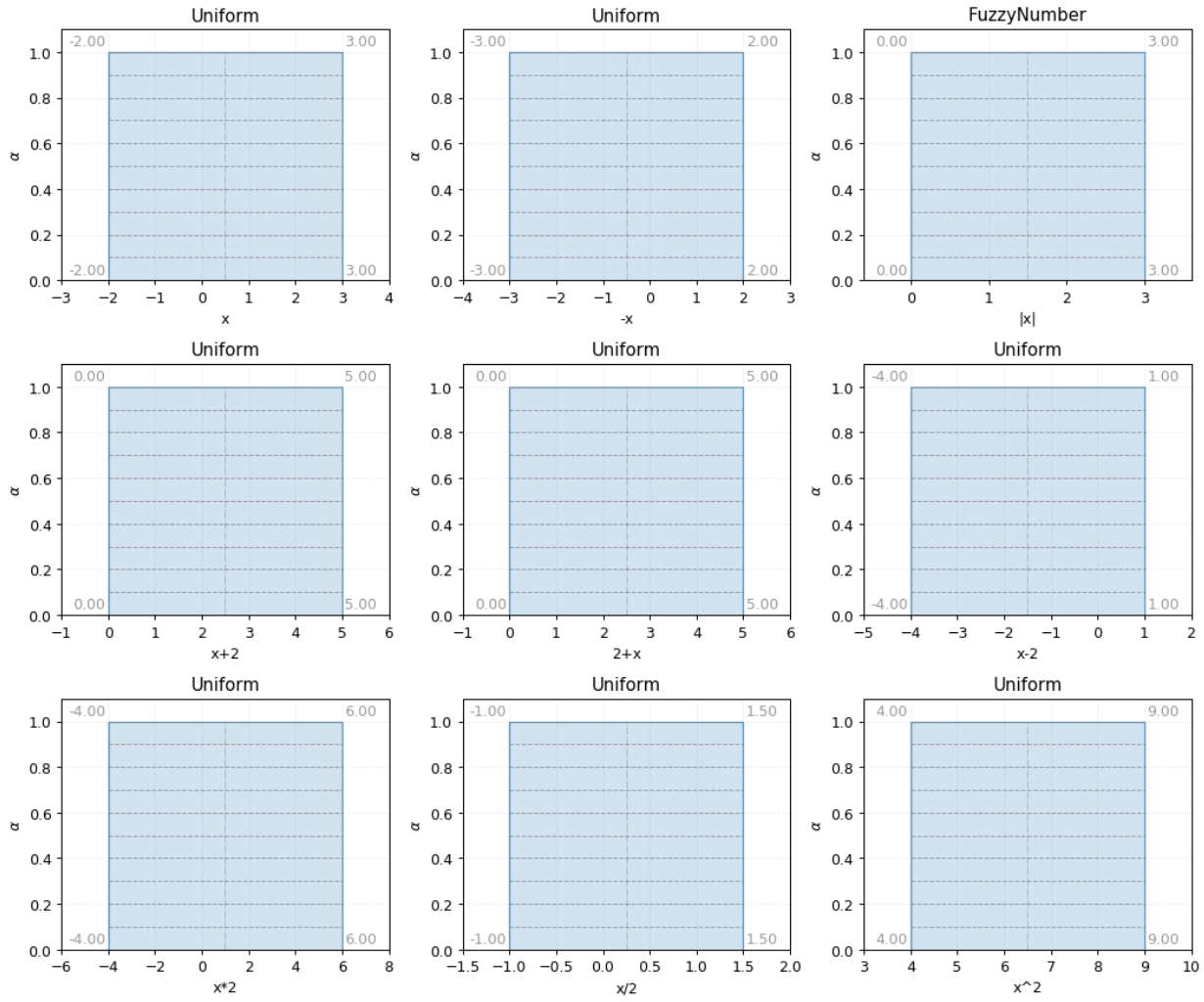

Fig. 2: Uniform fuzzy number operations

4.2 Triangle

```
import phuzzy.mpl as phm

tri = phm.Triangle(alpha0=[1, 4], alpha1=[2], number_of_alpha_levels=5)
tri.plot(show=False, filepath="/tmp/triangle.png", title=True)
```

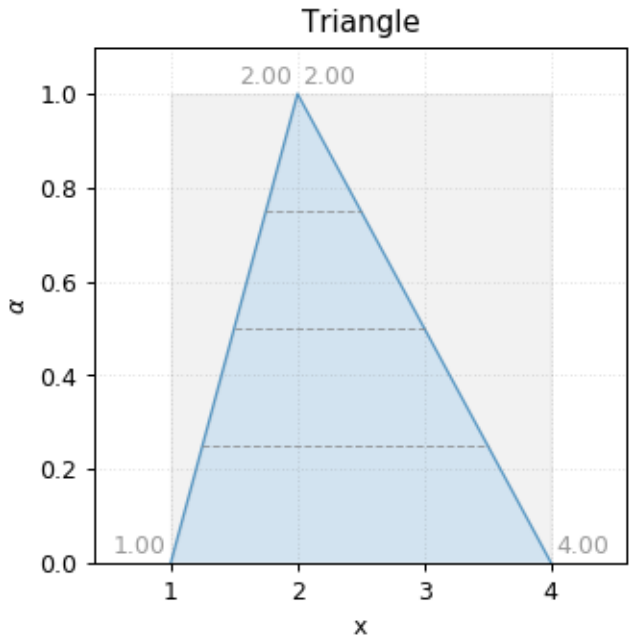

Fig. 3: Triangle fuzzy number

4.3 Trapezoid

```
import phuzzy.mpl as phm
trap = phm.Trapezoid(alpha0=[1, 5], alpha1=[2, 3], number_of_alpha_levels=5)
trap.plot(show=False, filepath="/tmp/trapezoid.png", title=True)
```

4.4 TruncNorm

```
import phuzzy.mpl as phm
tn = phm.TruncNorm(alpha0=[1, 3], number_of_alpha_levels=15, name="x")
tn.plot(show=False, filepath="/tmp/truncnorm.png", title=True)
```

