Library lifeActuary

Pedro Corte Real & Gracinda Rita Guerreiro

June 19, 2023

Type: Python Package

Version: 1.3.2

Authors: Pedro Corte Real & Gracinda R. Guerreiro

Contact: parcr@fct.unl.pt & grg@fct.unl.pt

Repository: https://github.com/parcr/lifeactuary_1.3.2

Abstract

lifeActuary is a Python library to perform actuarial mathematics on life contingencies and classical financial mathematics computations. Versatile, simple and easy to use. The main functions are implemented using the usual actuarial approach, making it a natural choice for the life actuary.

This document is produced as a descriptive tool on how to use the package and as a user guide for the developed actuarial functions. For each actuarial function, an illustrative example is provided.

The package uses Python version 3.7 or higher. The package and functions herein provided were tested, but the authors distribute them without any guarantee regarding the accuracy of calculations. It's distributed using the MIT License and the authors disclaim any liability arising from any losses due to direct or indirect use of this package.

Version 1.3 includes new functions devoted to joint mortality of two lives that allows for the evaluation of **Joint Life** insurance policies. The new functions allow for the computation of joint life probabilities, annuities and insurance benefits, for both *joint life* and *last survivor* benefits.

Additionally, in this version:

– We changed the class **annuities.py** and now all the life annuity functions do not rely on the computation of the actuarial table. The previous functions are still available in commutationtable.py.

– We developed a new class **mortality_insurance.py** where functions for evaluating life insurance contracts are available without the need to compute the commutation table. Several new functions were included. The previous functions are still available in commutationtable.py.

– We made a small correction to the function "present_value" and a large correction to the functions "t_nIArx" and "t_nIArx_" in commutation_table.py.

- We made a small correction to the function "get_integral_px_method" in mortality_table.py.
- We made a small correction to the function "annuity_x" in annuities.py.
- We made a small correction to the "npx" and "nqx" functions in the module mortality_table.py, when producing the fractional commutation tables using the Balducci and the Constant Mortality Force.

This package is still under development and further useful and interesting functions will be available any time soon.

Contents

1 Introduction

The *lifeActuary* library for Python aims to provide a wide range of actuarial functions for life contingencies. The names of the functions follow the International Actuarial Notation and are intuitive (qx, px, lx, an, axn, nEx, Ax, ...) and the common parameters are set as usual (x for actuarial age, n for term of the contract, p for term of payments, ...). This aims to provide with an easy to use and "guessable" list of functions.

Using mortality tables, the library provides functions for computing (a) survival probabilities for integer and non-integer ages and terms, (b) life expectancy for both integer and non integer ages and terms, (c) expected present value of life annuities and (d) expected present value of traditional life insurances. All these features allow for the development of simple or more complicated actuarial evaluations and product developments.

This library can be used for academic or professional purposes. The library contains the functions of main traditional products in Life Insurance, allows for an easy computation of actuarial tables but also provides the tools for designing new products, building tariffs, computing reserves.

It incorporates a very generic and simple to use function to compute the Expected Present Value (*aka* actuarial expected present value) what is becoming a very relevant tool in the solvency analysis.

A set of examples is presented in the end of this manual, showing some potential uses of this library in life contingencies products and evaluations.

This library is still under development and further useful functions will be available any time soon. In the next release, the package will include functions that allow the use of continuous models as well as the computation of variance of life annuities and variance of classical life insurances.

2 Mortality Tables

The functions developed on this section are related to common actuarial biometric functions, computed upon a mortality table.

2.1 Class MortalityTable

---

class *MortalityTable*
This class instantiates a life table. Data can be provided in the form of the l_x, q_x or p_x. Note that the first value is the first age considered in the table. The life table will be complete, that is, from age 0 to age ω (the last age where $l_x > 0$). It includes the computation of common biometric functions: l_x, p_x, q_x, d_x, e_x for integer ages and for non-integers ages, using methods as Uniform Distribution of Death (udd), Constant Force of Mortality (cfm) and Balducci approximation (bal).

Usage

```
MortalityTable(data_type='q', mt=None, perc=100, last_q=1)
```

Description
Initializes the MortalityTable class so that we can construct a mortality table with the usual fields.

Parameters

data_type	Use 'l' for l_x, 'p' for p_x and 'q' for q_x.
mt	The mortality table, in array format, according to the data_type defined
perc	The percentage of q_x to use, e.g., use 50 for 50%.
last_q	The value for q_ω.

2.2 Importing Mortality Tables

The package includes a wide number of mortality tables and allows for the inclusion of any other mortality tables extracted from the Society of Actuaries (SOA), in xml format, or by importing other ones in usual formats, such as xlsx, csv, txt files.
For instance, in the manual, one of the tables that we will be using is the TV7377 in the xml format supported by the Society of Actuaries.

2.2.1 Reading from lifeActuary Library

Example

```
from lifeActuary import mortality_table as mt
from soa_tables import read_soa_table_xml as rst

# reads TV7377 mortality table from SOA table
soa = rst.SoaTable('soa_tables/' + 'TV7377' + '.xml')

# creates mortality table from qx of SOA table
tv7377 = mt.MortalityTable(data_type='q', mt=soa.table_qx, perc=100, last_q=1)
```

2.2.2 Importing from File

When building a new mortality table to import from a file, please note that the first value of the table corresponds to the first age considered in the table. For instance, if the first value of the table is 20, it means that $l_x = 0$, for $x = 0, \dots, 19$.

Usage

```
from lifeActuary import mortality_table as mt
import pandas as pd

# reads manually imported mortality table
table_manual_qx = pd.read_excel('soa_tables/' + 'tables_manual' + '.xlsx', sheet_name='qx')
table_manual_lx = pd.read_excel('soa_tables/' + 'tables_manual' + '.xlsx', sheet_name='lx')

# creates mortality table from lx of a xlsx file
grf95 = mt.MortalityTable(data_type='q', mt=list(table_manual_qx['GRF95']), perc=80)
grm95 = mt.MortalityTable(data_type='l', mt=list(table_manual_lx['GRM95']), perc=80)
```

2.3 Demographic Functions

After the mortality table is instantiated (here denoted by mt), the common demographic functions are available in the package, such as l_x (expected number of subjects alive at age x), d_x (expected number of deaths with age x), q_x (mortality rate at age x), e_x (complete life expectancy at age x), ω (terminal age of the mortality table):

2.3.1 lx[x]

Actuarial Notation	l_x
Usage	mt.lx[x]
Args	x: age as an integer number
Example	tv7377.lx[50]
Result	94055.99997478718

2.3.2 dx[x]

Actuarial Notation	d_x
Usage	mt.dx[x]
Args	x: age as an integer number
Example	tv7377.dx[50]
Result	353.99999675630613

2.3.3 qx[x]

Actuarial Notation	q_x
Usage	mt.qx[x]
Args	x: age as an integer number
Example	tv7377.qx[50]
Result	0.0037637152

2.3.4 px[x]

Actuarial Notation	p_x
Usage	mt.qx[x]
Args	x: age as an integer number
Example	tv7377.px[50]
Result	0.9962362848

2.3.5 ex[x]

Actuarial Notation	e_x
Usage	mt.ex[x]
Args	x: age as an integer number
Example	tv7377.ex[50]
Result	30.07981415164423

2.3.6 w

Actuarial Notation	ω
Usage	mt.w
Example	tv7377.w
Result	106

Other Examples

```
## Consulting information from an object

# Outputs the information necessary to clone the object
tv7377

# consults the lx of TV7377
tv7377.lx

# Consults the ex of GRF95
grf95.ex

# extracts all methods from the object grm95
grm95.__dict__
```

2.4 Survival Probabilities Functions

The package also allows for the direct computation of survival probabilities for an aged x individual. Focusing on the common actuarial probabilities, some functions are available for the computations of the following probabilities: ${}_nq_x$, ${}_np_x$, ${}_{t|n}q_x$ for integer and non-integer ages and periods. In fact, in this library, the non-integer ages and periods are just a particular case when using any method.

2.4.1 nqx

Actuarial Notation: ${}_nq_x$

Usage

```
nqx(x, n=1, method='udd')
```

Description: Returns the probability of (x) dying before age $x + n$.

Parameters

x	age at the beginning
n	number of years
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# probability that (50) dies before age 52.
tv7377.nqx(50, 2) # 0.0078038614928698236

# probability that an aged 50.5 individual dies before age 53.
tv7377.nqx(50.5, 2.5 ,method='udd') # 0.010321797187509807
tv7377.nqx(50.5, 2.5 ,method='cfm') # 0.010320038151286903
tv7377.nqx(50.5, 2.5 ,method='bal') # 0.010318279111937612
```

2.4.2 npx

Actuarial Notation: ${}_np_x$

Usage

```
npx(x, n=1, method='udd')
```

Description: Returns the probability of (x) surviving beyond age $x + n$.

Parameters

x	age at the beginning
n	number of years
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# probability that (80) reaches age 82.
tv7377.npx(80, 2) # 0.8563257446904969

# probability that an aged 80.5 individual reaches age 85.
tv7377.npx(80.5, 4.5, method='udd') # 0.3512032870461814
tv7377.npx(80.5, 4.5, method='cfm') # 0.3507713780990377
tv7377.npx(80.5, 4.5, method='bal') # 0.35033918162679567
```

2.4.3 t_nqx

Actuarial Notation: ${}_{t|n}q_x$

Usage

```
t_nqx(x, t=1, n=1, method='udd')
```

Description: Returns the probability of (x) surviving beyond age $x+t$ and die before age $x+t+n$.

Parameters

x	age at the beginning
t	deferment period
n	number of years
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# probability that (30) reaches age 40, but dies before age 50.
tv7377.t_nqx(30, 10, 20) # 0.07505208397820314

# probability that an aged 80.5 individual reaches age 85 but dies before age 90.5.
tv7377.t_nqx(80.5, 4.5, 10.5, 'udd') # 0.577558207777435
tv7377.t_nqx(80.5, 4.5, 10.5, 'cfm') # 0.5787577102068303
tv7377.t_nqx(80.5, 4.5, 10.5, 'bal') # 0.5799492293567563
```

2.5 Life Expectancy Function

The library allows for the computation of Complete Life Expectancy for integer and non-integer ages and periods.

Usage

```
exn(x, n, method='udd')
```

Description: Returns the life expectancy for (x) over the next n years.

Parameters

x	age at the beginning
n	number of years
method	For non-integer ages, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

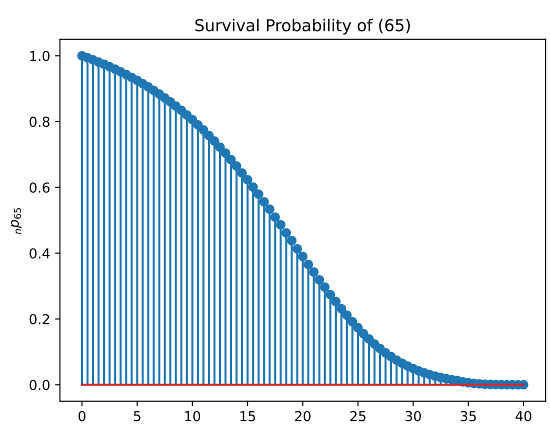
Examples

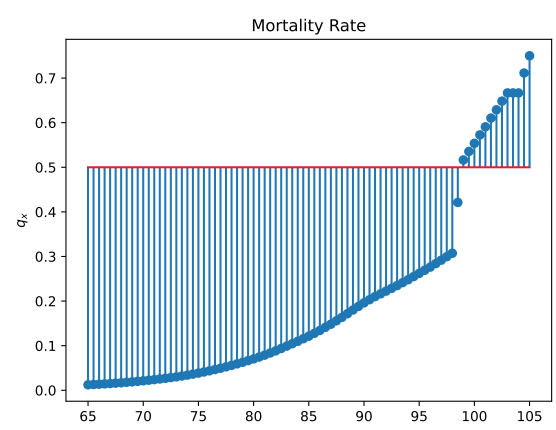
```
# complete life expectancy for (60) over the next 10 years
tv7377.exn(60, 10)        # 9.498277332706456
tv7377.exn(60, 10, 'cfm') # 9.498146560076156
tv7377.exn(60, 10, 'bal') # 9.498015788406414

```

```python
# complete life expectancy for (60.1) over the next 10.2 years
tv7377.exn(60.1, 10.2)        # 9.673511678284852
tv7377.exn(60.1, 10.2, 'cfm') # 9.67338786347054
tv7377.exn(60.1, 10.2, 'bal') # 9.673264041773342
```

2.6 Some Examples

```python
# Consider a 65 years old individual. Using TV7377 mortality table and considering Constant Force of Mortality approximation. :

import numpy as np
import matplotlib.pyplot as plt

# 1 - Compute the probability of (65) to survive $t$ years, $t=0, 0.5, 1, 1.5, ..., 19.5, 40$. Plot the probability.

x = 65
n = np.linspace(0, 40, num=2*40+1)
sprob = [tv7377.npx(x=x, n=i, method='cfm') for i in n]
plt.stem(n, sprob)
plt.xlabel('n')
plt.ylabel(r'${}_{n}p_{65}$')
plt.title('Survival Probability of (65)')
plt.savefig('example26' + '.eps', format='eps', dpi=3600)
plt.show()

# 2 - Compute the mortality rate for ages (65+t), $t=0, 0.5, 1, 1.5, ..., 19.5, 40$. Plot the obtained results, highlighting the differences to a 0.5 probability.