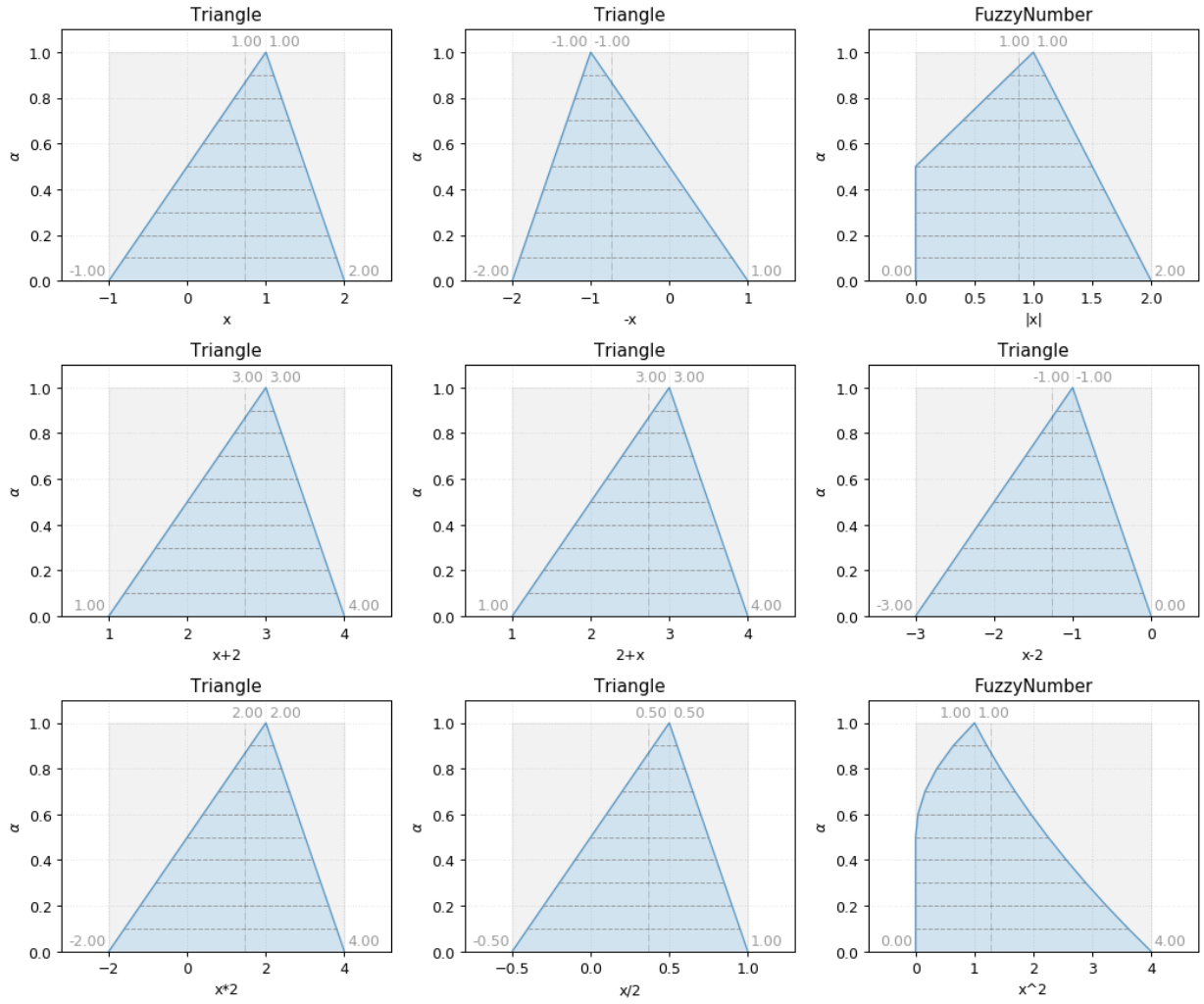

Fig. 4: Triangle fuzzy number operations

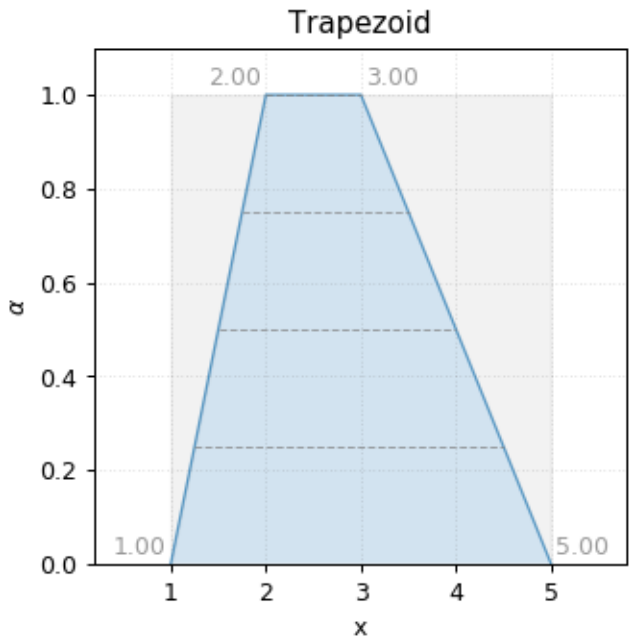

Fig. 5: Trapezoid fuzzy number

4.5 TruncGenNorm

```
import phuzzy.mpl as phm
tgn = phm.TruncGenNorm(alpha0=[1, 4], alpha1=[2, 3], number_of_alpha_levels=15,␣
↪beta=3.)
tgn.plot(show=False, filepath="/tmp/truncgennorm.png", title=True)
```

4.6 Superellipse

```
import phuzzy.mpl as phm
se = phm.Superellipse(alpha0=[-1, 2.], alpha1=None, m=1.0, n=.5, number_of_alpha_
↪levels=17)
se.plot(show=True, filepath="/tmp/superellipse.png", title=True)
```

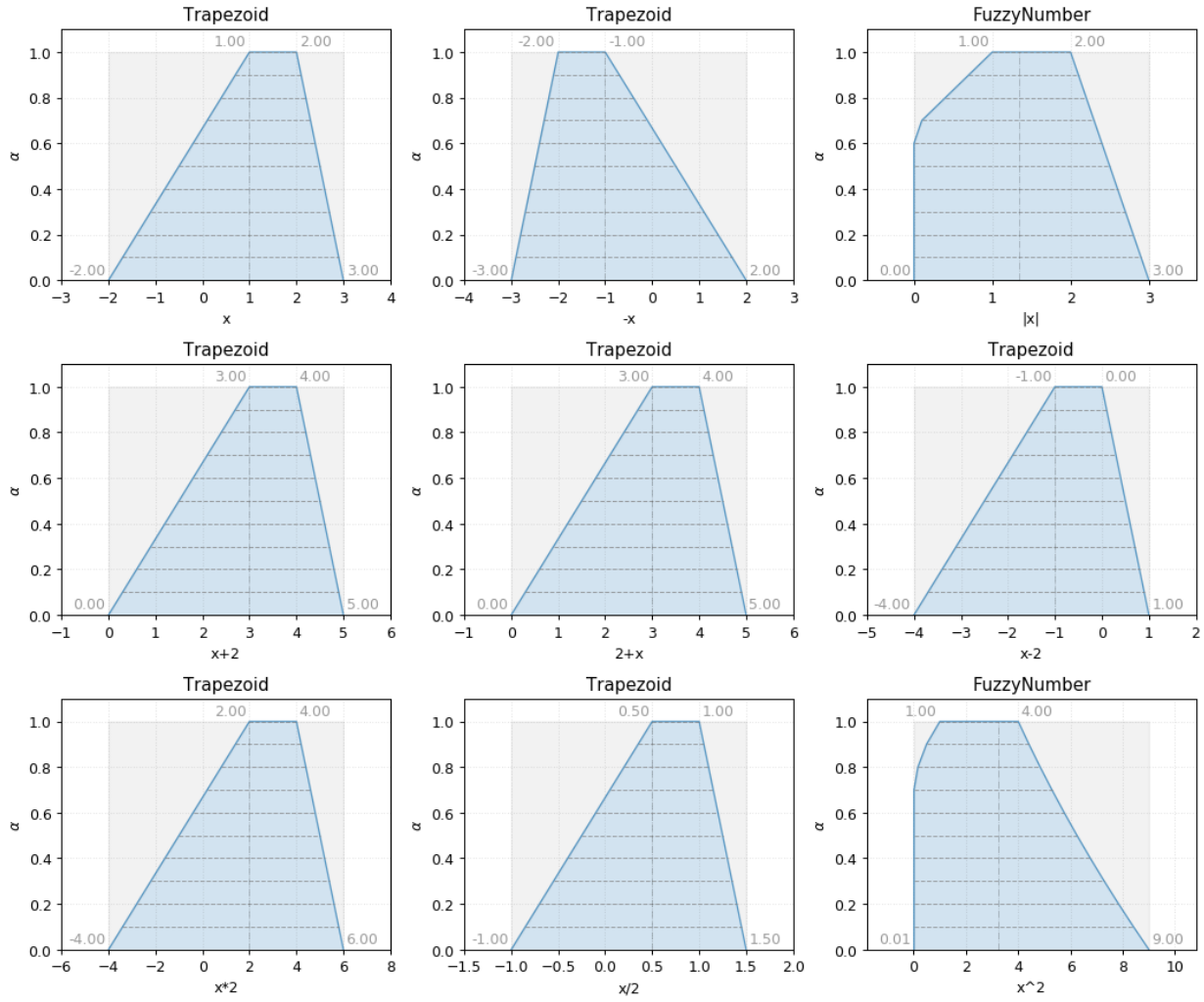

Fig. 6: Trapezoid fuzzy number operations

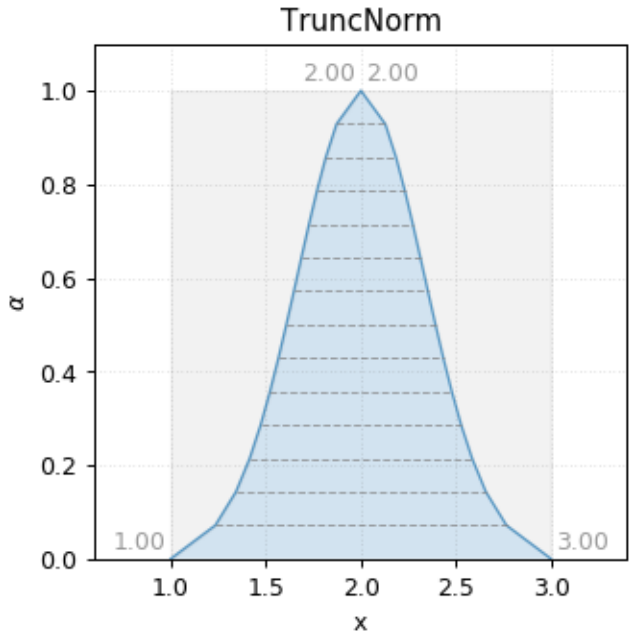

Fig. 7: TruncNorm fuzzy number

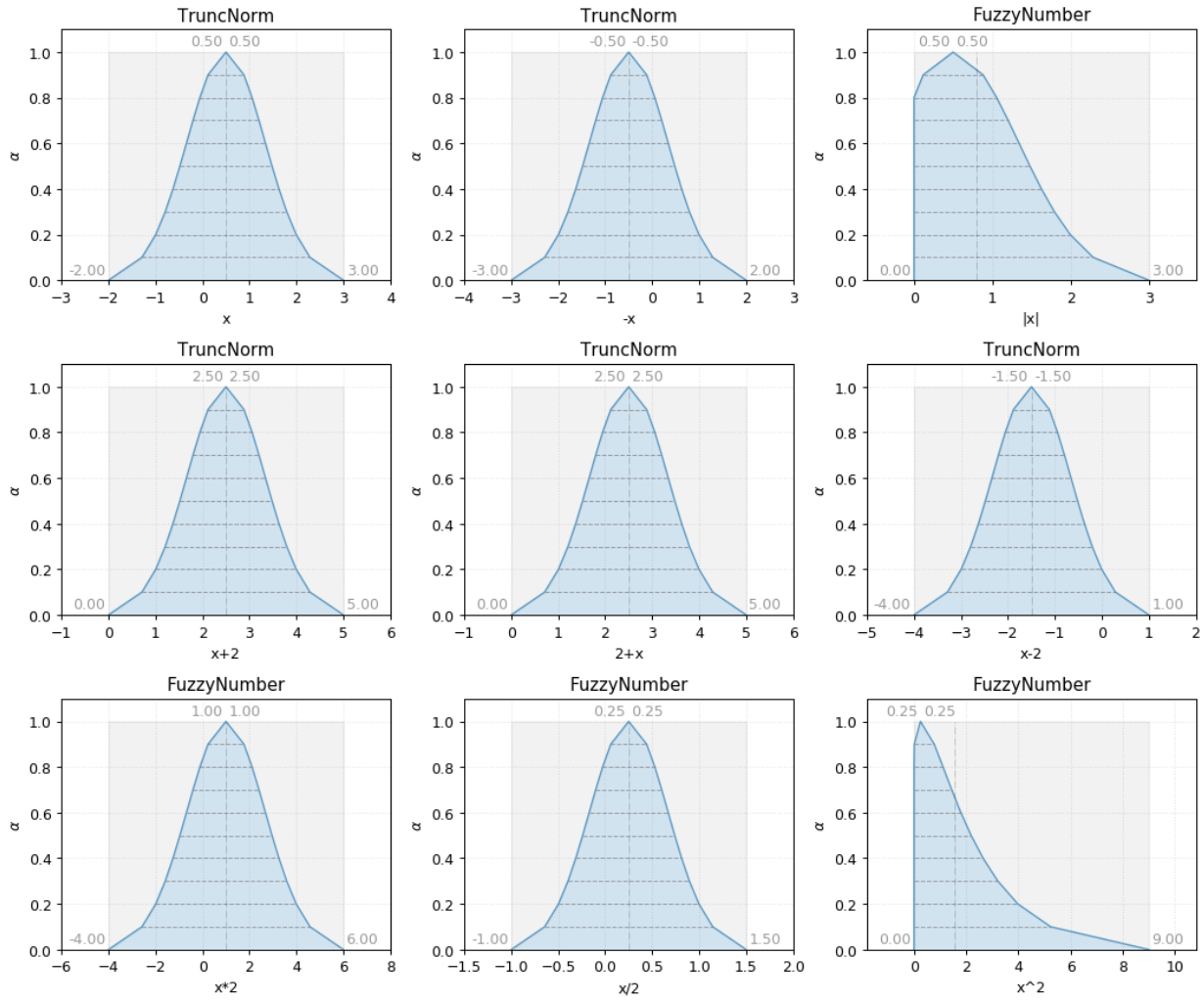

Fig. 8: TruncNorm fuzzy number operations

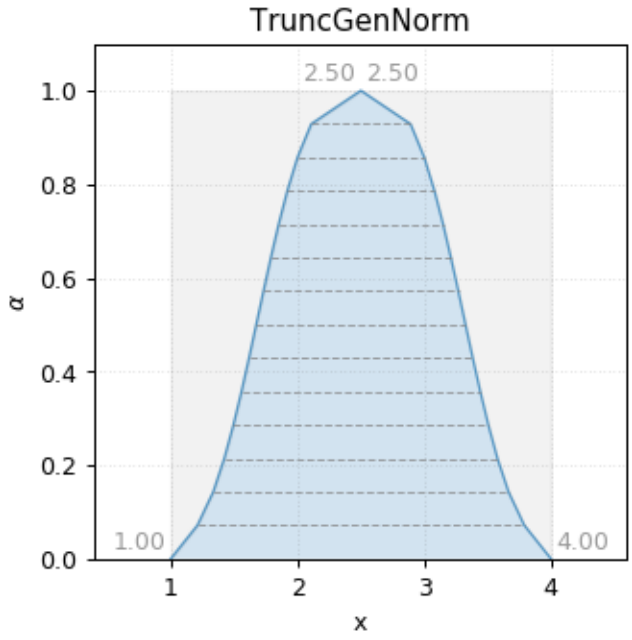

Fig. 9: TruncGenNorm fuzzy number

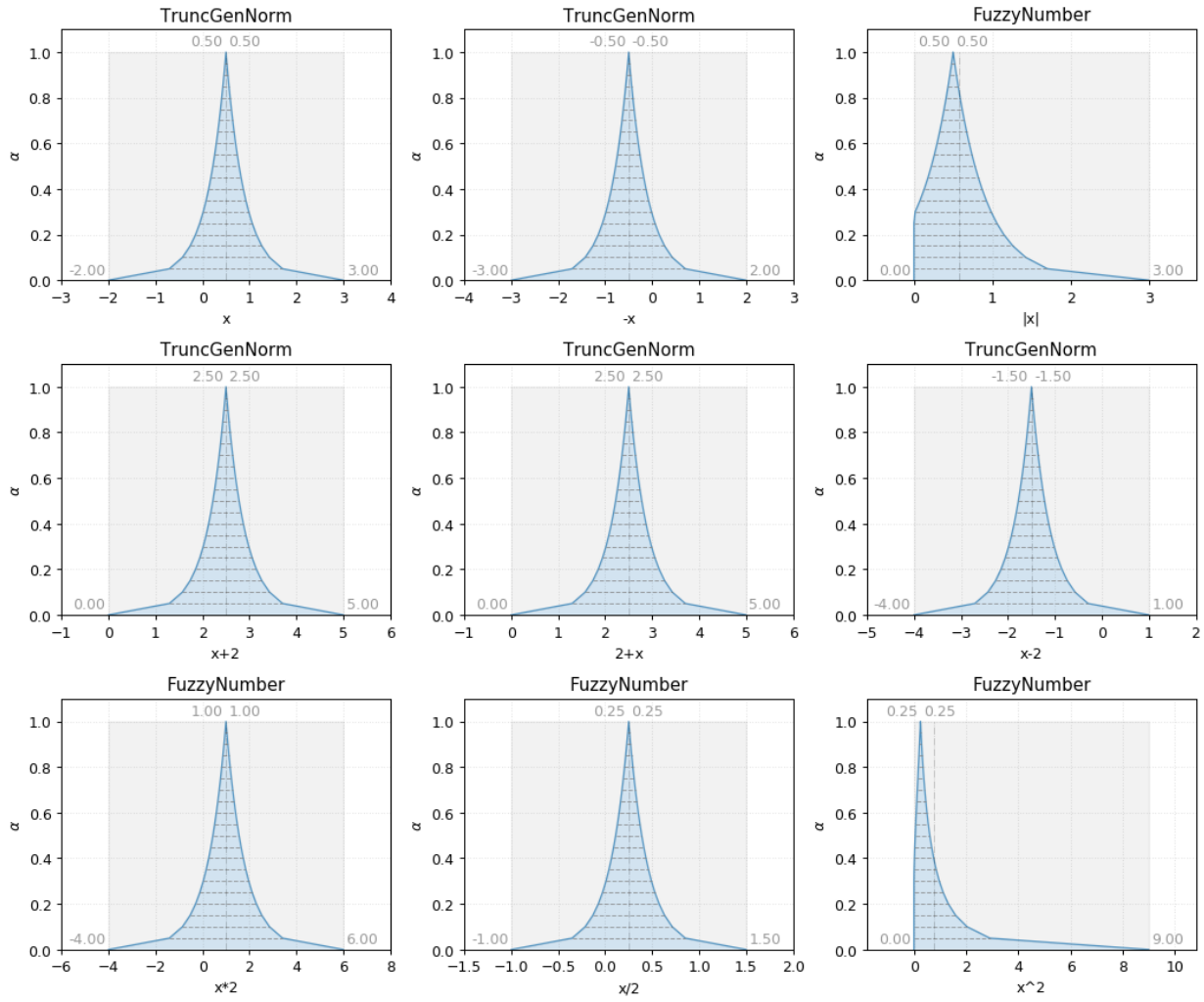

Fig. 10: TruncGenNorm fuzzy number operations

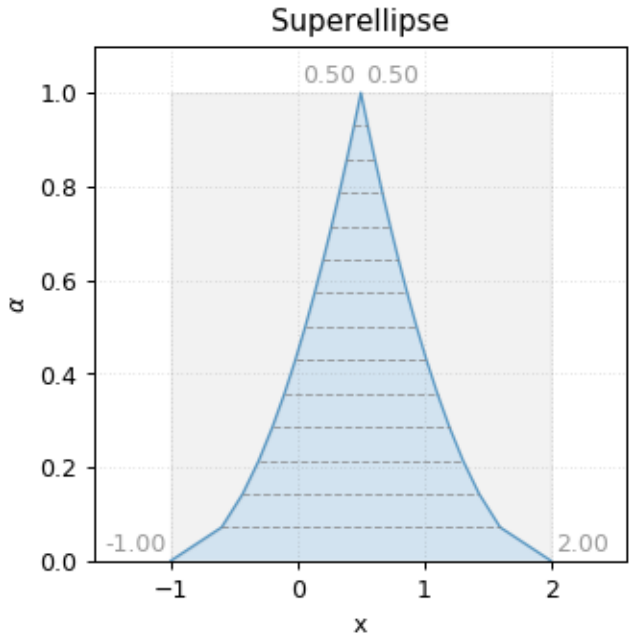

Fig. 11: Superellipse fuzzy number

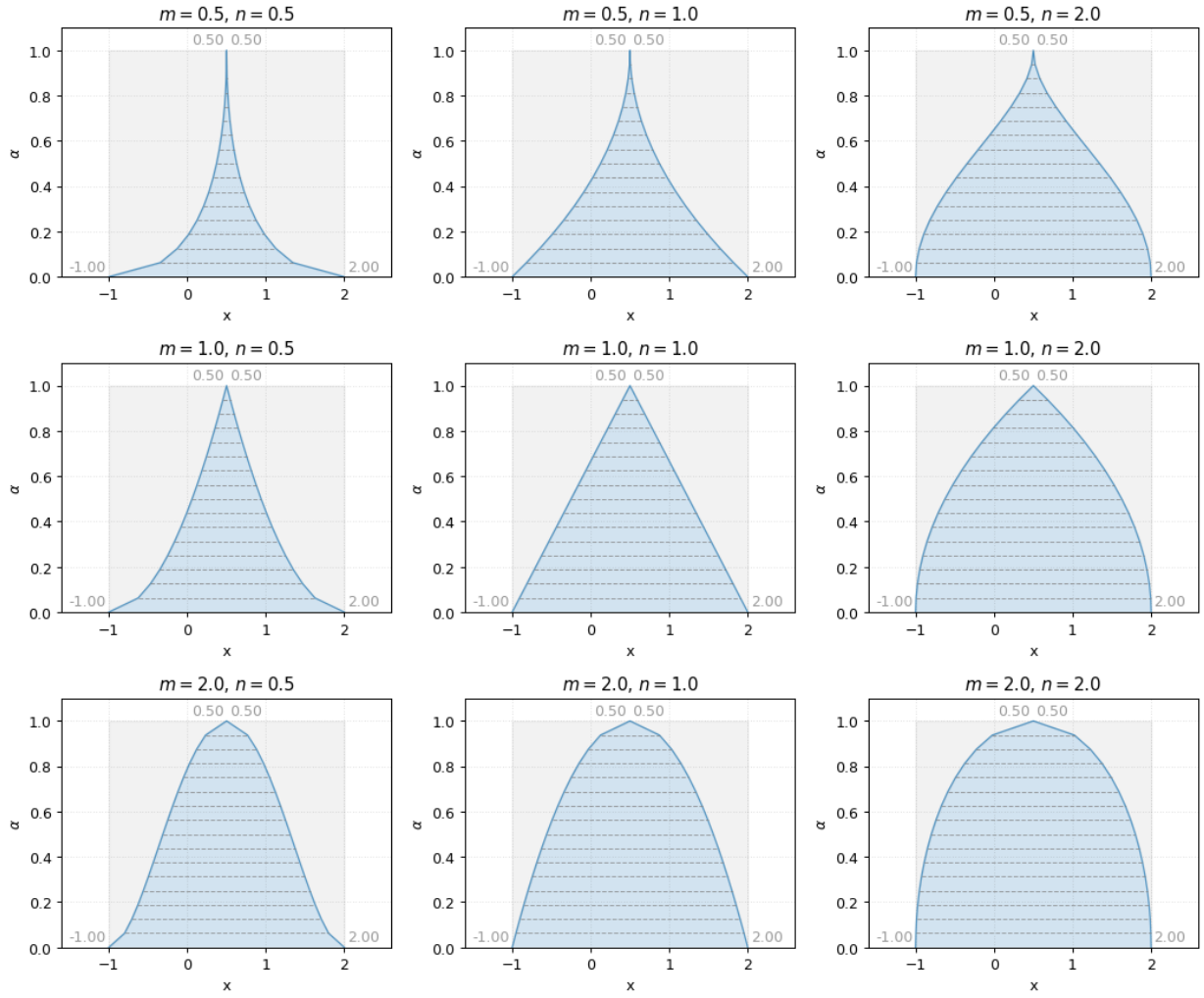