dprob = [tv7377.nqx(x=x+i, n=1, method='cfm') for i in n]
ages=x+n
plt.stem(ages, dprob, bottom=0.5)
plt.xlabel('x')
plt.ylabel(r'$q_{x}$')
plt.title('Mortality Rate')
plt.savefig('example26b' + '.pdf', format='pdf', dpi=3600)
plt.show()
```





3 Some lifeActuary Functions and Syntax

3.1 Mortality Table data and Survival Probabilities

Table 1: Actuarial Notation and Syntax Formula for Survival Probabilities

Notation	Description	Syntax
l_x	Expected number of living individuals aged x	lx[x]
d_x	Expected number of deaths with age x	dx[x]
q_x	Mortality rate at age x	qx[x]
p_x	Survival probability at age x	px[x]
e_x	Life Expectancy at age x	ex[x]
ω	Limit age of mortality table	w

3.2 Survival Probabilities for groups of two individuals

Table 2: Actuarial Notation and Syntax Formula for Survival Probabilities in joint-life groups

Notation	Description	Syntax
${}_np_{xy}$	Probability of a joint-life group survives at least n years	npxy(mtx, mty, x, y, n, 'joint-life', method)
${}_nq_{xy}$	Probability of a joint-life group extinguishes in the following n years	nqxy(mtx, mty, x, y, n, 'joint-life', method)
${}_{t\|n}q_{xy}$	Probability of a joint-life group at least t years and extinguishes before $t+n$ years	t_npxy(mtx, mty, x, y, n, 'joint-life', method)
e_{xy}	Complete life expectancy for a joint-life group	exy(mtx, mty, x, y, n, 'joint-life', method)
$e_{xy:\overline{n}\|}$	Life expectancy for a joint-life group, for the next n years	nexy(mtx, mty, x, y, n, 'joint-life', method)

Table 3: Actuarial Notation and Syntax Formula for Survival Probabilities in last survivor groups

Notation	Description	Syntax
${}_np_{\overline{xy}}$	Probability of a last-survivor group survives at least n years	npxy(mtx, mty, x, y, n, 'last-survivor', method)
${}_nq_{\overline{xy}}$	Probability of a last-survivor group extinguishes in the following n years	nqxy(mtx, mty, x, y, n, 'last-survivor', method)
${}_{t\|n}q_{\overline{xy}}$	Probability of a last-survivor group at least t years and extinguishes before $t+n$ years	t_npxy(mtx, mty, x, y, n, 'last-survivor', method)
$e_{\overline{xy}}$	Complete life expectancy for a last-survivor group	exy(mtx, mty, x, y, n, 'last-survivor', method)
$e_{\overline{xy}:\overline{n}\|}$	Life expectancy for a last-survivor group, for the next n years	nexy(mtx, mty, x, y, n, 'last-survivor', method)

3.3 Life Annuities

Table 4: Actuarial Notation and Syntax Formula for Life Annuities

Notation	Description	Syntax
a_x	whole life annuity	ax(mt, x, i, g=0, m=1, method)
$\ddot{a}_x$	whole life annuity due	aax(mt, x, i, g=0, m=1, method)
${}_{t\|}a_x$	t years deferred whole life annuity	t_ax(mt, x, i, g=0, m=1, defer=t, method)
${}_{t\|}\ddot{a}_x$	t years deferred whole life annuity due	t_aax(mt, x, i, g=0, m=1, defer=t, method)
$a_x^{(m)}$	whole life annuity payable m times per year	ax(mt, x, i, g=0, m=m, method)
$\ddot{a}_x^{(m)}$	whole life annuity due payable m times per year	aax(mt, x, i, g=0, m=m, method)
${}_{t\|}a_x^{(m)}$	t years deferred whole life annuity payable m times per year	t_ax(mt, x, i, g=0, m=m, defer=t, method)
${}_{t\|}\ddot{a}_x^{(m)}$	t years deferred whole life annuity due payable m times per year	t_aax(mt, x, i, g=0, m=m, defer=t, method)
$a_{x:\overline{n}\|}$	n year temporary life annuity	nax(mt, x, n, i, g=0, m=1, method)
$\ddot{a}_{x:\overline{n}\|}$	n year temporary life annuity due	naax(mt, x, n, i, g=0, m=1, method)
${}_{t\|}a_{x:\overline{n}\|}$	t year deferred n year temporary life annuity	t_nax(mt, x, n, i, g=0, m=1, defer=t, method)
${}_{t\|}\ddot{a}_{x:\overline{n}\|}$	t year deferred n year temporary life annuity due	t_naax(mt, x, n, i, g=0, m=1, defer=t, method)
$a_{x:\overline{n}\|}^{(m)}$	n year temporary life annuity payable m times per year	nax(mt, x, n, i, g=0, m=m, method)
$\ddot{a}_{x:\overline{n}\|}^{(m)}$	n year temporary life annuity due payable m times per year	naax(mt, x, n, i, g=0, m=m, method)
${}_{t\|}a_{x:\overline{n}\|}^{(m)}$	t year deferred n year temporary life annuity payable m times per year	t_nax(mt, x, n, i, g=0, m=m, defer=t, method)
${}_{t\|}\ddot{a}_{x:\overline{n}\|}^{(m)}$	t year deferred n year temporary life annuity due payable m times per year	t_naax(mt, x, n, i, g=0, m=m, defer=t, method)

Table 5: Actuarial Notation and Syntax Formula for Increasing Life Annuities

Notation	Description	Syntax
$(Ia)_{x:\overline{n}\|}^{(m)r}$	n-year temporary increasing life annuity, payable m times per year. First payment 1 and increasing/decreasing amount r	nIax(mt, x, n, i, m=m, first_amount=1, increase_amount=r, method)
$(I\ddot{a})_{x:\overline{n}\|}^{(m)r}$	n-year temporary due increasing life annuity, payable m times per year. First payment 1 and increasing/decreasing amount r	nIaax(mt, x, n, i, m=m, first_amount=1, increase_amount=r, method)
${}_{t\|}(Ia)_{x:\overline{n}\|}^{(m)r}$	t-years deferred n-year temporary increasing life annuity, payable m times per year. First payment 1 and increasing/decreasing amount r	t_nIax(mt, x, n, i, m=m, defer=t, first_amount=1, increase_amount=r, method)
${}_{t\|}(I\ddot{a})_{x:\overline{n}\|}^{(m)r}$	t-years deferred n-year temporary due increasing life annuity, payable m times per year. First payment 1 and increasing/decreasing amount r	t_nIaax(mt, x, n, i, m=m, defer=t, first_amount=1, increase_amount=r, method)

Remark - Geometric Annuities:

For life annuities with terms varying geometrically, the Actuarial Table must be built with a growth rate g and the functions from Table 4 are applied.

Important Remark - Commutation Symbols:

If the user intends to compute life annuities using commutation symbols, as in version 1.2, the correspondent functions are resumed in section 7.

3.4 Life Insurances

The following table resumes the available function for life insurances. As usual in the actuarial notation, the capital letters with bar refer to payments due in the "moment of death" and the absense of bar refers to payments due in the end of the year in which the death occurs.

Table 6: Actuarial Notation and Syntax Formula for Life Insurances - fixed capitals

Notation	**Description**	**Syntax**
${}_nE_x$	pure endowment	nEx(mt, x, i, g=0, n, method)
A_x	whole life insurance (end of the year)	Ax(mt, x, i, g=0, method)
$\bar{A}_x$	whole life insurance (moment of death)	Ax_(mt, x, i, g=0, method)
${}_{t\|}A_x$	t years deferred whole life insurance (end of the year)	t_Ax(mt, x, defer=t, i, g=0, method)
${}_{t\|}\bar{A}_x$	t years deferred whole life insurance (moment of death)	t_Ax_(mt, x, defer=t, i, g=0, method)
$A^1_{x:\overline{n}\|}$	term life insurance (end of the year)	nAx(mt, x, n, i, g=0, method)
$\bar{A}^1_{x:\overline{n}\|}$	term life insurance (moment of death)	nAx_(mt, x, n, i, g=0, method)
${}_{t\|}A^1_{x:\overline{n}\|}$	t years deferred term life insurance (end of the year)	t_nAx(mt, x, n, defer=t, i, g=0, method)
${}_{t\|}\bar{A}^1_{x:\overline{n}\|}$	t years deferred term life insurance (moment of death)	t_nAx_(mt, x, n, defer=t, i, g=0, method)
$A_{x:\overline{n}\|}$	endowment insurance (end of the year)	nAEx(mt, x, n, i, g=0, method)
$\bar{A}_{x:\overline{n}\|}$	endowment insurance (moment of death)	nAEx_(mt, x, n, i, g=0, method)
${}_{t\|}A_{x:\overline{n}\|}$	t-years deferred endowment insurance (end of the year)	t_nAEx(mt, x, n, defer=t, i, g=0, method)
${}_{t\|}\bar{A}_{x:\overline{n}\|}$	t-years deferred endowment insurance (moment of death)	t_nAEx_(mt, x, n, defer=t, i, g=0, method)

Remarks:

- For life insurance with terms varying geometrically, the Actuarial Table must be built with a growth rate g and the functions from Table 6 are applied.
- For Life Insurances with capitals varying arithmetically, the set of available functions are resumed in Tables 7 and 8.
- For negative values of "inc" parameter, provided that the capitals are positive, the functions in Tables 7 and 8 may reflect decreasing capital insurance.

Table 7: Actuarial Notation and Syntax Formula for Life Insurances - variable capitals (First Capital $\neq$ Increase Amount)

Notation	Description	Syntax
${}_{t\|}(IA)^{r}_{x:\overline{n}\|}$	t-years deferred term life insurance with capitals evolving arithmetically (increasing or decreasing). First capital 1 and increase amount r (end of the year)	t_nIArx(mt, x, n, i, defer=t, first=1, inc=r, method)
${}_{t\|}(I\bar{A})^{r}_{x:\overline{n}\|}$	t-years deferred term life insurance with capitals evolving arithmetically (increasing or decreasing). First capital 1 and increase amount r (moment of death)	t_nIArx_(mt, x, n, i, defer=t, first=1, inc=r, method)

Table 8: Actuarial Notation and Syntax Formula for Life Insurances - variable capitals (First Capital=Increase Amount)

Notation	Description	Syntax
$(IA)_x$	whole life insurance with arithmetically varying capitals (end of the year)	IAx(mt, x, i, inc=1, method)
$(I\bar{A})_x$	whole life insurance with arithmetically varying capitals (moment of death)	IAx_(mt, x, i, inc=1, method)
${}_{t\|}(IA)_x$	t years deferred whole life insurance with arithmetically varying capitals (end of the year)	t_IAx(mt, x, defer=t, i, inc=1, method)
${}_{t\|}(I\bar{A})_x$	t years deferred whole life insurance with arithmetically varying capitals (moment of death)	t_IAx_(mt, x, defer=t, i, inc=1, method)
$(IA)^{1}_{x:\overline{n}\|}$	term life insurance with arithmetically varying capitals (end of the year)	nIAx(mt, x, n, i, inc=1, method)
$(I\bar{A})^{1}_{x:\overline{n}\|}$	term life insurance with arithmetically varying capitals (end of the year)	nIAx_(mt, x, n, i, inc=1, method)
${}_{t\|}(IA)^{1}_{x:\overline{n}\|}$	t-years deferred term life insurance with capitals evolving arithmetically (end of the year)	t_nIArx(mt, x, n, defer=t, i, inc=1, method)
${}_{t\|}(I\bar{A})^{1}_{x:\overline{n}\|}$	t-years deferred term life insurance with capitals evolving arithmetically (end of the year)	t_nIArx_(mt, x, n, defer=t, i, inc=1, method)
$(IA)_{x:\overline{n}\|}$	endowment life insurance with capitals evolving arithmetically (end of the year)	nIAErx(mt, x, n, i, inc=1, method)
$(I\bar{A})_{x:\overline{n}\|}$	endowment life insurance with capitals evolving arithmetically (moment of death)	nIAErx_(mt, x, n, i, inc=1, method)
${}_{t\|}(IA)_{x:\overline{n}\|}$	t-years deferred endowment life insurance with capitals evolving arithmetically (end of the year)	t_nIAErx(mt, x, n, defer=t, i, inc=1, method)
${}_{t\|}(I\bar{A})_{x:\overline{n}\|}$	t-years deferred endowment life insurance with capitals evolving arithmetically (moment of death)	t_nIAErx_(mt, x, n, defer=t, i, inc=1, method)

3.5 Life Annuities for groups of two lives

3.5.1 Joint-Life Annuities

Table 9: Actuarial Notation/Syntax Formula for Joint-life annuities (yearly constant payments)

Notation	Description	Syntax
a_{xy}	whole life annuity	axy(mtx, mty, x, y, i, g=0, m=1, status='joint-life')
$\ddot{a}_{xy}$	whole life annuity due	aaxy(mtx, mty, x, y, i, g=0, m=1, status='joint-life')
${}_{t\vert}a_{xy}$	t years deferred whole life annuity	t_axy(mtx, mty, x, y, i, g=0, m=1, defer=t, status='joint-life')
${}_{t\vert}\ddot{a}_{xy}$	t years deferred whole life annuity due	t_aaxy(mtx, mty, x, y, i, g=0, m=1, defer=t, status='joint-life')
$a_{xy:\overline{n}\vert}$	n years temporary life annuity	naxy(mtx, mty, x, y, n, i, g=0, m=1, status='joint-life')
$\ddot{a}_{xy:\overline{n}\vert}$	n years temporary life annuity due	naaxy(mtx, mty, x, y, n, i, g=0, m=1, status='joint-life')
${}_{t\vert}a_{xy:\overline{n}\vert}$	t years deferred temporary life annuity	t_axy(mtx, mty, x, y, n, i, g=0, m=1, defer=t, status='joint-life')
${}_{t\vert}\ddot{a}_{xy:\overline{n}\vert}$	t years deferred temporary life annuity due	t_aaxy(mtx, mty, x, y, n, i, g=0, m=1, defer=t, status='joint-life')

For joint-life annuities, payable m times per year, functions on Table 9 also apply, by defining an integer $m > 1$.

Table 10: Actuarial Notation/Syntax Formula for Joint-life annuities payable m times per year (constant payments and integer ages)

Notation	Description	Syntax
$a_{xy}^{(m)}$	whole life annuity payable m times per year	axy(mtx, mty, x, y, i, g=0, m, status='joint-life')
$\ddot{a}_{xy}^{(m)}$	whole life annuity due payable m times per year	aaxy(mtx, mty, x, y, i, g=0, m, status='joint-life')
${}_{t\vert}a_{xy}^{(m)}$	t years deferred whole life annuity payable m times per year	t_axy(mtx, mty, x, y, i, g=0, m, defer=t, status='joint-life')
${}_{t\vert}\ddot{a}_{xy}^{(m)}$	t years deferred whole life annuity due payable m times per year	t_aaxy(mtx, mty, x, y, i, g=0, m, defer=t, status='joint-life')
$a_{xy:\overline{n}\vert}^{(m)}$	n years temporary life annuity payable m times per year	naxy(mtx, mty, x, y, n, i, g=0, m, status='joint-life')
$\ddot{a}_{xy:\overline{n}\vert}^{(m)}$	n years temporary life annuity due payable m times per year	naaxy(mtx, mty, x, y, n, i, g=0, m, status='joint-life')
${}_{t\vert}a_{xy:\overline{n}\vert}^{(m)}$	t years deferred temporary life annuity payable m times per year	t_axy(mtx, mty, x, y, n, i, g=0, m, defer=t, status='joint-life')
${}_{t\vert}\ddot{a}_{xy:\overline{n}\vert}^{(m)}$	t years deferred temporary life annuity due payable m times per year	t_aaxy(mtx, mty, x, y, n, i, g=0, m, defer=t, status='joint-life')

For annuities payable m times per year, with geometric growth g, a $g \neq 0$ and an integer $m > 1$ should be considered. Table 11 resumes the functions, for general m and g.

Table 11: Actuarial Notation/Syntax Formula for Joint-life annuities payable m times per year (geometric growth payments and integer ages). Payments change in each payment period.

Notation	**Description**	**Syntax**
$(Ga^{(m)})^{(m)}_{xy}$	whole life annuity payable m times per year. Payments change in each payment period.	axy(mtx, mty, x, y, i, g, m, status='joint-life')
$(G\ddot{a}^{(m)})^{(m)}_{xy}$	whole life annuity due payable m times per year. Payments change in each payment period.	aaxy(mtx, mty, x, y, i, g, m, status='joint-life')
${}_{t\vert}(Ga^{(m)})^{(m)}_{xy}$	t years deferred whole life annuity payable m times per year. Payments change in each payment period.	t_axy(mtx, mty, x, y, i, g, m, defer=t, status='joint-life')
${}_{t\vert}(G\ddot{a}^{(m)})^{(m)}_{xy}$	t years deferred whole life annuity due payable m times per year. Payments change in each payment period.	t_aaxy(mtx, mty, x, y, i, g, m, defer=t, status='joint-life')
$(Ga^{(m)})^{(m)}_{xy:\overline{n}\vert}$	n years temporary life annuity payable m times per year. Payments change in each payment period.	naxy(mtx, mty, x, y, n, i, g, m, status='joint-life')
$(G\ddot{a}^{(m)})^{(m)}_{xy:\overline{n}\vert}$	n years temporary life annuity due payable m times per year. Payments change in each payment period.	naaxy(mtx, mty, x, y, n, i, g, m, status='joint-life')
${}_{t\vert}(Ga^{(m)})^{(m)}_{xy:\overline{n}\vert}$	t years deferred temporary life annuity payable m times per year. Payments change in each payment period.	t_axy(mtx, mty, x, y, n, i, g, m, defer=t, status='joint-life')
${}_{t\vert}(G\ddot{a}^{(m)})^{(m)}_{xy:\overline{n}\vert}$	t years deferred temporary life annuity due payable m times per year. Payments change in each payment period.	t_aaxy(mtx, mty, x, y, n, i, g, m, defer=t, status='joint-life')

Remark on non-integer ages, terms and deferment periods:
All functions of section 3.5 are prepared for computing annuities for non-integer ages, terms and deferment periods. The usual approximations *Uniform Distribution of Death*, *Constant Force of Mortality* and *Balducci Aproximation* are available. The user should include the choice of the method in the function signature. For details, see Chapter 6.

3.5.2 Last-Survivor Annuities

All the functions presented in Tables 9, 10 and 11 are available in the package for *last-survivor* groups. The user should only change the **status** to *'last-survivor'*, in order to compute the expected value of the annuities with the correspondent probabilities.
All the actuarial notations should be updated to $\overline{xy}$ instead of xy.

3.6 Life Insurance for groups of two lives

3.6.1 Indemnities paid in the end of periods

The following tables present the syntax for Life Insurance contracts for joint-life groups of two individuals. For *last survivor* groups, the xy in the actuarial notation is replaced by $\overline{xy}$ and in the Python syntax, 'joint-life' should be replaced by 'last-survivor'.
For non-integer ages, terms and deferment periods, the *method* must be included and defined in the signature.

Table 12: Actuarial Notation and Syntax Formula for Life Insurance in groups of two individuals (constant capital, payments in the end of the periods)

Notation	Description	Syntax
${}_nE_{xy}$	pure endowment	nExy(mtx, mty, x, y, i, n, status='joint-life')
A_{xy}	whole life insurance	Axy(mtx, mty, x, y, i, g=0, m=1, status='joint-life')
${}_{t\|}A_{xy}$	t years deferred whole life insurance	t_Axy(mtx, mty, x, y, i, g=0, m=1, defer=t, status='joint-life')
$A_{xy:\overline{n}\|}$	n years term life insurance	nAxy(mtx, mty, x, y, n, i, g=0, m=1, status='joint-life')
${}_{t\|}A_{xy:\overline{n}\|}$	t years deferred term life insurance	t_Axy(mtx, mty, x, y, n, i, g=0, m=1, defer=t, status='joint-life')

For fractional life insurance (assume that payments may occur the end of the m-th period of the year, an $m > 1$ should be defined.

Table 13: Actuarial Notation and Syntax Formula for Life Insurance in groups of two individuals (geometrically increasing/decreasing capital, payments in the end of the periods)

Notation	Description	Syntax
$(GA)_{xy}$	whole life insurance	Axy(mtx, mty, x, y, i, g, m=1, status='joint-life')
${}_{t\|}(GA)_{xy}$	t years deferred whole life insurance	t_Axy(mtx, mty, x, y, i, g, m=1, defer=t, status='joint-life')
$(GA)_{xy:\overline{n}\|}$	n years term life insurance	nAxy(mtx, mty, x, y, n, i, g, m=1, status='joint-life')
${}_{t\|}(GA)_{xy:\overline{n}\|}$	t years deferred term life insurance	t_Axy(mtx, mty, x, y, n, i, g, m=1, defer=t, status='joint-life')

For fractional life insurance (assume that payments may occur the end of the m-th period of the year, an $m > 1$ should be defined. For fractional ages, terms and deferrement periods, *method* should be included and defined in the end of the signature.
For **Endowment Insurance**, user can use AExy instead of Axy in all functions included in Tables 12 and 13.
For the cases where first_is different from the rate of increment, both parameters should be defined with the adequate amounts.

Table 14: Actuarial Notation and Syntax Formula for Life Insurance in groups of two individuals (arithmetically increasing/decreasing capital, payments in the end of the periods)

Notation	**Description**	**Syntax**
$(IA)_{xy}$	whole life insurance	t_nIArxy(mtx, mty, x, y, n=ω-x, defer=0, first_payment=1, inc=1, status='joint-life')
$_{t\|}(IA)_{xy}$	t years deferred whole life insurance	t_nIArxy(mtx, mty, x, y, n=ω-(x+t), defer=t, first_payment=1, inc=1, status='joint-life')
$(IA)_{xy:\overline{n}\|}$	n years term life insurance	t_nIArxy(mtx, mty, x, y, n, defer=0, first_payment=1, inc=1, status='joint-life')
$_{t\|}(IA)_{xy:\overline{n}\|}$	t years deferred term life insurance	t_nIArxy(mtx, mty, x, y, n, defer=t, first_payment=1, inc=1, status='joint-life')

3.6.2 Indemnities paid in the "moment of death"

If the evaluation is performed considering payments in the moment of death, all the functions in tables 12, 13 and 14 are replaved by similar functions with syntax Axy_, t_Axy_, nAxy_, t_nIArxy_, respectively.

3.7 Financial Annuities

Table 15: Actuarial Notation and Syntax Formula for Financial Annuities

Notation	Description	Syntax
$a_{\overline{n}\vert}$	n-year immediate financial annuity	an(terms=n)
$\ddot{a}_{\overline{n}\vert}$	n-year due financial annuity	aan(terms=n)
a_{∞}	perpetual immediate financial annuity	a(terms=None)
$\ddot{a}_{\infty}$	perpetual due financial annuity	aa(terms=None)
${}_r(Ia)^{(m)}_{\overline{n}\vert}$	n-year immediate financial annuity with first payment 1 and evolving arithmetically (increasing $[r > 0]$ or decreasing $[r < 0]$). Payment increases in each period of the interest rate.	Ian(terms=n, payment=1, increase=r)
${}_r(I\ddot{a})^{(m)}_{\overline{n}\vert}$	n-year due financial annuity with first payment 1 and evolving arithmetically (increasing $[r > 0]$ or decreasing $[r < 0]$). Payment increases in each period of the interest rate.	Iaan(terms=n, payment=1, increase=r)
${}_r(I^{(m)}a)^{(m)}_{\overline{n}\vert}$	n-year immediate financial annuity with first payment 1 and evolving arithmetically (increasing $[r > 0]$ or decreasing $[r < 0]$). Payments increase in each payment period.	Iman(terms=n, payment=1, increase=r)
${}_r(I^{(m)}\ddot{a})^{(m)}_{\overline{n}\vert}$	n-year due financial annuity with first payment 1 and evolving arithmetically (increasing $[r > 0]$ or decreasing $[r < 0]$). Payments increase in each payment period.	Imaan(terms=n, payment=1, increase=r)
${}_g(Ga)^{(m)}_{\overline{n}\vert}$	n-year immediate financial annuity with first payment 1 and evolving geometrically with rate g. Payments change in each period of the interest rate.	Gan(terms=n, payment=1, grow=g)
${}_g(G\ddot{a})^{(m)}_{\overline{n}\vert}$	n-year due financial annuity with first payment 1 and evolving geometrically with rate g. Payments change in each period of the interest rate.	Gaan(terms=n, payment=1, grow=g)
${}_g(G^{(m)}a)^{(m)}_{\overline{n}\vert}$	n-year immediate financial annuity with first payment 1 and evolving geometrically with rate g. Payments change in each payment period.	Gman(terms=n, payment=1, grow=g)
${}_g(G^{(m)}\ddot{a})^{(m)}_{\overline{n}\vert}$	n-year due financial annuity with first payment 1 and evolving geometrically with rate g. Payments change in each payment period.	Gmaan(terms=n, payment=1, grow=g)

For annuities paid m times per year, the class Annuities Certain must be initiated with the correspondent frequency m. Examples are presented in Chapter 8.

4 Life Annuities

A life annuity corresponds to a series of payments paid as long as an individual is alive on the payment date. The life annuity can be temporary or payable for whole life, the payments are due in the beginning (annuity due) or at the end of the periods (annuity immediate) and starts immediately in the next period or after some delay (deferred annuities). The payments are constant through the all term of contract or are variable (with or without a mathematical regularity). The number of payments in each period of the interest rate may also be defined.

The computation of the present value of all these life annuities is available in the library, and are presented in this chapter. For a more general approach, the library includes a function, see subsection 4.3.6, which computes the present value of a given series of cash-flows, with a given set of interest rates and a defined set of probabilities. The developed functions allow for the computation of present values for integer and non-integer ages and terms. If using the *CommutationTable* or *CommutationTableFrac* classes, defined in section 7, all the life annuities presented in this chapter may also be computed by means of a Commutation Table, for either integer and fractional ages and terms. This approach is adequate for academic purposes, when commutation tables are used.

For using Life Annuities functions, the following code must be initiated if choosing, for instance, TV7377 SOA Table as the mortality table to adopt.