Fig. 12: Superellipse fuzzy number (variation m, n)

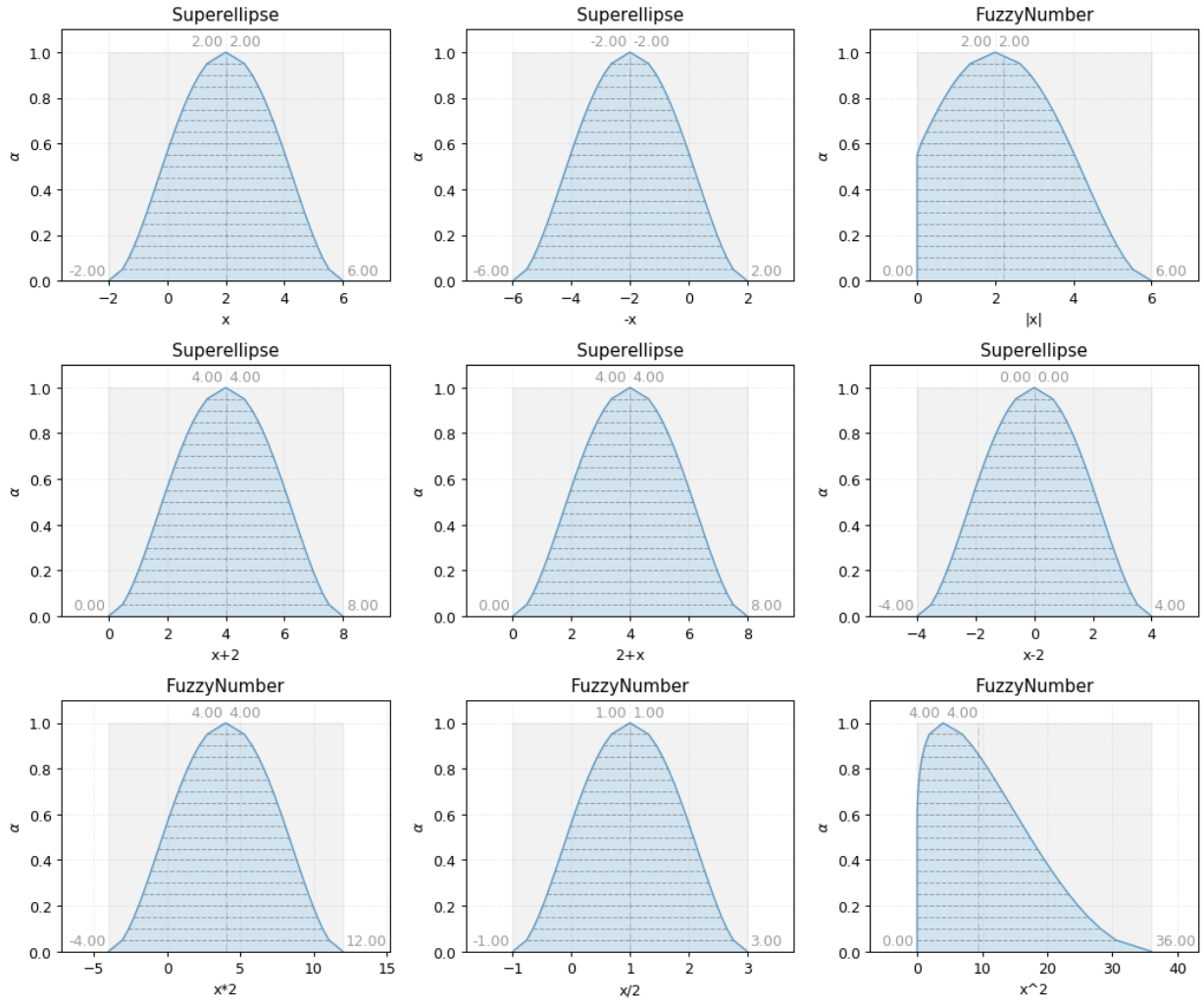

Fig. 13: Superellipse fuzzy number operations

CHAPTER 5

Plots

```
import phuzzy
from phuzzy.mpl import mix_mpl
import phuzzy.mpl.plots
x = phuzzy.TruncNorm(alpha0=[1, 2], name="x")
y = phuzzy.Triangle(alpha0=[3, 6], alpha1=[4], name="y")
mix_mpl(x)
x.plot(filepath="FuzzyNumber_plot.png")
```

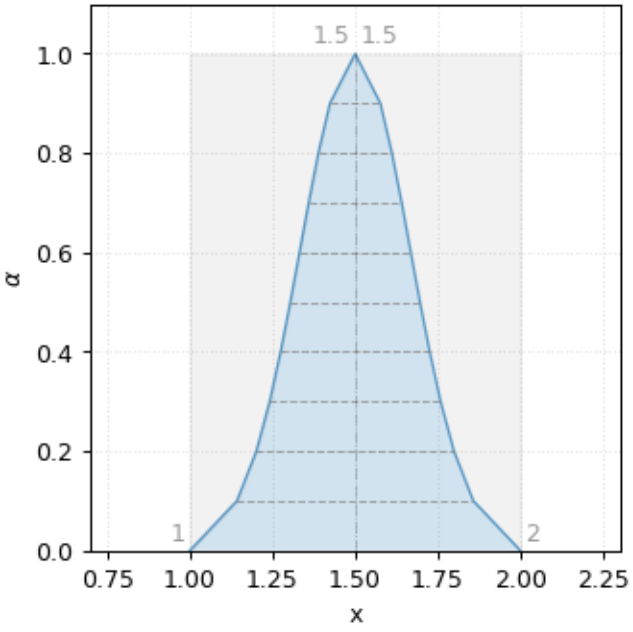

Fig. 1: x.plot()

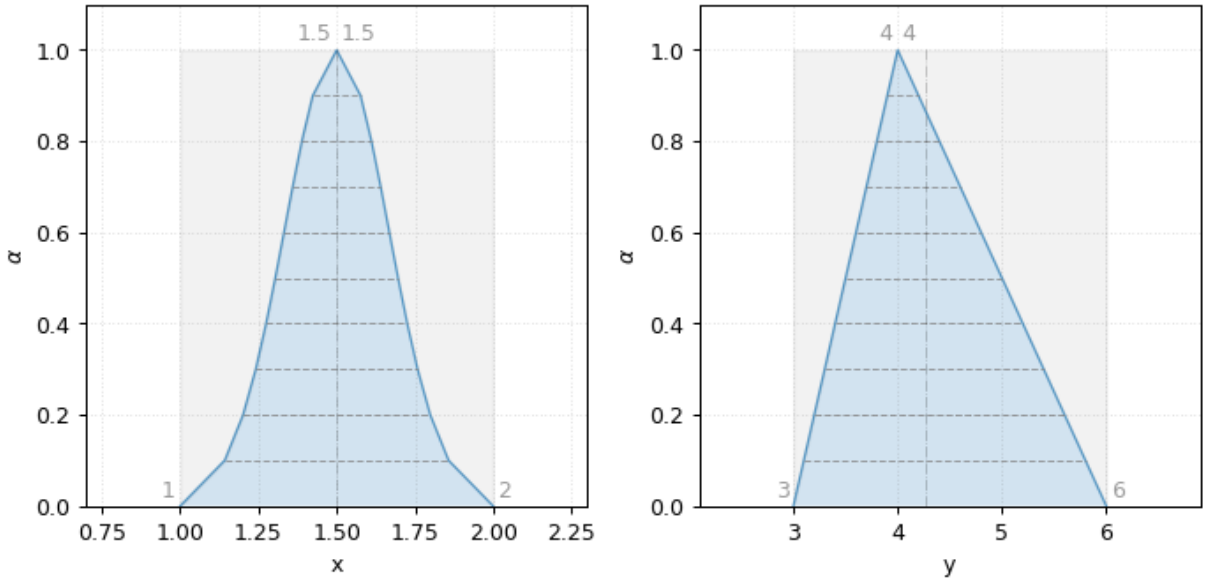

Fig. 2: fig, ax = phuzzy.mpl.plots.plot_xy(x, y)