```
from soa_tables import read_soa_table_xml as rst
from lifeActuary import mortality_table as mt
from lifeActuary import annuities as la

# reads soa table TV7377
soa = rst.SoaTable('soa_tables/' + 'TV7377' + '.xml')

# creates a mortality table
tv7377 = mt.MortalityTable(data_type='q', mt=soa.table_qx, perc=100, last_q=1)
```

4.1 Whole Life Annuities

4.1.1 ax

Actuarial Notation: a_x and $a_x^{(m)}$

Usage

```
ax(mt, x, i=None, g=0, m=1, method='udd')
```

Description: Returns the actuarial present value of a whole life annuity of 1 per time period. Payments of $1/m$ are made m times per year at the end of the periods. If g≠0, payments increase by $(1+g/100)$ each period.

Parameters

mt	mortality table
x	age at the beginning of the contract
i	interest rate, in percentage (e.g. 2 for 2%
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1% $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant pauments
m	number of payments in each period of the interest rate
method	approximation method for non-integer ages, periods and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation*.

Examples

```
la.ax(mt=tv7377, x=50, i=2, g=0, m=1)                 # 21.554432773700235
la.ax(mt=tv7377, x=50, i=2, g=0, m=4, method='udd')   # 21.927012923715320

la.ax(mt=tv7377, x=50.5, i=2, g=0, m=1, method='udd') # 21.31196504242326
la.ax(mt=tv7377, x=50.5, i=2, g=0, m=1, method='cfm') # 21.30528881312939
la.ax(mt=tv7377, x=50.5, i=2, g=0, m=1, method='bal') # 21.29867410830813
```

4.1.2 aax

Actuarial Notation: $\ddot{a}_x$ and $\ddot{a}_x^{(m)}$

Usage

```
aax(mt, x, i=None, g=0, m=1, method='udd')
```

Description: Returns the actuarial present value of a whole life annuity due of 1 per time period. The payments of $1/m$ are made m times per year at the beginning of the periods. If g≠0, payments increase by (1+g/100) each period.

Parameters

mt	mortality table
x	age at the beginning of the contract
i	interest rate, in percentage (e.g. 2 for 2%
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1% $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant pauments
m	number of payments in each period of the interest rate
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation*.

Examples

```
la.aax(mt=tv7377, x=50, i=2, g=0, m=1)                  # 22.55443277370024
la.aax(mt=tv7377, x=50, i=2, g=0, m=4, method='udd')    # 22.17701292371532

la.aax(mt=tv7377, x=50.5, i=2, g=0, m=1, method='udd')  # 22.31196504242326
la.aax(mt=tv7377, x=50.5, i=2, g=0, m=1, method='cfm')  # 22.30528881312939
la.aax(mt=tv7377, x=50.5, i=2, g=0, m=1, method='bal')  # 22.298674108308134
```

4.1.3 t_ax

Actuarial Notation: ${}_{t|}a_x$ and ${}_{t|}a_x^{(m)}$

Usage

```
t_ax(mt, x, i=None, g=0, m=1, defer=0, method='udd')
```

Description: Returns the actuarial present value of an immediate whole life annuity of 1 per time period, deferred t periods. The payments of $1/m$ are made m times per year at the end of the periods. If g≠0, payments increase by (1+g/100) each period.

Parameters

mt	mortality table
x	age at the beginning of the contract
i	interest rate, in percentage (e.g. 2 for 2%
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1% $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant pauments
m	number of payments in each period of the interest rate
defer	number of deferment years
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation*.

Observation: t_ax(mt, x, i, g, m, defer=0, method) = ax(mt, x, i, g, m, method)

Examples

```
la.t_ax(mt=tv7377, x=50, i=2, g=0, m=1, defer=5)                 # 16.89919659176826
la.t_ax(mt=tv7377, x=50, i=2, g=0, m=4, defer=5, method='udd')     # 17.229163322375726

la.t_ax(mt=tv7377, x=50.5, i=2, g=0, m=1, defer=5, method='udd')   # 16.65908585991419
la.t_ax(mt=tv7377, x=50.5, i=3, g=0, m=1, defer=5, method='udd')   # 14.021600722805644
la.t_ax(mt=tv7377, x=50.5, i=2, g=1, m=1, defer=5, method='udd')   # 18.855665526541156
```

4.1.4 t_aax

Actuarial Notation: ${}_{t|}\ddot{a}_x$ and ${}_{t|}\ddot{a}_x^{(m)}$

Usage

```
t_aax(mt, x, i=None, g=0, m=1, defer=0, method='udd')
```

Description: Returns the actuarial present value of a whole life annuity due of 1 per time period, deferred t periods. The payments of $1/m$ are made m times per year at the beginning of the periods. If g≠0, payments increase by (1+g/100) each period.

Parameters

mt	mortality table
x	age at the beginning of the contract
i	interest rate, in percentage (e.g. 2 for 2%
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%
	$g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant pauments
m	number of payments in each period of the interest rate
defer	number of deferment years
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation*.

Observation: t_aax(mt, x, i, g, m, defer=0, method) = aax(mt, x, i, g, m, method)

Examples

```
la.t_aax(mt=tv7377, x=50, i=2, g=0, m=1, defer=5)               #17.78500355792074
la.t_aax(mt=tv7377, x=50, i=2, g=0, m=4, defer=5, method='udd') # 17.450615063913848

la.t_aax(mt=tv7377, x=50.5, i=2, g=0, m=1, defer=5, method='udd') # 17.544107552895813
la.t_aax(mt=tv7377, x=50.5, i=3, g=0, m=1, defer=5, method='udd') # 14.864486355546632
la.t_aax(mt=tv7377, x=50.5, i=2, g=1, m=1, defer=5, method='udd') # 19.929243874788195
```

4.2 Temporary Life Annuities

4.2.1 nax

Actuarial Notation: $a_{x:\overline{n}|}$ and $a^{(m)}_{x:\overline{n}|}$

Usage

```
nax(mt, x, n, i=None, g=0, m=1, method='udd')
```

Description: Returns the actuarial present value of an immediate n term life annuity of 1 per time period. The payments of $1/m$ are made m times per year at the end of the periods. If g≠0, payments increase by (1+g/100) each period.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%
	$g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant pauments
m	number of payments in each period of the interest rate
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation*.

Examples

```
la.nax(mt=tv7377, x=50, n=10, i=2, g=0, m=1)                # 8.756215803256637
la.nax(mt=tv7377, x=50, n=10, i=2, g=0, m=2, method='udd') # 8.811587860311260
la.nax(mt=tv7377, x=50, n=10, i=2, g=0, m=2, method='cfm') # 8.811571464621458
```

4.2.2 naax

Actuarial Notation: $\ddot{a}_{x:\overline{n}|}$ and $\ddot{a}^{(m)}_{x:\overline{n}|}$

Usage

```
naax(mt, x, n, i=None, g=0, m=1, method='udd')
```

Description: Returns the actuarial present value of a n term life annuity due of 1 per time period. The payments of $1/m$ are made m times per year at the beginning of the periods. If g≠0, payments increase by (1+g/100) each period.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%
	$g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant pauments
m	number of payments in each period of the interest rate
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*,
	'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation*.

Examples

```
la.nax(mt=tv7377, x=50, n=10, i=2, g=0, m=1)                  # 8.756215803256637
la.nax(mt=tv7377, x=50, n=10, i=2, g=0, m=2)                  # 8.81158786031126
la.nax(mt=tv7377, x=50, n=10, i=2, g=0, m=2, method='cfm')    # 8.811571464621458
```

4.2.3 t_nax

Actuarial Notation: ${}_{t|}a_{x:\overline{n}|}$ and ${}_{t|}a^{(m)}_{x:\overline{n}|}$

Usage

```
t_nax(mt, x, n, i=None, g=0, m=1, defer=0, method='udd')
```

Description: Returns the actuarial present value of an immediate n term life annuity of 1 per time period, deferred t periods. The payments of $1/m$ are made m times per year at the end of the periods. If g≠0, payments increase by (1+g/100) each period.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1% $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant paument
m	number of payments in each period of the interest rate
defer	number of deferment years
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation*.

Observation: t_nax(mt, x, n, i, g, m, defer=0, method) = nax(mt, x, n, i, g, m, method)

Examples

```
la.t_nax(mt=tv7377, x=50, n=10, i=2, g=0, m=1, defer=2)                # 8.316881544013759
la.t_nax(mt=tv7377, x=50, n=10, i=2, g=0, m=2, defer=1.5)              # 8.480554177218124
la.t_nax(mt=tv7377, x=50, n=10, i=2, g=0, m=2, defer=1.5, method='cfm') # 8.480533451243083
```

4.2.4 t_naax

Actuarial Notation: ${}_{t|}\ddot{a}_{x:\overline{n}|}$ and ${}_{t|}\ddot{a}^{(m)}_{x:\overline{n}|}$

Usage