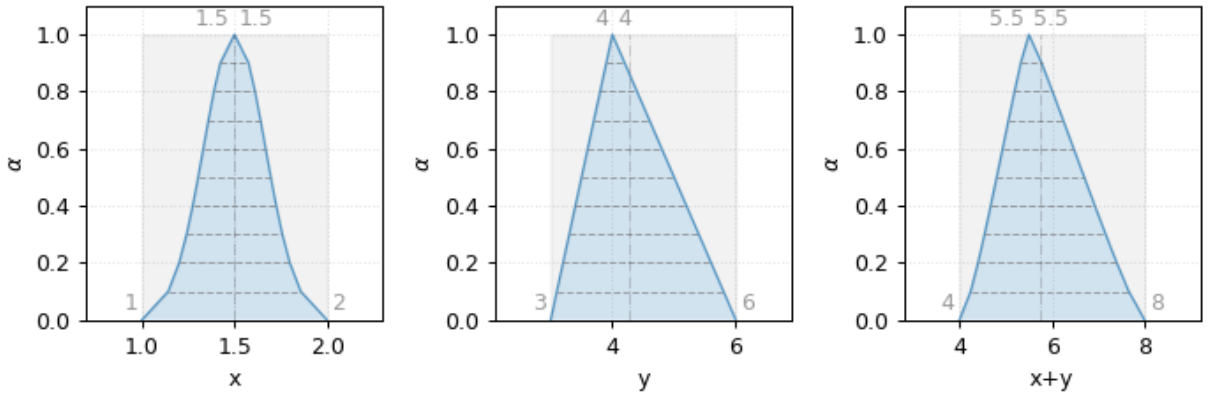

Fig. 3: fig, ax = phuzzy.mpl.plots.plot_xyz(x, y, x+y)

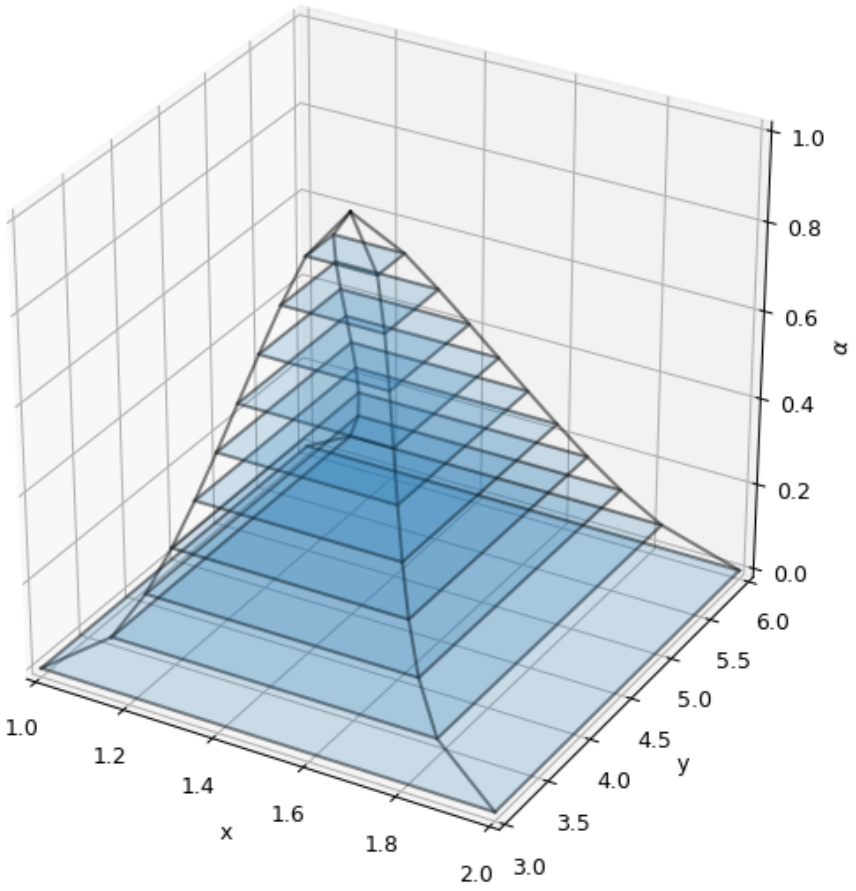

Fig. 4: fig, ax = phuzzy.mpl.plots.plot_3d(x, y)

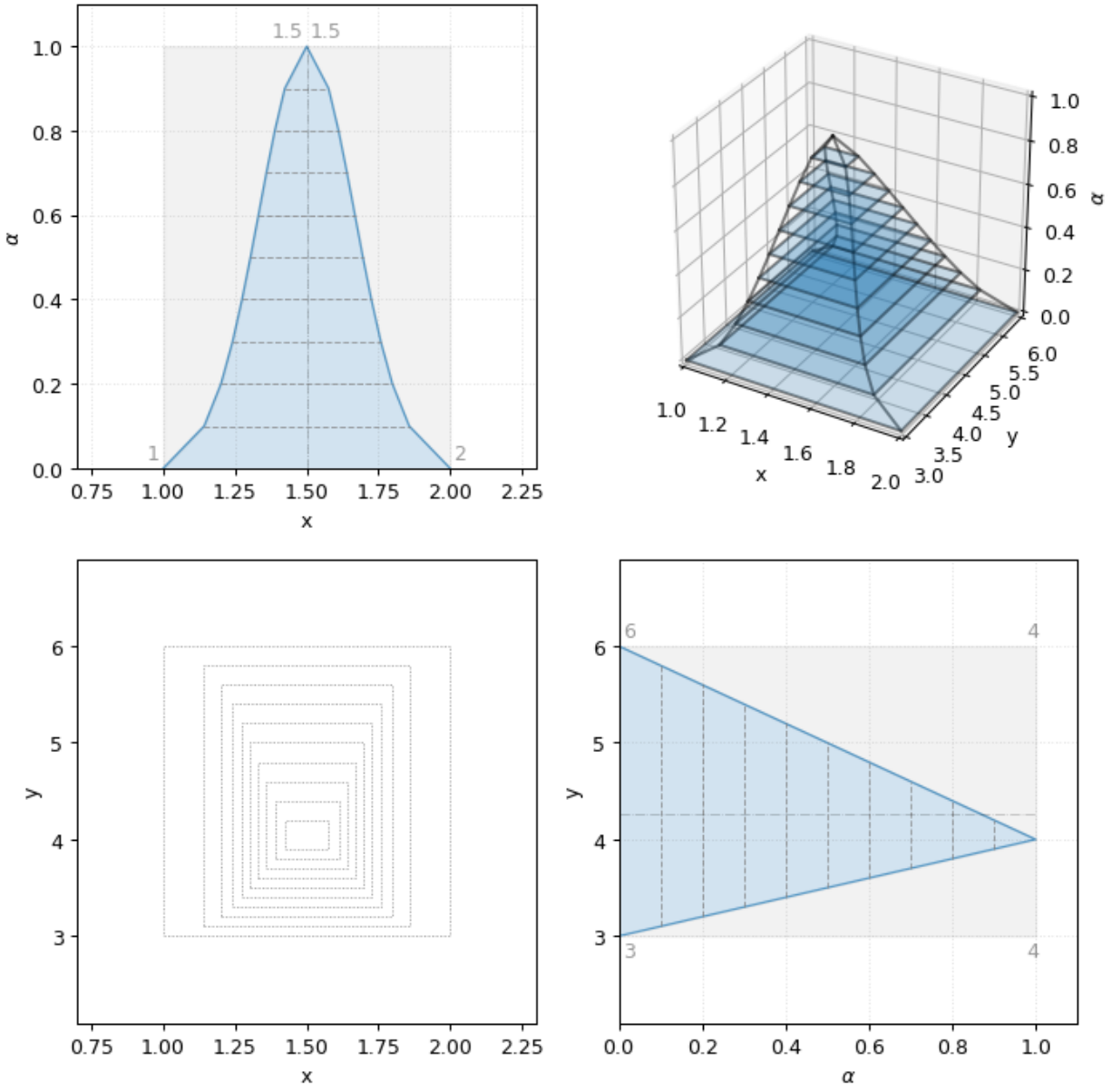

Fig. 5: fig, ax = phuzzy.mpl.plots.plot_xy_3d(x, y)

CHAPTER 6

Examples

6.1 Three-point bending

6.1.1 Simply supported beam with central load

What is the maximum deflection of a simple supported beam with central load, if there is an uncertainty of only 1% for all input parameter? The input parameter are load P, beam length L, beam width W, beam height H and young's modulus E.

$$
\begin{aligned}
P &= 5\,kN \pm 1\% \\
L &= 2\,m \pm 1\% \\
W &= 50\,mm \pm 1\% \\
H &= 100\,mm \pm 1\% \\
E &= 30000\,N/mm^2 \pm 1\%
\end{aligned}
$$

$$
\begin{aligned}
w(x) &= \begin{cases} -\frac{Px(4x^2-3L^2)}{48EI}, & \text{for } 0 \leq x \leq \frac{L}{2} \\ \frac{P(x-L)(L^2-8Lx+4x^2)}{48EI}, & \text{for } \frac{L}{2} < x \leq L \end{cases} \\
w &= w_{L/2} = \frac{PL^3}{48EI} \\
A &= WH \\
I &= WH^3
\end{aligned}
$$

6.1.1.1 Code

```
import phuzzy as ph
number_of_alpha_levels = 31
```

(continues on next page)

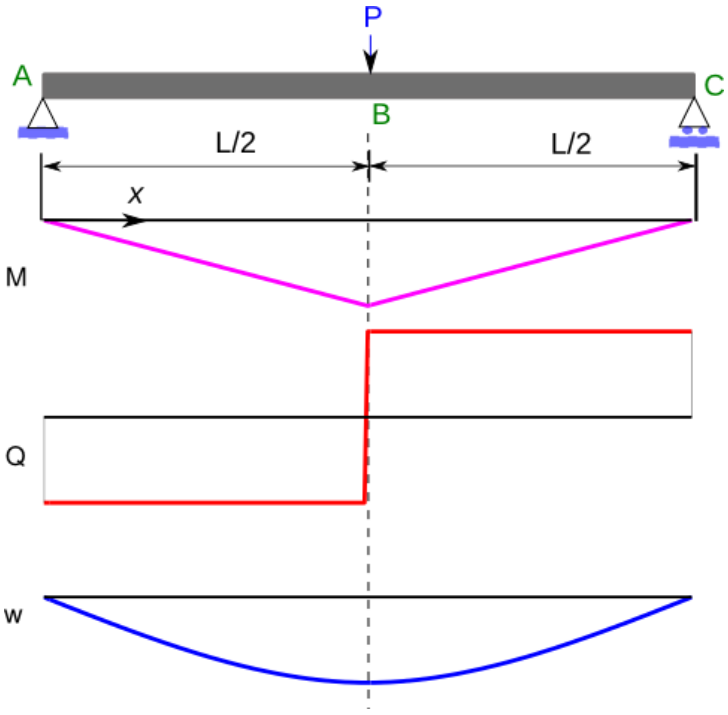

Fig. 1: Three point bending test [wikipedia.org]