```
t_naax(mt, x, n, i=None, g=0, m=1, defer=0, method='udd')
```

Description: Returns the actuarial present value of a n term life annuity due of 1 per time period, deferred t periods. The payments of $1/m$ are made m times per year at the beginning of the periods. If g≠0, payments increase by (1+g/100) each period.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1% $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
m	number of payments in each period of the interest rate
defer	number of deferment years
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation*.

Observation: t_naax(mt, x, n, i, g, m, defer=0, method) = naax(mt, x, n, i, g, m, method)

Examples

```
la.t_naax(mt=tv7377, x=50, n=10, i=2, g=0, m=1, defer=2)                # 8.535558101895862
la.t_naax(mt=tv7377, x=50, n=10, i=2, g=0, m=2, defer=1.5)              # 8.590388221834296
la.t_naax(mt=tv7377, x=50, n=10, i=2, g=0, m=2, defer=1.5, method='bal') # 8.590351413627872
```

4.3 Life Annuities with variable terms

In this section, are presented functions that compute the actuarial present value of life annuities whose payments are not constant overtime.
As special regularities, life annuities with terms evolving in arithmetic or geometric progression (increasing or decreasing) are well known and easily computed and this library presents easy to use solutions and functions.
As a more general case, see section 4.3.6, the library also allows for the computation of the actuarial present value for a given set of cash-flows, interest rates and survival probabilities, which can vary without any regularity. The set of parameters should be provided in vector formats.

4.3.1 nIax

Actuarial Notation: $(Ia)_{x:\overline{n}|}$, $(Ia)^{(m)}_{x:\overline{n}|}$ and $(Da)_{x:\overline{n}|}$, $(Da)^{(m)}_{x:\overline{n}|}$

Usage

```
nIax(mt, x, n, i=None, m=1, first_amount=1, increase_amount=1, method='udd')
```

Description: Returns the actuarial present value of an immediate n term life annuity with payments evolving in arithmetic progression. Payments of $1/m$ are made m times per year at the end of the periods. First amount and Increase amount may be different. For decreasing life annuities, the Increase Amount should be negative.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
m	number of payments in each period of the interest rate
first_amount	amount of the first payment
increase_amount	amount of the increase rate increasing life annuities: increase_amount > 0 decreasing life annuities: increase_amount < 0
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation.*

Observation: nIax(mt, x, n, i, m, first_amount=1, increase_amount=0, method) = nax(mt, x, n, i, m, first_amount=1, increase_amount=0, method)

Examples

```python
la.nIax(mt=tv7377, x=50, n=10, i=2, m=1)
# 46.330171698412386
la.nIax(mt=tv7377, x=50, n=10, i=2, m=1, first_amount=1, increase_amount=2)
# 83.90412759356813
la.nIax(mt=tv7377, x=50, n=10, i=2, m=1, first_amount=100, increase_amount=-2)
# 800.4736685353522
la.nIax(mt=tv7377, x=50, n=10, i=2, m=1, first_amount=1, increase_amount=2)
# 83.90412759356813
la.nIax(mt=tv7377, x=50.3, n=10, i=2, m=4, first_amount=1, increase_amount=2, method='cfm')
# 84.66224090334902
```

4.3.2 nIaax

Actuarial Notation: $(I\ddot{a})_{x:\overline{n}|}$, $(I\ddot{a})^{(m)}_{x:\overline{n}|}$ and $(D\ddot{a})_{x:\overline{n}|}$, $(D\ddot{a})^{(m)}_{x:\overline{n}|}$

Usage

```python
nIaax(mt, x, n, i=None, m=1, first_amount=1, increase_amount=1, method='udd')
```

Description: Returns the actuarial present value of a due n term life annuity with payments evolving in arithmetic progression. Payments of $1/m$ are made m times per year at the beginning of the periods. First amount and Increase amount may be different. For decreasing life annuities, the Increase Amount should be negative.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of periods of the contract (measured in periods of the interest rate)
i	interest rate, in percentage (e.g. 2 for 2%)
m	number of payments in each period of the interest rate
defer	number of deferment years
first_amount	amount of the first payment
increase_amount	amount of the increase amount increasing life annuities: increase_amount > 0 decreasing life annuities: increase_amount < 0
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation.*

Observation: nIaax(mt, x, n, i, m, first_amount=1, increase_amount=0, method) = naax(mt, x, n, i, m, first_amount=1, increase_amount=0, method)

Examples

```
la.nIaax(mt=tv7377, x=50, n=10, i=2, m=1)
# 47.53746439543621
la.nIaax(mt=tv7377, x=50, n=10, i=2, m=1, first_amount=1, increase_amount=2)
# 86.09588781545513
la.nIaax(mt=tv7377, x=50, n=10, i=2, m=1, first_amount=100, increase_amount=-2)
# 820.787250701691
la.nIaax(mt=tv7377, x=50, n=10, i=2, m=1, first_amount=1, increase_amount=2)
# 86.09588781545513
la.nIaax(mt=tv7377, x=50.3, n=10, i=2, m=4, first_amount=1, increase_amount=2, method='cfm')
# 85.21250336665355
```

4.3.3 t_nIax

Actuarial Notation: ${}_{t|}(Ia)_{x:\overline{n}|}$, ${}_{t|}(Ia)^{(m)}_{x:\overline{n}|}$ and $(Da)_{x:\overline{n}|}$, ${}_{t|}(Da)^{(m)}_{x:\overline{n}|}$

Usage

```
t_nIax(mt, x, n, i=None, m=1, defer=0, first_amount=1, increase_amount=1, method='udd')
```

Description: Returns the actuarial present value of an immediate n term life annuity, deferred t periods, with payments evolving in arithmetic progression. Payments of $1/m$ are made m times per year at the end of the periods. First amount and Increase amount may be different. For decreasing life annuities, the Increase Amount should be negative.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of periods of the contract (measured in periods of the interest rate)
i	interest rate, in percentage (e.g. 2 for 2%)
m	number of payments in each period of the interest rate
defer	number of deferment years
first_amount	amount of the first payment
increase_amount	amount of the increase amount increasing life annuities: increase_amount > 0 decreasing life annuities: increase_amount < 0
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation.*

Observation: t_nIax(mt, x, n, i, m, defer, first_amount=1, increase_amount=0, method) = t_nax(mt, x, n, i, m, defer, method)

Examples

```
la.t_nIax(mt=tv7377, x=50, n=10, i=2, m=1, defer=2)
# 45.89083743916951
la.t_nIax(mt=tv7377, x=50, n=10, i=2, m=1, defer=2, first_amount=1, increase_amount=2)
# 83.46479333432525
la.t_nIax(mt=tv7377, x=50, n=10, i=2, m=1, defer=10, first_amount=100, increase_amount=-2)
# 585.2087875846838
la.t_nIax(mt=tv7377, x=50, n=10, i=2, m=1, defer=1, first_amount=1, increase_amount=2)
# 83.68346989220736
la.t_nIax(mt=tv7377, x=50.3, n=10, i=2, m=4, defer=1, first_amount=1, increase_amount=2,
    method='cfm')  # 84.43983387860666
```

4.3.4 t_nIaax

Actuarial Notation: ${}_{t|}(I\ddot{a})_{x:\overline{n}|}$, ${}_{t|}(I\ddot{a})^{(m)}_{x:\overline{n}|}$ and ${}_{t|}(D\ddot{a})_{x:\overline{n}|}$, ${}_{t|}(D\ddot{a})^{(m)}_{x:\overline{n}|}$

Usage

```
t_nIaax(mt, x, n, i=None, m=1, defer=0, first_amount=1, increase_amount=1, method='udd')
```

Description: Returns the actuarial present value of a n term life annuity due, deferred t periods, with payments evolving in arithmetic progression. The payments are made m times per year at the beginning of the periods and the payment of each period is divided into m equal payments. First amount and Increase amount may be different. For decreasing life annuities, the *Increase Amount* should be negative.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
m	number of payments in each period of the interest rate
defer	number of deferment years
first_amount	amount of the first payment
increase_amount	amount of the increase amount increasing life annuities: increase_amount > 0 decreasing life annuities: increase_amount < 0
method	approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Aproximation.*

Observation: t_nIaax(mt, x, n, i, m, defer, first_amount=1, increase_amount=0, method) = t_naax(mt, x, n, i, m, defer, method)

Examples

```
la.t_nIaax(mt=tv7377, x=50, n=10, i=2, m=1, defer=2)
# 47.093981521914785
la.t_nIaax(mt=tv7377, x=50, n=10, i=2, m=1, defer=2, first_amount=1, increase_amount=2)
# 85.6524049419337
la.t_nIaax(mt=tv7377, x=50, n=10, i=2, m=1, defer=10, first_amount=100, increase_amount=-2)
# 606.6457012514408
la.t_nIaax(mt=tv7377, x=50, n=10, i=2, m=1, defer=1, first_amount=1, increase_amount=2)
# 604.6767662017144
la.t_nIaax(mt=tv7377, x=50.3, n=10, i=2, m=4, defer=1, first_amount=1, increase_amount=2,
    method='cfm')
#  84.98956500937922
```

4.3.5 Geometric Life Annuities

For life annuities with payments evolving in geometric progression (increasing or decreasing) the growth rate (g) should be included when computing the annuities functions presented in the previous sections.

4.3.6 Present_Value Function

This function generalizes any of the above, returning the present value of a series of cash-flows (introduced in vector mode), where the interest rate for each period may differ as well as the probability assigned to each payment/benefit. According to the defined probabilities, the function returns the present value (for probabilities equal to 1) or the actuarial present value of the series of cash-flows.

Usage

```
present_value(mt, age, spot_rates, capital, probs=None)
```

Description: This function computes the expected present value of a cash-flow, that can be contingent on some probabilities. The payments are considered at the end of the period.

Parameters

mt	mortality table. If mt=None, probabilities must be defined in probs.
age	age at the beginning of the contract
spot_rates	vector of interest rates for the considered time periods. Use 1 for 1%.
capital	vector of cash-flow amounts
probs	vector of probabilities. For using the mortality table mt, use probs=None.

Examples

```
pv1 = la.present_value(mt=None, age=None, spot_rates=[1.2, 1.4, 1.8, 1.6, 1.9], capital=[100,
    -25, 120, 300, -50], probs=1)
print('Present Value:', pv1)
# 425.750701233034

pv2 = la.present_value(mt=tv7377, age=35, spot_rates=[1.2, 1.4, 1.8, 1.6, 1.9], capital=[100,
    -25, 120, 300, -50], probs=None)
print('Present Value:', pv2)
# 424.2408517830521
```

5 Pricing Life Insurance

When pricing life insurance, there is a need to compute the present value of the benefit payment, for a given mortality table and an interest rate.
This library includes functions that allow for pricing the most common contracts in life insurance, such as Pure Endowment, Whole Life and Temporary Insurances, Endowment Insurance as well as the traditional life insurance with increasing (or decreasing) capitals.
The available functions allow for the pricing of any other type of life insurance, whose actuarial evaluation makes use of the functions available in this library.

For using Life Insurance functions, the following code must be initiated if choosing, for instance, TV7377 SOA Table as the mortality table.

```
from soa_tables import read_soa_table_xml as rst
from lifeActuary import mortality_table as mt
from lifeActuary import annuities as la
from lifeActuary import mortality_insurance as lins

# reads soa table TV7377
soa = rst.SoaTable('soa_tables/' + 'TV7377' + '.xml')

# creates a mortality table
tv7377 = mt.MortalityTable(data_type='q', mt=soa.table_qx, perc=100, last_q=1)
```

5.1 Pure Endowment / Deferred Capital / Expected Present Value / nEx

Actuarial Notation: ${}_nE_x$

Usage

```
nEx(mt, x, i=None, g=0, n=0, method='udd')
```

Description: Returns the present value of a Pure Endowment of 1 for an aged x individual, paid at age $x+n$.

Parameters

mt	mortality table
x	age at the beginning of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
n	number of years of the contract
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
la.nEx(mt=tv7377, x=50, i=2, g=0, n=5)  # 0.8858069661524853
la.nEx(mt=tv7377, x=50, i=2, g=0, n=10) # 0.7771748278393478
la.nEx(mt=tv7377, x=80, i=2, g=0, n=10) # 0.2283081320230278
la.nEx(mt=tv7377, x=50.4, i=2, g=0, n=10.5, method='bal') # 0.7653132063796898
```

5.2 Whole Life Insurance

5.2.1 Ax

Actuarial Notation: A_x

Usage

```
Ax(mt, x, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a whole life insurance (i.e. net single premium), that pays 1 at the end of the year of death.

Parameters

mt	mortality table
x	age at the beginning of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%)
	$g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.Ax(mt=tv7377, x=50, i=2)     # 0.5577562201235239
lins.Ax(mt=tv7377, x=50.7, i=2)   # 0.5613776896906858
lins.Ax(mt=tv7377, x=50.7, i=2, method='cfm') # 0.5615358294624957
```

5.2.2 Ax_

Actuarial Notation: $\bar{A}_x$

Usage

```
Ax_(x)
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a whole life insurance (i.e. net single premium), that pays 1 at the moment of death.

Parameters

mt	mortality table
x	age at the beginning of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%)
	$g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.Ax_(mt=tv7377, x=50, i=2)    # 0.5633061699539693
lins.Ax_(mt=tv7377, x=50.7, i=2)  # 0.5669636749317376
lins.Ax_(mt=tv7377, x=50.7, i=2, method='cfm')  # 0.5671233882723721
```

5.2.3 t_Ax

Actuarial Notation: $_{t|}A_x$

Usage

```
t_Ax(mt, x, defer=0, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred whole life insurance (i.e. net single premium), that pays 1 at the end of year of death.

Parameters

mt	mortality table
x	age at the beginning of the contract
defer	deferment period (in years)
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Observation: t_Ax(mt, x, defer=0, i, g, method) = Ax(mt, x, i, g, method)

Examples

```
lins.t_Ax(mt=tv7377, x=50, defer=2, i=2) # 0.550183040772438
```

5.2.4 t_Ax_

Actuarial Notation: $_{t|}\bar{A}_x$

Usage

```
t_Ax_(mt, x, defer=0, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred whole life insurance (i.e. net single premium), that pays 1 at the moment of death.

Parameters

mt	mortality table
x	age at the beginning of the contract
defer	deferment period (in years)
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Observation: t_Ax_(mt, x, defer=0, i, g, method) = Ax_(x, x, i, g, method)

Examples

```
lins.t_Ax_(mt=tv7377, x=50, defer=2, i=2) # 0.5556576337284301
```

5.3 Term Life Insurance

5.3.1 nAx

Actuarial Notation: $A^{1}_{x:\overline{n}|}$

Usage

```
nAx(mt, x, n, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a term (temporary) life insurance (i.e. net single premium), that pays 1, at the end of the year of death.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.nAx(mt=tv7377, x=50, n=10, i=2)      # 0.04676554519168518
lins.nAx(mt=tv7377, x=50, n=10, i=2, g=3) # 0.054219259550225045
```

5.3.2 nAx_

Actuarial Notation: $\bar{A}^{1}_{x:\overline{n}|}$

Usage

```
nAx_(mt, x, n, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a term (temporary) life insurance (i.e. net single premium), that pays 1, at the moment of death.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.nAx_(mt=tv7377, x=50, n=10, i=2)      # 0.04723088546086194
lins.nAx_(mt=tv7377, x=50, n=10, i=2, g=3) # 0.05475876795818331
```

5.3.3 t_nAx

Actuarial Notation: ${}_{t|}A^{1}_{x:\overline{n}|}$

Usage

```
t_nAx(mt, x, n, defer=0, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred term (temporary) life insurance (i.e. net single premium), that pays 1, at the end of the year of death.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
defer	number of years of deferment
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Observation: t_nAx(mt, x, n, defer=0, i, g, method) = nAx(mt, x, n, i, g, method)

Examples

```
lins.t_nAx(mt=tv7377, x=50, n=10, defer=5, i=2, g=0)  # 0.059615329779335056
lins.t_nAx(mt=tv7377, x=50, n=10, defer=5, i=2, g=10) # 0.09883714561436167
```

5.3.4 t_nAx_

Actuarial Notation: ${}_{t|}\bar{A}^{1}_{x:\overline{n}|}$

Usage

```
t_nAx_(mt, x, n, defer=0, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred term (temporary) life insurance (i.e. net single premium), that pays 1, at the moment of death.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
defer	number of years of deferment
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Observation: t_nAx_(mt, x, n, defer=0, i, g, method) = nAx_(mt, x, n, i, g, method)

Examples

```
lins.t_nAx_(mt=tv7377, x=50, n=10, defer=5, i=2, g=0) # 0.0602085317508478 24
lins.t_nAx_(mt=tv7377, x=50, n=10, defer=5, i=2, g=10) # 0.09982062402258575
```

5.4 Endowment Insurance

5.4.1 nAEx

Actuarial Notation: $A_{x:\overline{n}|}$

Usage

```
nAEx(mt, x, n, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of an Endowment life insurance (i.e. net single premium), that pays 1, at the end of year of death or 1 if (x) survives to age $x+n$.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.nAEx(mt=tv7377, x=50, n=10, i=2)       # 0.823940373031033
lins.nAEx(mt=tv7377, x=50, n=10, i=2, g=12) # 0.8627337141815358
```

5.4.2 nAEx_

Actuarial Notation: $\bar{A}_{x:\overline{n}|}$

Usage

```
nAEx_(mt, x, n, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of an Endowment life insurance (i.e. net single premium), that pays 1, at the moment of death or 1 if (x) survives to age $x + n$.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.nAEx_(mt=tv7377, x=50, n=10, i=2)       # 0.8244057133002097
lins.nAEx_(mt=tv7377, x=50, n=10, i=2, g=12) # 0.8635850673527166

lins.nAEx_(mt=tv7377, x=50.25, n=10.75, i=2, g=12, method='udd') # 0.8552846772135826
lins.nAEx_(mt=tv7377, x=50.25, n=10.75, i=2, g=12, method='cfm') # 0.8552909163204725
lins.nAEx_(mt=tv7377, x=50.25, n=10.75, i=2, g=12, method='bal') # 0.8552971473777993
```

5.4.3 t_nAEx

Actuarial Notation: ${}_{t|}A_{x:\overline{n}|}$

Usage

```
t_nAEx(mt, x, n, defer=0, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred Endowment life insurance (i.e. net single premium) that pays 1, at the end of year of death or 1 if (x) survives to age $x + t + n$.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
defer	number of years of deferment
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.t_nAEx(mt=tv7377, x=50, n=10, defer=2, i=2)        # 0.786304068847034
lins.t_nAEx(mt=tv7377, x=50, n=10, defer=10, i=2, g=12) # 0.7133653742582529
```

5.4.4 t_AEx_

Actuarial Notation: ${}_{t|}\bar{A}_{x:\overline{n}|}$

Usage

```
t_nAEx_(mt, x, n, defer=0, i=None, g=0, method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred Endowment life insurance (i.e. net single premium) that pays 1, at the moment of death or 1 if (x) survives to age $x + t + n$.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
defer	number of years of deferment
i	interest rate, in percentage (e.g. 2 for 2%)
g	rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.t_nAEx_(mt=tv7377, x=50, n=10, defer=2, i=2)        # 0.7868146552887255
lins.t_nAEx_(mt=tv7377, x=50, n=10, defer=10, i=2, g=12) # 0.7148635174567862
```

5.5 Life Insurance with Capitals evolving in arithmetic progression

In this section we introduce functions that allow for the computation of the present value of life insurance contracts with capitals increasing/decreasing in arithmetic progression.
The common actuarial functions for increasing life insurance, where the capital of the first year is equal to the rate of the progression are developed (see sections 5.5.1 to 5.5.4), but the library also contains general functions (see section **??**) where the capital of first year may differ from the rate of progression. These last functions allow for the computation of life insurance with decreasing capital (in arithmetic progression).
In section **??**, the library also includes functions that compute the present value of Increasing/Decreasing Endowment Insurance.
As in the previous sections, the library contains functions that evaluate the liability of the contract, if payments are performed in the end of the year or in the "moment of death" (approximated by the the middle of the year). The syntax of each of these function follow the same rational of the previous sections. In this section, for simplicity, we present both approaches in the same sub-sections.

5.5.1 IAx and IAx_

Actuarial Notation: $(IA)_x$ and $(I\bar{A})_x$

Usage

```
IAx(mt, x, i=None, inc=1, method='udd')   # end of the year

IAx_(mt, x, i=None, inc=1, method='udd')  # moment of death
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a Whole Life Insurance (i.e. net single premium), that pays $1 + k$, at the end of year/moment of death, if death occurs between ages $x+k$ and $x+k+1$, for k=0, 1, ... The capital of the first year equals the rate of the progression.

Parameters

mt	mortality table
x	age at the beginning of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
inc	rate of the progression (in monetary units)
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.IAx(mt=tv7377, x=50, i=2, inc=1)   # 15.807431562003352
lins.IAx_(mt=tv7377, x=50, i=2, inc=1)  # 15.964723312327344

lins.IAx(mt=tv7377, x=50, i=2, inc=10)  # 158.0743156200335
lins.IAx(mt=tv7377, x=50, i=2, inc=0)   # 0 (first capital = increment = 0)
```

5.5.2 t_IAx and t_IAx_

Actuarial Notation: ${}_{t|}(IA)_x$ and ${}_{t|}(I\bar{A})_x$

Usage

```
t_IAx(mt, x, defer=0, i=None, inc=1, method='udd')   # end of the year

t_IAx_(mt, x, defer=0, i=None, inc=1, method='udd')  # moment of death
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred Whole Life Insurance (i.e. net single premium), that pays $1+k$, at the end of year/moment of death, if death occurs between ages $x+t+k$ and $x+t+k+1$, for k=0, 1, ... The capital of the first year equals the rate of the progression.

Parameters

mt	mortality table
x	age at the beginning of the contract
defer	number of years of deferment
i	interest rate, in percentage (e.g. 2 for 2%)
inc	rate of the progression (in monetary units)
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.t_IAx(mt=tv7377, x=50, defer=5, i=2, inc=1)  # 13.057686275247685
lins.t_IAx_(mt=tv7377, x=50, defer=5, i=2, inc=1) # 13.187616702044672
lins.t_IAx(mt=tv7377, x=50, defer=5, i=2, inc=10) # 130.57686275247684
```

5.5.3 nIAx and nIAx_

Actuarial Notation: $(IA)_{x:\overline{n}|}$ and $(I\bar{A})_{x:\overline{n}|}$

Usage

```
nIAx(mt, x, n, i=None, inc=1, method='udd')  # end of the year

nIAx_(mt, x, n, i=None, inc=1, method='udd') # moment of death
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of an arithmetically increasing Term Life Insurance (i.e. net single premium), that pays $1+k$, at the end of the year/moment of death if death happens between age $x+k$ and $x+k+1$, k=0,..., $n-1$.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
i	interest rate, in percentage (e.g. 2 for 2%)
inc	rate of the progression (in monetary units)
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.nIAx(mt=tv7377, x=50, n=10, i=2, inc=1)     # 0.2751855520152558
lins.nIAx_(mt=tv7377, x=50, n=10, i=2, inc=1)    # 0.27792378415439706

lins.nIAx(mt=tv7377, x=50, n=10, i=2, inc=10)  # 2.7518555201525583
```

5.5.4 t_nIAx and t_nIAx_

Actuarial Notation: ${}_{t|}(IA)_{x:\overline{n}|}$ and ${}_{t|}(I\bar{A})_{x:\overline{n}|}$

Usage

```
t_nIAx(mt, x, n, defer=0, i=None, inc=1, method='udd')   # end of the year

t_nIAx_(mt, x, n, defer=0, i=None, inc=1, method='udd')  # moment of death
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred Term Life Insurance (i.e. net single premium), that pays $1 + k$, at the end of year/moment of death, if death occurs between ages $x + t + k$ and $x + t + k + 1$, for k=0, 1, . . . , $n - 1$. The capital of the first year equals the rate of the progression.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
defer	number of years of deferment
i	interest rate, in percentage (e.g. 2 for 2%)
inc	rate of the progression (in monetary units)
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.t_nIAx(mt=tv7377, x=50, n=10, defer=5, i=2, inc=1)  # 0.3529086516825162
lins.t_nIAx_(mt=tv7377, x=50, n=10, defer=5, i=2, inc=1) # 0.35642026704582747
lins.t_nIAx(mt=tv7377, x=50, n=10, defer=5, i=2, inc=10) # 3.5290865168251617
```

5.5.5 t_nIArx and t_nIArx_

This function computes the actuarial present value of a term insurance whose capitals increase/decrease arithmetically, allowing for different amounts of initial capital and increase amount. It also corresponds to a generalization of functions nIAx and t_nIAx, of sections 5.5.3 and 5.5.4 which also allows for decreasing capitals.

Actuarial Notation: ${}_{t|}(IA)^{r}_{x:\overline{n}|}$, ${}_{t|}(I\bar{A})^{r}_{x:\overline{n}|}$ and ${}_{t|}(DA)^{r}_{x:\overline{n}|}$, ${}_{t|}(D\bar{A})^{r}_{x:\overline{n}|}$

Usage

```
t_nIArx(mt, x, n, defer=0, i=None, first_amount=1, inc=1, method='udd')  # end of the year

t_nIArx_(mt, x, n, defer=0, i=None, first_amount=1, inc=1, method='udd') # moment of death
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a term life insurance (i.e. net single premium), that pays (first_amount + $k\times$ increase_amount), at the end of the year/moment of death, if death occurs between ages $x + defer + k$ and $x + k + defer + 1$, for $k = 0, \ldots, n-1$. Allows the computation for decreasing capitals. The first capital may differ from the increasing/decreasing amount.

Parameters

mt	mortality table
x	age at the beginning of the contract
n	number of years of the contract
defer	number of deferment years
i	interest rate, in percentage (e.g. 2 for 2%)
first_amount	insured amount in the first year of the contract
inc	rate of increasing (if inc>0) or decreasing (if inc<0)
method	approximation method for non-integer ages and terms. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* or 'bal' for *Balducci Approximation*.

Examples

```
lins.t_nIArx(mt=tv7377, x=50, n=10, defer=0, i=2, first_amount=1, inc=1)
# 0.2751855520152558
lins.t_nIArx(mt=tv7377, x=50, n=10, defer=0, i=2, first_amount=1000, inc=50)
# 58.18654553286372
lins.t_nIArx(mt=tv7377, x=50, n=10, defer=10, i=2, first_amount=1000, inc=50)
# 101.10261167944806
lins.t_nIArx(mt=tv7377, x=50, n=10, defer=0, i=2, first_amount=1000, inc=-50)
# 35.34454485050665
lins.t_nIArx(mt=tv7377, x=50, n=10, defer=10, i=2, first_amount=1000, inc=-50)
# 60.26561732559179

lins.t_nIArx_(mt=tv7377, x=50, n=10, defer=0, i=2, first_amount=1, inc=1)
# 0.27792378415439706
lins.t_nIArx_(mt=tv7377, x=50, n=10, defer=0, i=2, first_amount=1000, inc=50)
# 58.765530395538704
lins.t_nIArx_(mt=tv7377, x=50, n=10, defer=10, i=2, first_amount=1000, inc=50)
# 102.10863259378893
lins.t_nIArx_(mt=tv7377, x=50, n=10, defer=0, i=2, first_amount=1000, inc=-50)
# 35.696240526185186
lins.t_nIArx_(mt=tv7377, x=50, n=10, defer=10, i=2, first_amount=1000, inc=-50)
# 60.865289979325354
```

Observations:

- t_nIArx(mt, x, n, i, defer=0, first=1, amount=1, inc=1, method) = nIAx(mt, x, n, i, method)
- t_nIArx_(mt, x, n, i, defer=0, first=1, amount=1, inc=1, method) = nIAx_(mt, x, n, i, method)

5.5.6 t_nIAErx and _nIAErx_

This function computes the actuarial present value of an endowment insurance with capital evolving arithmetically, allowing for different amounts of initial capital and increase/decrease amount.

Usage

```
t_nIAErx(mt, x, n, defer=0, i=None, first_amount=1, inc=1, method='udd')  # end of the year

t_nIAErx_(mt, x, n, defer=0, i=None, first_amount=1, inc=1, method='udd') # moment of death
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of an endowment life insurance (i.e. net single premium), that pays (first_amount + $k\times$ increase_amount), at the end of the year/moment of death if death happens between age $x+k$ and $x+k+1$, for $k=0,\ldots,n-1$ and a capital of first_amount + $(n-1)\times$ increase_amount in case of life at the end of the contract.

Parameters

x	age at the beginning of the contract
n	number of years of the contract
defer	number of deferment years
first_amount	insured amount in the first year of the contract
increase_amount	rate of increasing (if >0) or decreasing (if <0)

Examples

```
lins.t_nIAErx(mt=tv7377, x=50, n=10, defer=0, i=2, first_amount=1, inc=1)
# 8.046933830408733
lins.t_nIAErx(mt=tv7377, x=50, n=10, defer=0, i=2, first_amount=1000, inc=50)
# 1185.0900458999179
lins.t_nIAErx(mt=tv7377, x=50, n=10, defer=0, i=2, first_amount=1000, inc=-50)
# 462.7907001621479

lins.t_nIAErx_(mt=tv7377, x=50, n=10, defer=5, i=2, first_amount=1, inc=1)
# 7.072355164462292
lins.t_nIAErx_(mt=tv7377, x=50, n=10, defer=5, i=2, first_amount=1000, inc=50)
# 1048.8296786409842
lins.t_nIAErx_(mt=tv7377, x=50, n=10, defer=5, i=2, first_amount=1000, inc=-50)
# 414.7743643440044
```

6 Multiple Lives Contracts

In this section, we present the developed functions that allow for the evaluation of insurance contracts in which the benefits are dependent on the joint mortality of a group of lives. As commonly, we develop functions for groups of two lives, allowing for the computation of probabilities, annuities and insurance benefits when risk is evaluated under **joint life** and **last survivor** contingencies.

Focusing on a group of two lives aged x and y, and using the common actuarial notation for groups of two lives, we consider (x, y) and $(\overline{x, y})$ for the joint life group and last survivor group, respectively.

6.1 Survival Probability Functions

The package includes functions that allow for the computation of the standard probabilities ${}_np_{xy}$, ${}_np_{\overline{xy}}$, ${}_nq_{xy}$, ${}_nq_{\overline{xy}}$, ${}_{t|n}q_{xy}$, ${}_{t|n}q_{\overline{xy}}$ for both integer and non-integer ages, as well as life expectancy of the groups, represented by e_{xy} and $e_{\overline{xy}}$.
These new functions create a flexible framework for computing standard probabilities and actuarial expected present values of common contracts, allowing for dependence between lives. Moreover, mortality tables may be different for individuals (x) and (y).

In the examples of this chapter, the following mortality tables will be used:

```
from lifeActuary import mortality_table as mt
from lifeActuary import life_2heads as l2h
from soa_tables import read_soa_table_xml as rst

soa_TV7377 = rst.SoaTable('soa_tables/TV7377.xml')
soa_GRF95 = rst.SoaTable('soa_tables/GRF95.xml')
grf95 = mt.MortalityTable(mt=soa_GRF95.table_qx)
tv7377 = mt.MortalityTable(mt=soa_TV7377.table_qx)
```

6.1.1 npxy

Actuarial Notation: ${}_np_{xy}$ and ${}_np_{\overline{xy}}$

Usage

```
npxy(mtx,mty,x,y,n=1,status='joint-life',method='udd')
```

Description: Returns the probability of survival of a joint life or last survival group with ages (x, y).