(continued from previous page)

```
# load P
P0 = 5000.  # N
dP = 0.01 * P0  # N
P = ph.Triangle(alpha0=[P0 - dP, P0 + dP], alpha1=[P0], name="P", number_of_alpha_
↪levels=number_of_alpha_levels)

# dimensions L, W, H
W0 = 50  # mm
H0 = 100  # mm
L0 = 2000  # mm

dW = 0.01 * W0  # mm
dH = 0.01 * H0  # mm
dL = 0.01 * L0  # mm

L = ph.Triangle(alpha0=[L0 - dL, L0 + dL], alpha1=[L0], name="L", number_of_alpha_
↪levels=number_of_alpha_levels)
W = ph.Triangle(alpha0=[W0 - dW, W0 + dW], alpha1=[W0], name="W", number_of_alpha_
↪levels=number_of_alpha_levels)
H = ph.Triangle(alpha0=[H0 - dH, H0 + dH], alpha1=[H0], name="H", number_of_alpha_
↪levels=number_of_alpha_levels)

# material

E0 = 30000.  # N/mm2
dE = 0.1 * E0  # N/mm2
E = ph.TruncNorm(alpha0=[E0 - dE, E0 + dE], alpha1=[E0], name="E", number_of_alpha_
↪levels=number_of_alpha_levels)
```

(continues on next page)

(continued from previous page)

```
I0 = W0 * H0 ** 3 / 12.
w0 = P0 * L0 ** 3 / (48 * E0 * I0)

print("I0 = {:.4g} mm^4".format(I0))
# I0 = 4.167e+06 mm^4
print("w0 = {:.4g} mm".format(w0))
# w0 = 6.667 mm

I = W * H** 3 / 12.
I.name = "I"
w = P * L ** 3 / (48 * E * I)
w.name = r"P L^3 / (48 EI)"

print("I = {} mm^4".format(I))
# I = FuzzyNumber(W*H^3/12.0:[[4002483.375, 4335850.041666667], [4166666.6666666665,
↪4166666.6666666665]]) mm^4

print("w = {} mm".format(w))
# w = FuzzyNumber(P*L^3/E*48*W*H^3/12.0:[[5.594629603627992, 8.024370049019725], [6.
↪666666666666667, 6.6666666666666667]]) mm

w_mean = w.mean()
dw_l = w_mean - w.min()
dw_r = w.max() - w_mean
print("w = {:.4g} mm (- {:.4g}|+ {:.4g})".format(w_mean, dw_l, dw_r))
# w = 6.703 mm (- 1.109|+ 1.321)
print("w = {:.4g} mm [{:.4g},{:.4g}]".format(w_mean, w.min(), w.max()))
# w = 6.703 mm [5.595,8.024]
```

6.1.1.2 Parameter

6.1.1.3 Results

- Source code ssb.py

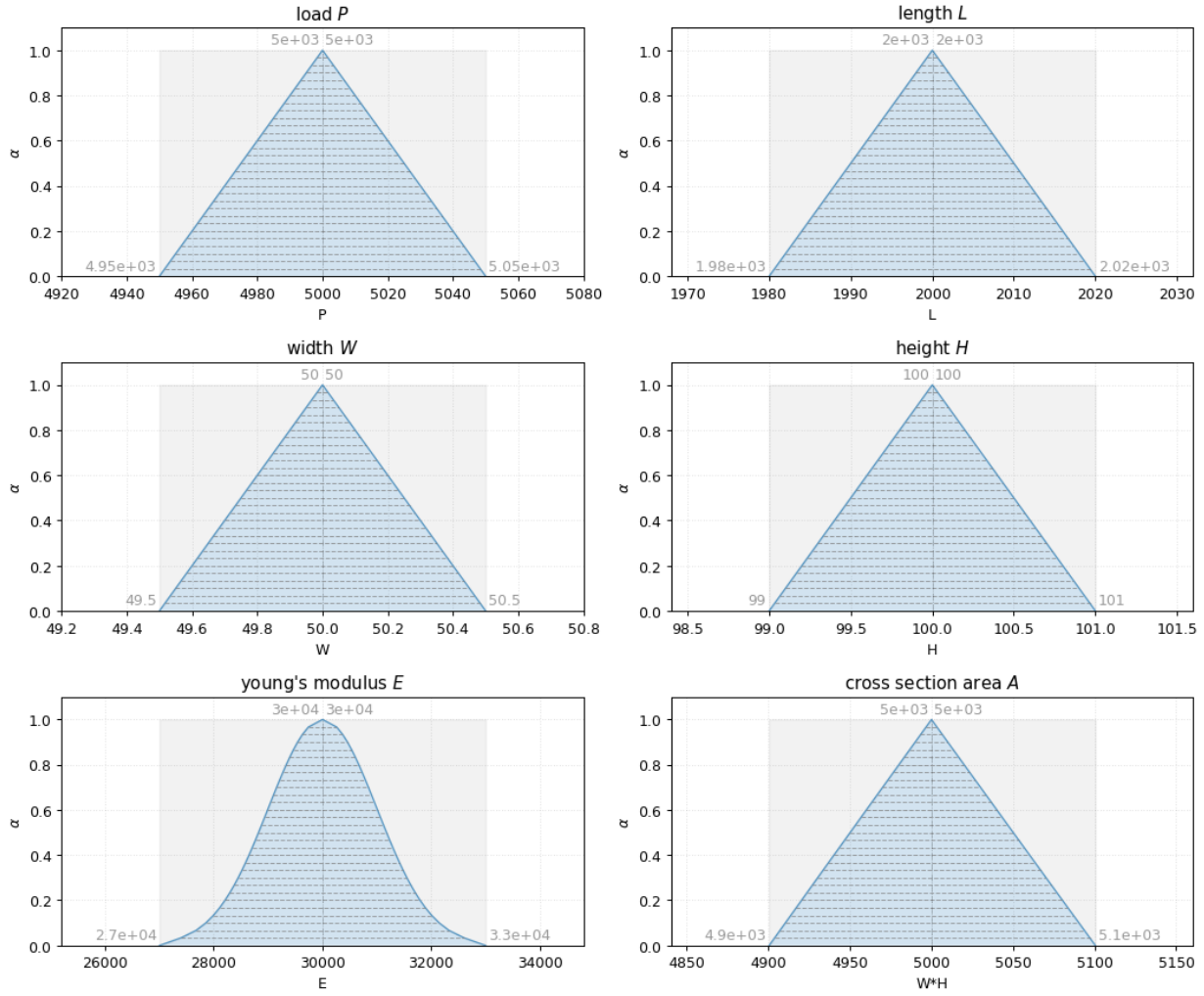

Fig. 2: used Parameter

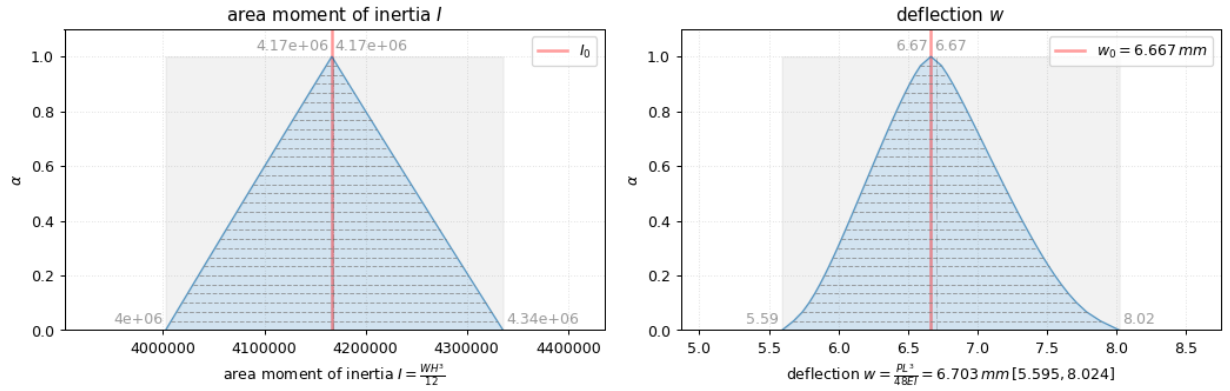

Fig. 3: results area moment of inertia `I` and deflections `w`

CHAPTER 7

phuzzy package

class phuzzy.**Analysis**(***kwargs*)

Bases: object

__dict__ = dict_proxy({'add_designvar': <function add_designvar>, '__module__': 'phu

__init__(***kwargs*)

Analysis(kwargs)

__module__ = 'phuzzy'

__repr__()

x.__str__() <==> str(x)

__str__() <==> *str(x)*

__weakref__

list of weak references to the object (if defined)

add_designvar(*designvar*)

add design variable to doe

Parameters **designvar** – design variable

Returns None

add_designvars(*designvars*)

add design variables to doe

Parameters **designvars** – list of design variables

Returns None

designvars

returns all design variables of doe

Returns dict of designvars

7.1 phuzzy.shapes package

class phuzzy.shapes.**FuzzyNumber**(***kwargs*)