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x) at the beginning of the contract
y	age of (y) at the beginning of the contract
n	number of years
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
l2h.npxy(mtx=grf95, mty=tv7377, x=25, y=28, n=10, status='joint-life')    # 0.9849888566208177
l2h.npxy(mtx=grf95, mty=tv7377, x=25, y=28, n=10, status='last-survivor') # 0.9999455887520334

l2h.npxy(mtx=grf95, mty=tv7377, x=20.5, y=50.75, n=10.25, status='joint-life', method='bal')
# 0.9382616738238869
l2h.npxy(mtx=grf95, mty=tv7377, x=20.5, y=50.75, n=10.25,status='last-survivor', method='bal')
# 0.9997296719615928
```

6.1.2 nqxy

Actuarial Notation: ${}_nq_{xy}$ and ${}_nq_{\overline{xy}}$

Usage

```
nqxy(mtx,mty,x,y,n,status='joint-life',method='udd')
```

Description: Returns the probability of extinction of a group of two individuals (x) and (y) for joint life or last survival methods, ie, (x, y) and $(\overline{x, y})$

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x) at the beginning of the contract
y	age of (y) at the beginning of the contract
n	number of years of the contract
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
l2h.nqxy(mtx=grf95, mty=tv7377, x=25, y=28, n=10, status='joint-life')  # 0.015011143379182301
l2h.nqxy(mtx=grf95, mty=tv7377, x=25, y=28, n=10, status='last-survivor') # 5.44112479666e-05

l2h.nqxy(mtx=grf95, mty=tv7377, x=25.3, y=28.9, n=10.2, status='joint-life')
# 0.016149189892446625
l2h.nqxy(mtx=grf95, mty=tv7377, x=25.3, y=28.9, n=10.2, status='last-survivor')
# 6.235816078524757e-05
```

6.1.3 t_nqxy

Actuarial Notation: ${}_{t|n}q_{xy}$ and ${}_{t|n}q_{\overline{xy}}$

Usage

```
t_nqxy(mtx,mty,x,y,n,t,status='joint-life',method='udd')
```

Description: Returns the probability of a group (x, y) (joint life) or $(\overline{x, y})$ (last survivor) to survive t years and extinguishes before $t + n$ years.

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x) at the beginning of the contract
y	age of (y) at the beginning of the contract
t	deferment period
n	period of time
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
l2h.t_nqxy(mtx=grf95, mty=tv7377, x=25, y=28, n=10, t=5, status='joint-life')
# 0.02113247574184618
l2h.t_nqxy(mtx=grf95, mty=tv7377, x=25, y=28, n=10, t=5, status='last-survivor')
# 0.0001707562649220229
```

Some more Examples

```
## Considering two individuals of ages x=35 and y=39:
# probability of at least one of the individuals to die in the folowing 6 years
l2h.nqxy(mtx=grf95, mty=tv7377, x=35, y=39, n=6, status='joint-life')  # 0.0167531655998

# probability that both individuals are dead in the next 10 years
l2h.nqxy(mtx=grf95, mty=tv7377, x=35, y=39, n=10, status='last-survivor') # 0.000236058575

# probability of both individuals being alive 3 years from now
l2h.npxy(mtx=grf95, mty=tv7377, x=35, y=39, n=3, status='joint-life') # 0.9925748168560576

# probability that, at least one of them is alive 20 years from now
l2h.npxy(mtx=grf95, mty=tv7377, x=35, y=39, n=20, status='last-survivor') # 0.998065729626633
```

6.1.4 exy

Actuarial Notation: e_{xy} and $e_{\overline{xy}}$

Usage

```
exy(mtx, mty, x, y, status='joint-life', method='udd')
```

Description: Returns the life expectancy of a group of two individuals (x) and (y) for joint life or last survival methods, ie, (x, y) and $(\overline{x, y})$.

Parameters

x	age of (x) at the beginning of the contract
y	age of (y) at the beginning of the contract
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'

Examples

```
l2h.exy(mt_GRF95,mt_TV7377, x=50, y=45,status='joint-life')    # 30.61022001821019
l2h.exy(mt_GRF95,mt_TV7377, x=50, y=45,status='last-survivor') # 44.732448724147964
```

6.1.5 Groups with more than two lives

Using the previous presented functions, it is possible to compute some probabilities for groups with more that two heads. Although the package does not contain specific functions for more than two heads, the following example illustrates how to use the available functions to evaluate some probabilities for a group of three individuals $(x), (y)$ and (z).

Example:
Let us consider a group of three individuals, which extinguishes upon the second death.
The probability of such a group survives at least n years may be computed by

$$\begin{aligned} & {}_np_{xy} + {}_np_{xz} + {}_np_{yz} - 2\,{}_np_{xyz} = \\ = \quad & {}_np_{xy} + {}_np_{xz} + {}_np_{yz} - 2\,{}_np_x \cdot {}_np_y \cdot {}_np_z \end{aligned}$$

Considering $(x) = (35)$, $(y) = (40)$, $(z) = (50)$ and $n = 10$, the above probability may be computed as:

```
x = 35
y = 40
z = 50

px = grf95.npx(x, 10)
py = tv7377.npx(y, 10)
pz = tv7377.npx(z, 10)
pxy = l2h.npxy(mtx=grf95, mty=tv7377,x=x, y=y, n=10, status='joint-life')
pxz = l2h.npxy(mtx=grf95, mty=tv7377, x=x, y=z, n=10, status='joint-life')
pyz = l2h.npxy(mtx=tv7377, mty=tv7377, x=y, y=z, n=10, status='joint-life')

prob_surv_group = pxy + pxz + pyz - 2 * px * py * pz
print(f'probability: {prob_surv_group}')
# 0.9979281371806732
```

Example:
Let us consider a group of three individuals, which exists as long as exactly two individuals are alive.
The probability of two of the individuals are alive after n years is given by:

$$\begin{aligned} & {}_np_{xy} + {}_np_{xz} + {}_np_{yz} - 3\,{}_np_{xyz} = \\ = \quad & {}_np_{xy} + {}_np_{xz} + {}_np_{yz} - 3\,{}_np_x \cdot {}_np_y \cdot {}_np_z \end{aligned}$$

and for the same $(x) = (35)$, $(y) = (40)$, $(z) = (50)$ individuals of the previous example, and again for $n = 10$, the previous probability may be computed as:

```python
x = 35
y = 40
z = 50
px = grf95.npx(x, 10)
py = tv7377.npx(y, 10)
pz = tv7377.npx(z, 10)
pxy = l2h.npxy(mtx=grf95, mty=tv7377, x=x, y=y, n=10, status='joint-life')
pxz = l2h.npxy(mtx=grf95, mty=tv7377, x=x, y=z, n=10, status='joint-life')
pyz = l2h.npxy(mtx=tv7377, mty=tv7377, x=y, y=z, n=10, status='joint-life')

prob_surv_group2 = pxy + pxz + pyz - 3 * px * py * pz
print(f'probability: {prob_surv_group2}')
# 0.0834682538323328
```

6.2 Multiple Lives Annuities

In this section, we present the functions that allow for the computation of most common annuities for groups of two lives: Joint life annuities, Last Survivor annuities and Reversionary annuities. All functions are generically developed for whole life, temporary terms, deferment periods, fractional payments, geometric growth and non-integer ages.

All annuities are developed for both **joint life** (annuity paid while both (x) ad (y) are still alive) and **last survivor** (annuity paid while atr least one of (x) and (y) are still alive).

6.2.1 axy

Actuarial Notation: a_{xy}, $a_{\overline{xy}}$, $a_{xy}^{(m)}$, $a_{\overline{xy}}^{(m)}$, $(Ga^{(m)})_{xy}^{(m)}$ and $(Ga^{(m)})_{\overline{xy}}^{(m)}$

Usage

```python
axy(mtx, mty, x, y, i=None, g=0, m=1, status='joint-life', method='udd')
```

Description: Returns the actuarial present value of an immediate whole life annuity paid for a group of two lives. For constant annuities, pays 1 per time period. For fractional annuities, payments of $1/m$ are made m times per year at the end of the periods. For annuities with geometric growth, the rate is g for each payment period.

Parameters

mtx	Mortality table for (x)
mty	Mortality table for (y)
x	Age of (x) at the beginning of the contract
y	Age of (x) at the beginning of the contract
i	Interest rate, in percentage (e.g. 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments.
m	Number of payments in each period of the interest rate
status	Use *joint-life* or *last-survivor*
method	Approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# Whole life unitary, immediate annuity, paid annually
l2h.axy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=1, status='joint-life')
# 2.1993512333648
l2h.axy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=1, status='last-survivor')
# 6.8225885201728

# Whole life unitary, immediate annuity, paid semi-annually
l2h.axy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=2, status='joint-life')
# 2.4380423029643
l2h.axy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=2, status='last-survivor')
# 7.0791923426166

# Whole life unitary, immediate annuity, paid annually, with geometric growth of payments
l2h.axy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=1, m=1, status='joint-life')
# 2.2390979768651
l2h.axy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=1, m=1, status='last-survivor')
# 2.4886702281877

# Whole life unitary, immediate annuity, paid semi-annually, with geometric growth of payments
l2h.axy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=1, m=2, status='joint-life')
# 7.1346827052150
l2h.axy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=1, m=2, status='last-survivor')
# 7.4269936907418
```

6.2.2 aaxy

Actuarial Notation: $\ddot{a}_{xy}$, $\ddot{a}_{\overline{xy}}$, $\ddot{a}^{(m)}_{xy}$, $\ddot{a}^{(m)}_{\overline{xy}}$, $(G\ddot{a}^{(m)})^{(m)}_{xy}$ and $(G\ddot{a}^{(m)})^{(m)}_{\overline{xy}}$

Usage

```
aaxy(mtx, mty, x, y, i=None, g=0, m=1, status='joint-life', method='udd')
```

Description: Returns the actuarial present value of a due whole life annuity paid for a group of two lives. For constant annuities, pays 1 per time period. For fractional annuities, payments of $1/m$ are made m times per year at the beginning of the periods. For annuities with geometric growth, the rate is g for each payment period.

Parameters

mtx	Mortality table for (x)
mty	Mortality table for (y)
x	Age of (x) at the beginning of the contract
y	Age of (x) at the beginning of the contract
i	Interest rate, in percentage (e.g. 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments.
m	Number of payments in each period of the interest rate
status	Use *joint-life* or *last-survivor*
method	Approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# Whole life unitary, immediate annuity, paid annually
l2h.aaxy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=1, status='joint-life')
# 3.1993512333648
l2h.aaxy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=1, status='last-survivor')
# 7.8225885201728

# Whole life unitary, immediate annuity, paid semi-annually
l2h.aaxy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=2, status='joint-life')
# 2.9380423029643
l2h.aaxy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=2, status='last-survivor')
# 7.5791923426166

# Whole life unitary, immediate annuity, paid annually, with geometric growth of payments
l2h.aaxy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=1, m=1, status='joint-life')
# 3.2614889566337
l2h.aaxy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=1, m=1, status='last-survivor')
# 8.2060295322671

# Whole life unitary, immediate annuity, paid semi-annually, with geometric growth of payments
l2h.aaxy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=1, m=2, status='joint-life')
# 3.0010826255273
l2h.aaxy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=1, m=2, status='last-survivor')
# 7.9640362830804
```

6.2.3 t_axy

Actuarial Notation: ${}_{t|}a_{xy}$, ${}_{t|}a_{\overline{xy}}$, ${}_{t|}a_{xy}^{(m)}$, ${}_{t|}a_{\overline{xy}}^{(m)}$, ${}_{t|}(Ga^{(m)})_{xy}^{(m)}$ and ${}_{t|}(Ga^{(m)})_{\overline{xy}}^{(m)}$

Usage

```
t_axy(mtx, mty, x, y, i=None, g=0, m=1, defer=0, status='joint-life', method='udd')
```

Description: Returns the actuarial present value of an immediate whole life annuity paid for a group of two lives. For constant annuities, pays 1 per time period. For fractional annuities, payments of $1/m$ are made m times per year at the end of the periods. For annuities with geometric growth, the rate is g for each payment period.

Parameters

mtx	Mortality table for (x)
mty	Mortality table for (y)
x	Age of (x) at the beginning of the contract
y	Age of (x) at the beginning of the contract
i	Interest rate, in percentage (e.g. 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments.
m	Number of payments in each period of the interest rate
defer	number of deferment years
status	Use *joint-life* or *last-survivor*
method	Approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# Whole life unitary, annuity, paid annually, with 2 years deferment
l2h.t_axy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=1, defer=2, status='joint-life')
# 0.9670660101740
l2h.t_axy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=1, defer=2, status='last-survivor')
# 4.9549321537268
```

6.2.4 t_aaxy

Actuarial Notation: ${}_{t|}\ddot{a}_{xy}$, ${}_{t|}\ddot{a}_{\overline{xy}}$, ${}_{t|}\ddot{a}^{(m)}_{xy}$, ${}_{t|}\ddot{a}^{(m)}_{\overline{xy}}$, ${}_{t|}(G\ddot{a}^{(m)})^{(m)}_{xy}$ and ${}_{t|}(G\ddot{a}^{(m)})^{(m)}_{\overline{xy}}$

Usage

```
t_aaxy(mtx, mty, x, y, i=None, g=0, m=1, defer=0, status='joint-life', method='udd')
```

Description: Returns the actuarial present value of a due whole life annuity paid for a group of two lives. For constant annuities, pays 1 per time period. For fractional annuities, payments of $1/m$ are made m times per year at the beginning of the periods. For annuities with geometric growth, the rate is g for each payment period.

Parameters

mtx	Mortality table for (x)
mty	Mortality table for (y)
x	Age of (x) at the beginning of the contract
y	Age of (x) at the beginning of the contract
i	Interest rate, in percentage (e.g. 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments.
m	Number of payments in each period of the interest rate
defer	number of deferment years
status	Use *joint-life* or *last-survivor*
method	Approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# Temporary life annuity due, paid annually, with two years deferment
l2h.t_aaxy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=1, defer=2, status='joint-life')
# 1.4765856167554
l2h.t_aaxy(mtx=tv7377, mty=grf95, x=90, y=95, i=2, g=0, m=1, defer=2, status='last-survivor')
# 5.8581438045273
```

6.2.5 naxy

Actuarial Notation: $a_{xy:\overline{n}|}$, $a_{\overline{xy}:\overline{n}|}$, $a^{(m)}_{xy:\overline{n}|}$, $a^{(m)}_{\overline{xy}:\overline{n}|}$, $(Ga^{(m)})^{(m)}_{xy:\overline{n}|}$ and $(Ga^{(m)})^{(m)}_{\overline{xy}:\overline{n}|}$

Usage

```
naxy(mtx, mty, x, y, n, i=None, g=0, m=1, status='joint-life', method='udd')
```

Description: Returns the actuarial present value of an immediate temporary life annuity paid for a group of two lives. For constant annuities, pays 1 per time period. For fractional annuities, payments of $1/m$ are made m times per year at the end of the periods. For annuities with geometric growth, the rate is g for each payment period.

Parameters

mtx	Mortality table for (x)
mty	Mortality table for (y)
x	Age of (x) at the beginning of the contract
y	Age of (x) at the beginning of the contract
n	number of periods
i	Interest rate, in percentage (e.g. 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments.
m	Number of payments in each period of the interest rate
status	Use *joint-life* or *last-survivor*
method	Approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# Temporary immediate life annuity, paid semi-annually with 10 years term
l2h.naxy(mtx=tv7377, mty=grf95, x=90, y=95, n=10, i=2, g=0, m=2, status='joint-life')
# 2.4319176604755
l2h.naxy(mtx=tv7377, mty=grf95, x=90, y=95, n=10, i=2, g=0, m=2, status='last-survivor')
# 6.2483535823922
```

6.2.6 naaxy

Actuarial Notation: $\ddot{a}_{xy:\overline{n}|}$, $\ddot{a}_{\overline{xy}:\overline{n}|}$, $\ddot{a}^{(m)}_{xy:\overline{n}|}$, $\ddot{a}^{(m)}_{\overline{xy}:\overline{n}|}$, $(G\ddot{a}^{(m)})^{(m)}_{xy:\overline{n}|}$ and $(G\ddot{a}^{(m)})^{(m)}_{\overline{xy}:\overline{n}|}$

Usage

```
naaxy(mtx, mty, x, y, n, i=None, g=0, m=1, status='joint-life', method='udd')
```

Description: Returns the actuarial present value of a temporary life annuity due paid for a group of two lives. For constant annuities, pays 1 per time period. For fractional annuities, payments of $1/m$ are made m times per year at the end of the periods. For annuities with geometric growth, the rate is g for each payment period.