Bases: object

convex fuzzy number

__abs__()

apply abs operator to a fuzzy number

Parameters **other** – phuzzy.FuzzyNumber

Returns fuzzy number

__add__(*other*)

adds a fuzzy number

Parameters **other** – phuzzy.FuzzyNumber

Returns fuzzy number

__contains__(*other*)

operator in

Return type bool

Returns True or False

__dict__ = dict_proxy({'cdf': <function cdf>, 'ppf': <function ppf>, 'get_shape': <

__div__(*other*)

divide by a fuzzy number

Parameters **other** – phuzzy.FuzzyNumber

Returns fuzzy number

__eq__(*other*)

operation ==

Return type bool

Returns True or False

__gt__(*other*)

operation >

Return type bool

Returns True or False

__hash__ = None

__init__(***kwargs*)

base fuzzy number

Parameters **kwargs** –

__lt__(*other*)

operation <

Returns True or False

__module__ = 'phuzzy.shapes'

__mul__(*other*)

multiply with a fuzzy number

Parameters **other** – phuzzy.FuzzyNumber

Returns fuzzy number

__ne__(*other*)
operation !=

Return type bool

Returns True or False

__neg__()
apply unary neg operator to a fuzzy number

Parameters **other** – phuzzy.FuzzyNumber

Returns fuzzy number

__pow__(*other*)
apply power of a fuzzy number

Parameters **other** – phuzzy.FuzzyNumber

Returns fuzzy number

__radd__(*other*)

__rdiv__(*other*)

__repr__() <==> *repr(x)*

__rmul__(*other*)

__rpow__(*other*)
apply exponent of a fuzzy number

Parameters **other** – phuzzy.FuzzyNumber

Returns fuzzy number

__rsub__(*other*)

__str__() <==> *str(x)*

__sub__(*other*)
substract a fuzzy number

Parameters **other** – phuzzy.FuzzyNumber

Returns fuzzy number

__truediv__(*other*)
divide by a fuzzy number

Parameters **other** – phuzzy.FuzzyNumber

Returns fuzzy number

__weakref__
list of weak references to the object (if defined)

abs()
calculate absolute value

Returns FuzzyNumber

alpha(*x*)
get alpha from x

alpha0
row for alpha=0

alpha1
row for alpha=1

cdf(*x*, ***kwargs*)
Cumulative distribution function

Parameters

- **x** – x values
- **n** – number of integration points

Returns y

convert_df(*alpha_levels=None*, *zero=0*)

copy()
return a copy

Returns copy of fuzzy number

defuzzification(*method='centroid'*)

defuzzification_alpha_one()
defuzzification

$$(alpha1_r + alpha1_l)/2$$

defuzzification_centroid()
defuzzification center of gravity

defuzzification_centroid2()
defuzzification center of gravity

defuzzification_mean()
defuzzification mean with discrete values

defuzzification_p50()
defuzzification mean my means of ppf(0.5)

df
number of alpha levels

discretize(*alpha0*, *alpha1*, *alpha_levels*)
discretize shape function

Parameters

- **alpha0** – range at alpha=0
- **alpha1** – range at alpha=1
- **alpha_levels** – number of alpha levels

Returns None

export_csv(*filepath=None*)
export alpha levels to csv

Parameters **filepath** – csv file path

Returns

classmethod from_data(***kwargs*)
instantiate fuzzy number from attributes

Parameters **kwargs** –

Return type phuzzy.FuzzyNumber or derived object

Returns fuzzy number

classmethod from_results(*df_res*, *name=None*, *number_of_alpha_levels=11*)
create FuzzyNumber from DataFrame("alpha", "res")

Parameters **df** – DataFrame with columns=["alpha", "res"]

Returns FuzzyNumber

classmethod from_str(*s*)
deserialize fuzzy number to string

Returns fuzzy number string

get_01
get alpha=0 and alpha=1 values

Returns [[a0_l, a0_r], [a1_l, a1_r]]

get_01_str
get alpha=0 and alpha=1 values

Returns [[a0_l, a0_r], [a1_l, a1_r]]

get_alpha_from_value(*x*)
get alpha values from given x values

Parameters **x** – x values

Returns alpha values

get_shape()
get shape dataframe

Returns pandas.DataFrame(columns=["alpha", "x"])

has_zero()
is zero in range

Return type bool

Returns True od False

import_csv(*fh*)
load alpha levels from csv

Parameters **fh** – csv file path or file handle

Returns alpha level dataframe

make_convex()
make fuzzy number convex

Returns None

max()
maximal

Return type float

Returns max value of df

mean()
mean value

Return type float

Returns mean value

min()
minimum

Return type float

Returns min value of df

number_of_alpha_levels
number of alpha levels

pdf(*x*)
Probability density function

Parameters **x** – x values

Returns

ppf(*x*, ***kwargs*)
Percent point function (inverse of cdf-percentiles).

Parameters

- **x** – x values
- **n** – number of integration points

Returns y

rvs(*size*, *seed=None*)
Sample points according membership function

Parameters **size** – number of sample points

Returns sample points

to_str()
serialize fuzzy number to string

Returns fuzzy number string

update(*alpha0=None*, *alpha1=None*, *alpha_levels=None*)

class phuzzy.shapes.**Trapezoid**(***kwargs*)
Bases: *phuzzy.shapes.FuzzyNumber*

triange fuzzy number

__init__(***kwargs*)
base fuzzy number

Parameters **kwargs** –

__module__ = 'phuzzy.shapes'

cdf(*x*, ***kwargs*)
Cumulative distribution function

Parameters

- **x** – x values

- **n** – number of integration points

Returns y

discretize(*alpha0*, *alpha1*, *alpha_levels*)
discretize shape function

Parameters

- **alpha0** – range at alpha=0
- **alpha1** – range at alpha=1
- **alpha_levels** – number of alpha levels

Returns None

pdf(*x*)

Parameters **x** –

Returns

to_str()
serialize fuzzy number to string

Returns fuzzy number string

class phuzzy.shapes.**Triangle**(***kwargs*)
Bases: *phuzzy.shapes.FuzzyNumber*

triange fuzzy number

__init__(***kwargs*)
base fuzzy number

Parameters **kwargs** –

__module__ = 'phuzzy.shapes'

cdf(*x*, ***kwargs*)
Cumulative distribution function

Parameters

- **x** – x values
- **n** – number of integration points

Returns y

discretize(*alpha0*, *alpha1*, *alpha_levels*)
discretize shape function

Parameters

- **alpha0** – range at alpha=0
- **alpha1** – range at alpha=1
- **alpha_levels** – number of alpha levels

Returns None

classmethod from_data(***kwargs*)
instantiate fuzzy number from attributes

Parameters **kwargs** –

Return type phuzzy.FuzzyNumber or derived object

Returns fuzzy number

pdf(*x*)
https://en.wikipedia.org/wiki/Triangular_distribution

to_str()
serialize fuzzy number to string

Returns fuzzy number string

class phuzzy.shapes.**Uniform**(***kwargs*)
Bases: *phuzzy.shapes.FuzzyNumber*

triange fuzzy number

__init__(***kwargs*)
base fuzzy number

Parameters **kwargs** –

__module__ = 'phuzzy.shapes'

cdf(*x*, ***kwargs*)
Cumulative distribution function

Parameters

- **x** – x values
- **n** – number of integration points

Returns y

discretize(*alpha0*, *alpha1*, *alpha_levels*)
discretize shape function

Parameters

- **alpha0** – range at alpha=0
- **alpha1** – range at alpha=1
- **alpha_levels** – number of alpha levels

Returns None

pdf(*x*)
https://en.wikipedia.org/wiki/Uniform_distribution_(continuous)

to_str()
serialize fuzzy number to string

Returns fuzzy number string

7.1.1 phuzzy.shapes.superellipse module

superelliptic fuzzy number

```
Superellipse(alpha0=[-1, 2], alpha1=None, m=1, n=.5, number_of_alpha_levels=15)
```

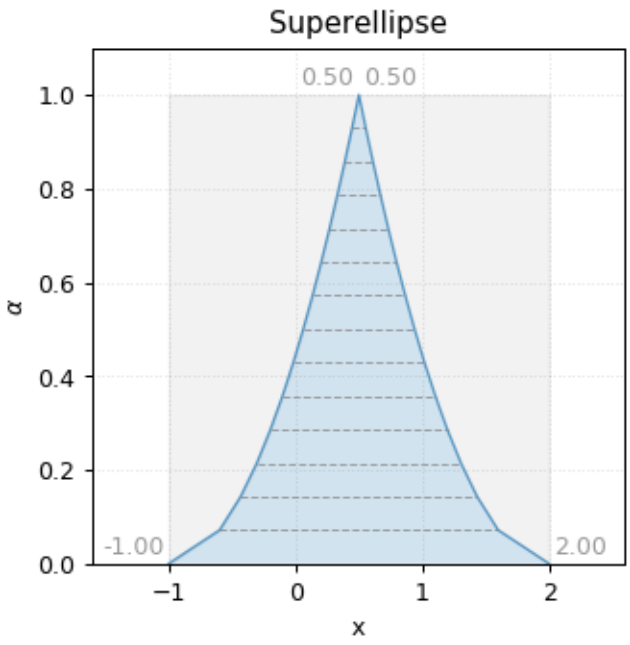

Fig. 1: Superellipse fuzzy number

class phuzzy.shapes.superellipse.**Superellipse**(***kwargs*)

Bases: *phuzzy.shapes.FuzzyNumber*

superelliptic fuzzy number

__init__(***kwargs*)

superelliptic fuzzy number

Parameters **kwargs** –

```
Superellipse(alpha0=[1, 2], alpha1=None, m=2, n=None, number_of_alpha_
↪levels=17)
```

__module__ = 'phuzzy.shapes.superellipse'

discretize(*alpha0*, *alpha1*, *alpha_levels*)

discretize shape function

Parameters

- **alpha0** – range at alpha=0
- **alpha1** – range at alpha=1
- **alpha_levels** – number of alpha levels