Parameters

mtx	Mortality table for (x)
mty	Mortality table for (y)
x	Age of (x) at the beginning of the contract
y	Age of (x) at the beginning of the contract
n	number of periods
i	Interest rate, in percentage (e.g. 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments.
m	Number of payments in each period of the interest rate
status	Use *joint-life* or *last-survivor*
method	Approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# Temporary due life annuity, paid semi-annually with 10 years term
l2h.naaxy(mtx=tv7377, mty=grf95, x=90, y=95, n=10, i=2, g=0, m=2, status='joint-life')
# 2.9278370585219
l2h.naaxy(mtx=tv7377, mty=grf95, x=90, y=95, n=10, i=2, g=0, m=2, status='last-survivor')
# 6.62436464286
```

6.2.7 t_naxy

Actuarial Notation: ${}_{t|}a_{xy:\overline{n|}}$, ${}_{t|}a_{\overline{xy}:\overline{n|}}$, ${}_{t|}a^{(m)}_{xy:\overline{n|}}$, ${}_{t|}a^{(m)}_{\overline{xy}:\overline{n|}}$, ${}_{t|}(Ga^{(m)})^{(m)}_{xy:\overline{n|}}$ and ${}_{t|}(Ga^{(m)})^{(m)}_{\overline{xy}:\overline{n|}}$

Usage

```
t_naxy(mtx, mty, x, y, n, i=None, g=0, m=1, defer=0, status='joint-life', method='udd')
```

Description: Returns the actuarial present value of a deferred temporary life annuity paid for a group of two lives. For constant annuities, pays 1 per time period. For fractional annuities, payments of $1/m$ are made m times per year at the end of the periods. For annuities with geometric growth, the rate is g for each payment period.

Parameters

mtx	Mortality table for (x)
mty	Mortality table for (y)
x	Age of (x) at the beginning of the contract
y	Age of (x) at the beginning of the contract
n	number of periods
i	Interest rate, in percentage (e.g. 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments.
m	Number of payments in each period of the interest rate
defer	Number of deferment periods
status	Use *joint-life* or *last-survivor*
method	Approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# Temporary deferred life annuity, paid semi-annually with 10 years term
l2h.t_naxy(mtx=tv7377,mty=grf95,x=90, y=95, n=10, i=2, g=0, m=2, defer=2, status='joint-life')
# 1.0874293826744
l2h.t_naxy(mtx=tv7377,mty=grf95,x=90,y=95, n=10,i=2,g=0, m=2, defer=2, status='last-survivor')
# 4.7199415824277
```

6.2.8 t_naaxy

Actuarial Notation: ${}_{t|}\ddot{a}_{xy:\overline{n|}}$, ${}_{t|}\ddot{a}_{\overline{xy}:\overline{n|}}$, ${}_{t|}\ddot{a}^{(m)}_{xy:\overline{n|}}$, ${}_{t|}\ddot{a}^{(m)}_{\overline{xy}:\overline{n|}}$, ${}_{t|}(G\ddot{a}^{(m)})^{(m)}_{xy:\overline{n|}}$ and ${}_{t|}(G\ddot{a}^{(m)})^{(m)}_{\overline{xy}:\overline{n|}}$

Usage

```
t_naaxy(mtx, mty, x, y, n, i=None, g=0, m=1, defer=0, status='joint-life', method='udd')
```

Description: Returns the actuarial present value of a deferred temporary life annuity paid for a group of two lives. For constant annuities, pays 1 per time period. For fractional annuities, payments of $1/m$ are made m times per year at the beginning of the periods. For annuities with geometric growth, the rate is g for each payment period.

Parameters

mtx	Mortality table for (x)
mty	Mortality table for (y)
x	Age of (x) at the beginning of the contract
y	Age of (x) at the beginning of the contract
n	number of periods
i	Interest rate, in percentage (e.g. 2 for 2%)
g	Growth rate (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constant payments.
m	Number of payments in each period of the interest rate
defer	Number of deferment periods
status	Use *joint-life* or *last-survivor*
method	Approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# Temporary deferred life annuity, paid semi-annually with 10 years term paid in the beginning
    of periods
l2h.t_naaxy(mtx=tv7377,mty=grf95,x=90,y=95, n=10,i=2,g=0,m=2, defer=2, status='joint-life')
# 1.3417110613438
l2h.t_naaxy(mtx=tv7377,mty=grf95,x=90,y=95,n=10,i=2,g=0,m=2,defer=2,status='last-survivor')
# 5.0967616156346
```

6.3 Multiple Lives Insurance

In this section we present the classical Expected Values of Life Insurance benefits, for products whose benefits depend on the death and/or survival of two individuals aged (x) and (y). As in the previous section, we consider the two common approaches: *joint-life* (x, y) and *last survivor* $(\overline{x, y})$.

Functions are developed for payments at the end of the period of the death or at "the moment of death" (in average, in the middle of the period). Using the fractional commutation tables, the life insurance contracts can be evaluated with the intended fractional time.

As commonly in the actuarial notation, we use A_{xy} for benefits paid in the end of the year, $A_{xy}^{(m)}$ for payments in the end of the m-th period of the year and $\bar{A}_{xy}$ for benefits paid in the "moment of death" (in average, in the middle of the year).

Once again, all functions are available for non-integer ages, and the user may choose between three approximation methods: *Uniform Distribution of Death*, *Constant Force of Mortality* and *Balducci Approximation*.

In each subsection, when applicable, we present both functions when considering payments in the end of the year of in the "moment of death" (approximated by the middle of the year).

6.3.1 Pure Endowment/ Deferred Capital/ Expected Present Value / nExy

Actuarial Notation: ${}_nE_{xy}$ and ${}_nE_{\overline{xy}}$

Usage

```
nExy(mtx, mty, x, y, i=None, n=1, status='joint-life', method='udd')
```

Description: Returns the actuarial expected present value of an unitary capital paid in n years term, upon the survival of a group of two lives. Probabilities for a group of two lives (x) and (y), for joint life or last survival methods, ie, (x, y) and $(\overline{x, y})$ are considered.

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x) at the beginning
y	age of (y) at the beginning
i	Interest rate, in percentage (e.g. 2 for 2%)
n	number of years until term
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
l2h.nExy(mtx=grf95,mty=tv7377, x=35, y=40, i=2, n=1, status='joint-life')
# 0.9780058667674981
l2h.nExy(mtx=grf95,mty=tv7377, x=35, y=40, i=2, n=1, status='last-survivor')
# 0.9803908602913254

l2h.nExy(mtx=grf95,mty=tv7377,x=51.8,y=48.3,i=2,n=10.5, status='joint-life',method='bal')
# 0.7501997252543674
l2h.nExy(mtx=grf95,mty=tv7377,x=51.8,y=48.3,i=2,n=10.5,status='last-survivor',method='bal')
# 0.81113659782566
```

Some Other Examples

```
# Actuarial Expected Present Value of 50.000 m.u. paid to a group with two lives (x)=40 and
    (y)=50, after 15 years

# Joint Life Group
50000*l2h.nExy(mtx=grf95,mty=tv7377, x=40, y=50, i=2, n=15, status='joint-life')
# 32809.08 m.u.

# Last Survivor Group
50000*l2h.nExy(mtx=grf95,mty=tv7377, x=40, y=50, i=2, n=15, status='last-survivor')
# 37068.79 m.u.
```

6.3.2 Axy and Axy_

Actuarial Notation: A_{xy}, $A_{xy}^{(m)}$, $(GA)_{xy}$, $(GA)_{xy}^{(m)}$ and $\bar{A}_{xy}$, $\bar{A}_{xy}^{(m)}$, $(G\bar{A})_{xy}$, $(G\bar{A})_{xy}^{(m)}$

$$A_{\overline{xy}},\ A_{\overline{xy}}^{(m)},\ (GA)_{\overline{xy}},\ (GA)_{\overline{xy}}^{(m)} \text{ and } \bar{A}_{\overline{xy}},\ \bar{A}_{\overline{xy}}^{(m)},\ (G\bar{A})_{\overline{xy}},\ (G\bar{A})_{\overline{xy}}^{(m)}$$

Usage

```
Axy(mtx, mty, x, y, i=None, g=0, m=1, status='joint-life', method='udd')   # end of the year

Axy_(mtx, mty, x, y, i=None, g=0, status='joint-life', method='udd')  # moment of death
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a whole life insurance (i.e. net single premium), that pays 1 at the end of the period/moment of death of a group of two individuals, for joint-life or last survivor methods. For fractional years, payments are made in the end of the periods.

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x)
y	age of (y)
i	Interest rate (e.g. use 2 for 2%)
g	Growth rate (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constants payments.
m	Number of fractions for each period of the interest rate
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# End of the Year
l2h.Axy(mtx=grf95, mty=tv7377, x=35, y=40, i=2, status='joint-life')    # 0.4883589555345963
l2h.Axy(mtx=grf95, mty=tv7377, x=35, y=40, i=2, status='last-survivor')
# 0.3279490658724815
l2h.Axy(mtx=grf95, mty=tv7377, x=35, y=40, i=2, m=2, status='joint-life')
# 0.4908020439476468
l2h.Axy(mtx=grf95, mty=tv7377, x=35, y=40, i=2, m=2, status='last-survivor')
# 0.3295673114271598
l2h.Axy(mtx=grf95, mty=tv7377, x=35, y=40, i=2, g=1, status='joint-life')
# 0.6947836396955362
l2h.Axy(mtx=grf95, mty=tv7377, x=35, y=40, i=2, g=1, status='last-survivor')
# 0.5709211125564813

# Moment of Death
l2h.Axy_(mtx=grf95, mty=tv7377, x=35, y=40, i=2, status='joint-life')   # 0.4932183683115002
l2h.Axy_(mtx=grf95, mty=tv7377, x=35, y=40, i=2, status='last-survivor')
# 0.33121232103103576
l2h.Axy_(mtx=grf95, mty=tv7377, x=35.5, y=40.8, i=2, status='joint-life')
# 0.49972941203977206
l2h.Axy_(mtx=grf95, mty=tv7377, x=55.5, y=40.8, i=2, status='last-survivor')
# 0.4263873876757644
```

6.3.3 t_Axy and t_Axy_

Actuarial Notation: ${}_tA_{xy}$, ${}_tA_{xy}^{(m)}$, ${}_t(GA)_{xy}$, ${}_t(GA)_{xy}^{(m)}$ and ${}_t\bar{A}_{xy}$, ${}_t\bar{A}_{xy}^{(m)}$, ${}_t(G\bar{A})_{xy}$, ${}_t(G\bar{A})_{xy}^{(m)}$

${}_tA_{\overline{xy}}$, ${}_tA_{\overline{xy}}^{(m)}$, ${}_t(GA)_{\overline{xy}}$, ${}_t(GA)_{\overline{xy}}^{(m)}$ and ${}_t\bar{A}_{\overline{xy}}$, ${}_t\bar{A}_{\overline{xy}}^{(m)}$, ${}_t(G\bar{A})_{\overline{xy}}$, ${}_t(G\bar{A})_{\overline{xy}}^{(m)}$

Usage

```
# end of the period
t_Axy(mtx, mty, x, y, i=None, g=0, m=1, defer=0, status='joint-life', method='udd')

# moment of death
t_Axy_(mtx, mty, x, y, i=None, g=0, defer=0, status='joint-life', method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred whole life insurance (i.e. net single premium), that pays 1 at the end of the period/moment of death of a group of two individuals, for joint-life or last survivor methods.

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x) at the beginning of the contract
y	age of (y) at the beginning of the contract
i	Interest rate (e.g. use 2 for 2%)
g	Growth rate (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constants payments.
m	Number of fractions for each period of the interest rate
defer	Number of deferment periods
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# End of the Period
l2h.t_Axy(mtx=grf95, mty=tv7377, x=35.2, y=40.6, i=2, defer=3, status='joint-life', method='udd')  # 0.48509331522128096
l2h.t_Axy(mtx=grf95, mty=tv7377, x=35.2, y=40.6, i=2, defer=3, status='last-survivor', method='udd')  # 0.3292496637625039
l2h.t_Axy(mtx=grf95, mty=tv7377, x=35.2, y=40.6, i=2, g=1, m=4, defer=3, status='joint-life', method='bal')  # 0.6723379046966992
l2h.t_Axy(mtx=grf95, mty=tv7377, x=35.2, y=40.6, i=2, g=1, m=4, defer=3, status='last-survivor', method='bal')  # 0.5573824479627387

# Moment of Death
l2h.t_Axy_(mtx=grf95, mty=tv7377, x=35.2, y=40, i=2, defer=3, status='joint-life', method='udd')    # 0.485844005557544
l2h.t_Axy_(mtx=grf95, mty=tv7377, x=35.2, y=40, i=2, defer=3, status='last-survivor', method='udd')    # 0.33174709690035026
l2h.t_Axy_(mtx=grf95, mty=tv7377, x=35.2, y=40.6, i=2, g=1, defer=3, status='joint-life', method='cfm')   # 0.676502134404232
l2h.t_Axy_(mtx=grf95, mty=tv7377, x=35.2, y=40.6, i=2, g=1, defer=3, status='last-survivor', method='cfm')   # 0.5608068279976409
```

6.3.4 nAxy and nAxy_

Actuarial Notation: $A^1_{xy:\overline{n}|}$, $A^{1(m)}_{xy:\overline{n}|}$, $(GA)^1_{xy:\overline{n}|}$, $(GA)^{1(m)}_{xy:\overline{n}|}$ and $\bar{A}^1_{xy:\overline{n}|}$, $\bar{A}^{1(m)}_{xy:\overline{n}|}$, $(G\bar{A})^1_{xy:\overline{n}|}$, $(G\bar{A})^{1(m)}_{xy:\overline{n}|}$

$A^1_{\overline{xy}:\overline{n}|}$, $A^{1(m)}_{\overline{xy}:\overline{n}|}$, $(GA)^1_{\overline{xy}:\overline{n}|}$, $(GA)^{1(m)}_{\overline{xy}:\overline{n}|}$ and $\bar{A}^1_{\overline{xy}:\overline{n}|}$, $\bar{A}^{1(m)}_{\overline{xy}:\overline{n}|}$, $(G\bar{A})^1_{\overline{xy}:\overline{n}|}$, $(G\bar{A})^{1(m)}_{\overline{xy}:\overline{n}|}$

Usage

```
nAxy(mtx, mty, x, y, n, i=None, g=0, m=1, status='joint-life', method='udd')  #end of the year

nAxy_(mtx, mty, x, y, n, i=None, g=0, status='joint-life', method='udd') #moment of death
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a temporary life insurance (i.e. net single premium), that pays 1 at the end of the period/moment of death of a group of two individuals, for joint-life or last survivor methods.

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x) at the beginning of the contract
y	age of (y) at the beginning of the contract
n	number of terms of years of the contract
i	Interest rate (e.g. use 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constants payments.
m	Number of fractions for each period of the interest rate
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# End of the Period
l2h.nAxy(mtx=grf95, mty=tv7377, x=55.8, y=40, n=10, i=2, status='joint-life', method='bal')
# 0.051536194942196634
l2h.nAxy(mtx=grf95, mty=tv7377, x=55.8, y=40, n=10, i=2, status='last-survivor', method='bal')
# 0.0007237026849450379
l2h.nAxy(mtx=grf95, mty=tv7377, x=35.9, y=40.6, n=15, i=2, g=1, m=4, status='joint-life')
# 0.17823058360706845
l2h.nAxy(mtx=grf95, mty=tv7377, x=35.9, y=40.6, n=15, i=2, g=1, m=4, status='last-survivor')
# 0.12669408363288304


# Moment of Death
l2h.nAxy_(mtx=grf95, mty=tv7377, x=55.8, y=40, n=10, i=2, status='joint-life', method='bal')
# 0.052049005532310566
l2h.nAxy_(mtx=grf95, mty=tv7377, x=55.8, y=40, n=10, i=2, status='last-survivor',method='bal')
# 0.0007309038840508306
l2h.nAxy_(mtx=grf95, mty=tv7377, x=35.9, y=40.6, n=15, i=2, g=1, status='joint-life')
# 0.17372546866918934
l2h.nAxy_(mtx=grf95, mty=tv7377, x=35.9, y=40.6, n=15, i=2, g=1, status='last-survivor')
# 0.1214957914607578
```