Returns None

classmethod from_str(*s*)

instantiate a fuzzy number from string

Parameters **s** –

Return type phuzzy.FuzzyNumber

Returns fuzzy number

shape(*x*)
shape function

Parameters **x** (*array or float*) – x values

Returns y values

to_str()
serialize fuzzy number to string

Return type str

Returns fuzzy string

7.1.2 phuzzy.shapes.truncnorm module

Normal distibuted membership function

```
TruncNorm(alpha0=[1, 3], alpha1=None, number_of_alpha_levels=15)
```

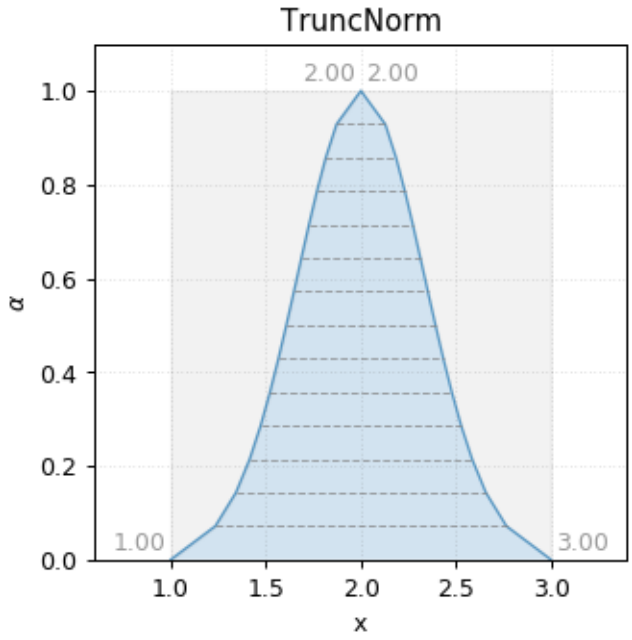

Fig. 2: TruncNorm fuzzy number

```
TruncGenNorm(alpha0=[1, 4], alpha1=None, number_of_alpha_levels=15, beta=5)
```

class phuzzy.shapes.truncnorm.**TruncGenNorm**(***kwargs*)
Bases: *phuzzy.shapes.FuzzyNumber*

Truncated generalized normal distibuted membership function

__init__(***kwargs*)
create a TruncNorm object

Parameters **kwargs** –

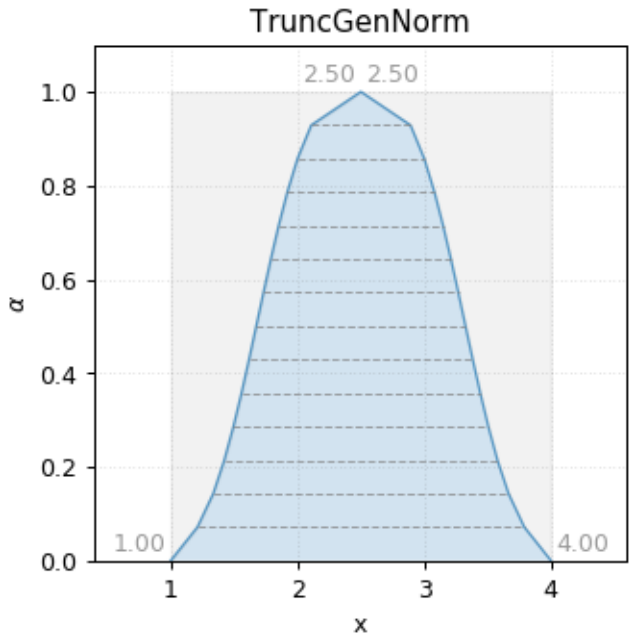

Fig. 3: TruncGenNorm fuzzy number

```
TruncGenNorm(alpha0=[1, 3], alpha1=None, number_of_alpha_levels=17, beta=3)
```

__module__ = 'phuzzy.shapes.truncnorm'

discretize(*alpha0*, *alpha1*, *alpha_levels*)
discretize shape function

Parameters

- **alpha0** – range at alpha=0
- **alpha1** – range at alpha=1
- **alpha_levels** – number of alpha levels

Returns None

distr

loc

mean

scale

std

class phuzzy.shapes.truncnorm.**TruncNorm**(***kwargs*)
Bases: *phuzzy.shapes.FuzzyNumber*

Normal distibuted membership function

__init__(***kwargs*)
create a TruncNorm object

Parameters **kwargs** –

```
TruncNorm(alpha0=[1, 3], alpha1=None, number_of_alpha_levels=17)
```

__module__ = 'phuzzy.shapes.truncnorm'

discretize(*alpha0*, *alpha1*, *alpha_levels*)
discretize shape function

Parameters

- **alpha0** – range at alpha=0
- **alpha1** – range at alpha=1
- **alpha_levels** – number of alpha levels

Returns None

distr
calculate truncated normal distribution

Returns distribution object

loc
mean value

Return type float

Returns mean value aka location

mean
mean value

Return type float

Returns mean value aka location

scale
standard deviation

Return type float

Returns standard deviation

std
standard deviation

Return type float

Returns standard deviation

7.2 phuzzy.mpl module

class phuzzy.mpl.**FuzzyNumber**(***kwargs*)
Bases: *phuzzy.shapes.FuzzyNumber*, *phuzzy.mpl.MPL_Mixin*

Uniform fuzzy number with matplotlib mixin

__init__(***kwargs*)
base fuzzy number

Parameters **kwargs** –

__module__ = 'phuzzy.mpl'

class phuzzy.mpl.**MPL_Mixin**

__module__ = 'phuzzy.mpl'

plot(*ax=None*, *filepath=None*, *show=False*, *xlim=None*, *labels=True*, *title=False*, *ppf=None*)
plots fuzzy number with mpl

vplot(*ax=None*, *filepath=None*, *show=False*, *xlim=None*, *labels=True*, *title=False*, *ppf=None*)
plots fuzzy number with mpl

class phuzzy.mpl.**Superellipse**(***kwargs*)
Bases: *phuzzy.shapes.superellipse.Superellipse*, *phuzzy.mpl.MPL_Mixin*

Superellipse fuzzy number with matplotlib mixin

__init__(***kwargs*)
superelliptic fuzzy number

Parameters **kwargs** –

```
Superellipse(alpha0=[1, 2], alpha1=None, m=2, n=None, number_of_alpha_
↪levels=17)
```

__module__ = 'phuzzy.mpl'

class phuzzy.mpl.**Trapezoid**(***kwargs*)
Bases: *phuzzy.shapes.Trapezoid*, *phuzzy.mpl.MPL_Mixin*

Trapezoid fuzzy number with matplotlib mixin

__init__(***kwargs*)
base fuzzy number

Parameters **kwargs** –

__module__ = 'phuzzy.mpl'

class phuzzy.mpl.**Triangle**(***kwargs*)
Bases: *phuzzy.shapes.Triangle*, *phuzzy.mpl.MPL_Mixin*

Triangle fuzzy number with matplotlib mixin

__init__(***kwargs*)
base fuzzy number

Parameters **kwargs** –

__module__ = 'phuzzy.mpl'

class phuzzy.mpl.**TruncGenNorm**(***kwargs*)
Bases: *phuzzy.shapes.truncnorm.TruncGenNorm*, *phuzzy.mpl.MPL_Mixin*

Truncates general normal fuzzy number with matplotlib mixin

__init__(***kwargs*)
create a TruncNorm object

Parameters **kwargs** –

```
TruncGenNorm(alpha0=[1, 3], alpha1=None, number_of_alpha_levels=17, beta=3)
```

__module__ = 'phuzzy.mpl'

class phuzzy.mpl.**TruncNorm**(***kwargs*)
Bases: *phuzzy.shapes.truncnorm.TruncNorm*, *phuzzy.mpl.MPL_Mixin*

TruncNorm fuzzy number with matplotlib mixin

__init__(***kwargs*)
create a TruncNorm object

Parameters **kwargs** –

```
TruncNorm(alpha0=[1, 3], alpha1=None, number_of_alpha_levels=17)
```

__module__ = 'phuzzy.mpl'

class phuzzy.mpl.**Uniform**(***kwargs*)
Bases: *phuzzy.shapes.Uniform*, *phuzzy.mpl.MPL_Mixin*

Uniform fuzzy number with matplotlib mixin

__init__(***kwargs*)
base fuzzy number

Parameters **kwargs** –

__module__ = 'phuzzy.mpl'

phuzzy.mpl.**extend_instance**(*obj*, *cls*)
Apply mixins to a class instance after creation

phuzzy.mpl.**mix_mpl**(*obj*)

CHAPTER 8

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

8.1 Types of Contributions

8.1.1 Report Bugs

Report bugs at https://github.com/lepy/phuzzy/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

8.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

8.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

8.1.4 Write Documentation

phuzzy could always use more documentation, whether as part of the official phuzzy docs, in docstrings, or even on the web in blog posts, articles, and such.

8.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/lepy/phuzzy/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

8.2 Get Started!

Ready to contribute? Here's how to set up *phuzzy* for local development.

1. Fork the *phuzzy* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/phuzzy.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv phuzzy
$ cd phuzzy/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 phuzzy tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

8.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/lepy/phuzzy/pull_requests and make sure that the tests pass for all supported Python versions.

8.4 Tips

To run a subset of tests:

```
$ py.test tests.test_phuzzy
```

8.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 9

Credits

9.1 Development Lead

- Lepy <lepy@mailbox.org>

9.2 Contributors

None yet. Why not be the first?

CHAPTER 10

History

10.1 0.4.0 (2018-04-13)

- First release on PyPI.

10.2 0.5.0 (2018-04-16)

- rename FuzzyNumber.df.columns = [“alpha”, “l”, “r”]
- add operators __add__
- add operators __sub__
- add operators __mul__
- add operators __div__

10.3 0.6.0 (2018-04-19)

- add operators __radd__, __rsub__, __rmul__, __rdiv__, __truediv__
- add operators __neg__, __abs__, __min__, __max__, __mean__
- add operators __lt__, __gt__, __eq__, __contain__

10.4 0.7.0 (2018-04-24)

- add plots
- add approximated function evaluation

lsdyna

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

R

S

T

U

V