6.3.5 t_nAxy and t_nAxy_

Actuarial Notation: ${}_{t|}A^{1}_{xy:\overline{n|}}$, ${}_{t|}A^{1(m)}_{xy:\overline{n|}}$, ${}_{t|}(GA)^{1}_{xy:\overline{n|}}$, ${}_{t|}(GA)^{1(m)}_{xy:\overline{n|}}$ and ${}_{t|}\bar{A}^{1}_{xy:\overline{n|}}$, ${}_{t|}\bar{A}^{1(m)}_{xy:\overline{n|}}$, ${}_{t|}(G\bar{A})^{1}_{xy:\overline{n|}}$, ${}_{t|}(G\bar{A})^{1(m)}_{xy:\overline{n|}}$

${}_{t|}A^{1}_{\overline{xy}:\overline{n|}}$, ${}_{t|}A^{1(m)}_{\overline{xy}:\overline{n|}}$, ${}_{t|}(GA)^{1}_{\overline{xy}:\overline{n|}}$, ${}_{t|}(GA)^{1(m)}_{\overline{xy}:\overline{n|}}$ and ${}_{t|}\bar{A}^{1}_{\overline{xy}:\overline{n|}}$, ${}_{t|}\bar{A}^{1(m)}_{\overline{xy}:\overline{n|}}$, ${}_{t|}(G\bar{A})^{1}_{\overline{xy}:\overline{n|}}$, ${}_{t|}(G\bar{A})^{1(m)}_{\overline{xy}:\overline{n|}}$

Usage

```
# End of the period
t_nAxy(mtx, mty, x, y, n, i=None, g=0, m=1, defer=0, status='joint-life', method='udd')

# Moment of Death
t_nAxy_(mtx, mty, x, y, n, i=None, g=0, defer=0, status='joint-life', method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred temporary life insurance (i.e. net single premium), that pays 1 at the end of the period/moment of death of a group of two individuals, for joint-life or last survivor methods.

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x) at the beginning of the contract
y	age of (y) at the beginning of the contract
n	number of terms of years of the contract
i	Interest rate (e.g. use 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constants payments.
m	Number of fractions for each period of the interest rate
defer	number of years of deferment
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# End of the period
l2h.t_nAxy_(mtx=grf95, mty=tv7377, x=35, y=40, n=20, i=2, defer=3, status='joint-life')
# 0.09156020490195788
l2h.t_nAxy_(mtx=grf95, mty=tv7377, x=35, y=40, n=20, i=2, defer=3, status='last-survivor')
# 0.00212904926970781
l2h.t_nAxy_(mtx=grf95, mty=tv7377, x=35.2, y=40.6, n=20, i=2, g=1, defer=3, status='joint-life
    ', method='cfm')  # 0.22899218321123424
l2h.t_nAxy_(mtx=grf95, mty=tv7377, x=35.2, y=40.6, n=20, i=2, g=1, defer=3, status='last-
    survivor', method='cfm')   # 0.14319533154878325
# Moment of Death
l2h.t_nAxy_(mtx=grf95, mty=tv7377, x=35, y=40, n=15, i=2, g=1, defer=20, status='joint-life')
# 0.20792147954614337
l2h.t_nAxy_(mtx=grf95, mty=tv7377, x=35, y=40, n=15, i=2, g=1,defer=20,status='last-survivor')
# 0.08858787155153025
l2h.t_nAxy_(mtx=grf95, mty=tv7377, x=55.8, y=40, n=10, i=2, defer=10, status='joint-life',
    method='bal')  # 0.09201248078665436
l2h.t_nAxy_(mtx=grf95, mty=tv7377, x=55.8, y=40, n=10, i=2, defer=10, status='last-survivor',
    method='bal')  # 0.0031667131777407304
```

6.3.6 nAExy and nAExy_

Actuarial Notation: $A_{xy:\overline{n}|}$, $A^{(m)}_{xy:\overline{n}|}$, $(GA)_{xy:\overline{n}|}$, $(GA)^{(m)}_{xy:\overline{n}|}$ and $\bar{A}_{xy:\overline{n}|}$, $\bar{A}^{(m)}_{xy:\overline{n}|}$, $(G\bar{A})_{xy:\overline{n}|}$, $(G\bar{A})^{(m)}_{xy:\overline{n}|}$

$A_{\overline{xy}:\overline{n}|}$, $A^{(m)}_{\overline{xy}:\overline{n}|}$, $(GA)_{\overline{xy}:\overline{n}|}$, $(GA)^{(m)}_{\overline{xy}:\overline{n}|}$ and $\bar{A}_{\overline{xy}:\overline{n}|}$, $\bar{A}^{(m)}_{\overline{xy}:\overline{n}|}$, $(G\bar{A})_{\overline{xy}:\overline{n}|}$, $(G\bar{A})^{(m)}_{\overline{xy}:\overline{n}|}$

Usage

```
# End of the period
nAExy(mtx, mty, x, y, n, i=None, g=0, m=1, status='joint-life', method='udd')

# Moment of Death
nAExy_(mtx, mty, x, y, n, i=None, g=0, status='joint-life', method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of an Endowment life insurance (i.e. net single premium), that pays 1 at the end of the period/moment of death of a group of two individuals or pays 1 if the group is alive at the end of the contract, for joint-life or last survivor methods.

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x) at the beginning of the contract
y	age of (y) at the beginning of the contract
n	number of terms of years of the contract
i	Interest rate (e.g. use 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constants payments.
m	Number of fractions for each period of the interest rate
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# End of the period
l2h.nAExy(mtx=grf95, mty=tv7377, x=36, y=41, n=15, i=2, g=1, status='joint-life')
# 0.8660767908641055
l2h.nAExy(mtx=grf95, mty=tv7377, x=36, y=41, n=15, i=2, g=1, status='last-survivor')
# 0.8626451905971021
l2h.nAExy(mtx=grf95, mty=tv7377, x=55.8, y=40, n=10, i=2, status='joint-life', method='cfm')
# 0.8242538413270563
l2h.nAExy(mtx=grf95, mty=tv7377, x=55.8, y=40, n=10, i=2, status='last-survivor',method='cfm')
# 0.8203804618128456
# Moment of Death
l2h.nAExy_(mtx=grf95, mty=tv7377, x=36, y=41, n=15, i=2, g=1, status='joint-life')
# 0.8678007454005405
l2h.nAExy_(mtx=grf95, mty=tv7377, x=36, y=41, n=15, i=2, g=1, status='last-survivor')
# 0.8638424731901125
l2h.nAExy_(mtx=grf95, mty=tv7377, x=55.8, y=40, n=10, i=2, status='joint-life', method='cfm')
# 0.8247666396432718
l2h.nAExy_(mtx=grf95, mty=tv7377, x=55.8, y=40, n=10, i=2,status='last-survivor',method='cfm')
# 0.8203876627116821
```

6.3.7 t_nAExy and t_nAExy_

Actuarial Notation: ${}_{t|}A_{xy:\overline{n}|}$, ${}_{t|}A^{(m)}_{xy:\overline{n}|}$, ${}_{t|}(GA)_{xy:\overline{n}|}$, ${}_{t|}(GA)^{(m)}_{xy:\overline{n}|}$ and ${}_{t|}\bar{A}_{xy:\overline{n}|}$, ${}_{t|}\bar{A}^{(m)}_{xy:\overline{n}|}$, ${}_{t|}(G\bar{A})_{xy:\overline{n}|}$, ${}_{t|}(G\bar{A})^{(m)}_{xy:\overline{n}|}$

${}_{t|}A_{\overline{xy}:\overline{n}|}$, ${}_{t|}A^{(m)}_{\overline{xy}:\overline{n}|}$, ${}_{t|}(GA)_{\overline{xy}:\overline{n}|}$, ${}_{t|}(GA)^{(m)}_{\overline{xy}:\overline{n}|}$ and ${}_{t|}\bar{A}_{\overline{xy}:\overline{n}|}$, ${}_{t|}\bar{A}^{(m)}_{\overline{xy}:\overline{n}|}$, ${}_{t|}(G\bar{A})_{\overline{xy}:\overline{n}|}$, ${}_{t|}(G\bar{A})^{(m)}_{\overline{xy}:\overline{n}|}$

Usage

```
# End of the Period
t_nAExy(mtx, mty, x, y, n, i=None, g=0, m=1, defer=0, status='joint-life', method='udd')

# Moment of Death
t_nAExy_(mtx, mty, x, y, n, i=None, g=0, defer=0, status='joint-life', method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred Endowment life insurance (i.e. net single premium), that pays 1 at the end of the period/moment of death of a group of two individuals or pays 1 at the end of the contract, if the group is alive, for joint-life or last survivor methods.

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x) at the beginning of the contract
y	age of (y) at the beginning of the contract
n	number of terms of years of the contract
i	Interest rate (e.g. use 2 for 2%)
g	Rate of growing (in percentage), for payments evolving geometrically (e.g. 1 for 1%) $g > 0$ for increasing payments, $g < 0$ for decreasing payments and $g = 0$ for constants payments.
m	Number of fractions for each period of the interest rate
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
defer	number of years of deferment
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# End of the period
l2h.t_nAExy(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, g=0, defer=10, status='joint-life')
# 0.5922187614008853
l2h.t_nAExy(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, g=0,defer=10,status='last-survivor')
# 0.6076108022457168

# Moment of Death
l2h.t_nAExy_(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, g=0, defer=10, status='joint-life')
# 0.5933881039489882
l2h.t_nAExy_(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, g=0, defer=10, status='last-survivor')
# 0.6076500583192895
```

6.3.8 t_nIArxy and t_nIArxy_

Actuarial Notation: ${}_{t|}IA^{r}_{xy:\overline{n}|}$, ${}_{t|}IA^{(m)\,r}_{xy:\overline{n}|}$, ${}_{t|}I\bar{A}^{r}_{xy:\overline{n}|}$, ${}_{t|}I\bar{A}^{(m)\,r}_{xy:\overline{n}|}$

$${}_{t|}IA^{r}_{\overline{xy}:\overline{n}|},\ {}_{t|}IA^{(m)\,r}_{\overline{xy}:\overline{n}|},\ {}_{t|}I\bar{A}^{r}_{\overline{xy}:\overline{n}|},\ {}_{t|}I\bar{A}^{(m)\,r}_{\overline{xy}:\overline{n}|}$$

Usage

```
# End of the Year
t_nIArxy(mtx, mty, x, y, n, i=None, defer=0, first_payment=1, inc=1, status='joint-life',
    method='udd')

# Moment of Death
t_nIArxy_(mtx, mty, x, y, n, i=None, defer=0, first_payment=1, inc=1, status='joint-life',
    method='udd')
```

Description: Returns the Expected Present Value (EPV) [Actuarial Present Value (APV)] of a deferred Term Life Insurance (i.e. net single premium), that pays $1 + k$, at the end of year/moment of death of a group of two individuals, if death occurs between ages $x + t + k$ and $x + t + k + 1$, for k=0, 1, ..., $n - 1$. The capital of the first year may differ from the rate of the progression.

Parameters

mtx	mortality table for (x)
mty	mortality table for (y)
x	age of (x) at the beginning of the contract
y	age of (y) at the beginning of the contract
n	number of terms of years of the contract
i	Interest rate (e.g. use 2 for 2%)
defer	number of deferment years
first_payment	amount of capital in the first year
inc	rate of the progression (in monetary units)
status	status under which the probability is to be computed: 'joint-life' or 'last-survivor'
method	For non-integer ages and periods, use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*

Examples

```
# End of the year
l2h.t_nIArxy(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, defer=0, first_payment=1, inc=1,
    status='joint-life')
# 0.6490737001595632
l2h.t_nIArxy(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, defer=1, first_payment=1, inc=1,
    status='joint-life')
# 0.6753668707865095
l2h.t_nIArxy(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, defer=0, first_payment=1, inc=1,
    status='last-survivor')
# 0.6547667655641614
l2h.t_nIArxy(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, defer=1, first_payment=1, inc=1,
    status='last-survivor')
# 0.6826053359315403
```

```
# Moment of Death
l2h.t_nIArxy_(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, defer=0, first_payment=1, inc=1,
    status='joint-life')
# 0.6555323040122456
l2h.t_nIArxy_(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, defer=1, first_payment=1, inc=1,
    status='joint-life')
# 0.6820871046714496
l2h.t_nIArxy_(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, defer=0, first_payment=1, inc=1,
    status='last-survivor')
# 0.6612820182290613
l2h.t_nIArxy_(mtx=grf95, mty=tv7377, x=40, y=45, n=15, i=2, defer=1, first_payment=1, inc=1,
    status='last-survivor')
# 0.6893975961192896
```

7 Actuarial Tables

From a given Mortality Table, the library allows for the construction of an Actuarial Table (or Commutation Table) providing and allowing to access the values of the common commutation symbols.
This functions are useful for academic purposes, or to implement "old" life contingency products where commutation symbols were commonly used.
At the moment, actuarial evaluation should use cashflow projections, for which interest rate curves should be considered instead of a fixed one, as in the case of actuarial tables. Despite this, this library allows for the computation of traditional methods in life insurance and, for that purpose, present value of life annuities and life insurance contracts are defined, in the library, as properties of the actuarial table.
Tables 16 and 17, in section 7.4 and Tables 18 and 19, in section 7.5, resume the syntax to compute these values from an actuarial table.

7.1 Class CommutationTable

class *CommutationTable*
This class instantiates, for a specific mortality table and interest rate, all the usual commutation functions: D_x, N_x, S_x, C_x ,M_x and R_x.

Usage

```
CommutationFunctions(i=None, g=0, data_type='q', mt=None, perc=100, app_cont=False)
```

Description

Initializes the CommutationTable class so that we can construct an actuarial table with the usual fields.

Parameters

i	Interest Rate, in percentage. For instance, use 5 for 5%.
g	Rate of growing (in percentage), for capitals evolving geometrically. $g > 0$ for increasing capitals and $g < 0$ for decreasing capitals
data_type	Use 'l' for l_x, 'p' for p_x and 'q' for q_x.
mt	The mortality table, in array format, according to the data_type defined
perc	The percentage of q_x to use, e.g., use 50 for 50%.
app_cont	Use True for continuous approach (deaths occur, in average, in the middle of the year) or False for considering death payments are considered in the end of the year.

Examples

```
from soa_tables import read_soa_table_xml as rst
from lifeActuary import commutation_table as ct

# reads soa table TV7377
soa = rst.SoaTable('soa_tables/' + 'TV7377' + '.xml')

# reads Excel file with mortality tables
table_manual_qx = pd.read_excel('soa_tables/' + 'tables_manual' + '.xlsx', sheet_name='qx')
table_manual_lx = pd.read_excel('soa_tables/' + 'tables_manual' + '.xlsx', sheet_name='lx')
```

```
# creates an Actuarial Table from qx of SOA table
tv7377_ct = ct.CommutationFunctions(i=2, g=0, data_type='q', mt=soa.table_qx, perc=100,
    app_cont=False)

# creates an Actuarial Table from qx of GRF95 available in the Excel file
grf95_ct = ct.CommutationFunctions(i=1.5, g=0, data_type='q', mt=list(table_manual_qx['GRF95'
    ]), perc=100, app_cont=False)

# creates an Actuarial Table from lx of GRM95 available in the Excel file
grm95_ct = ct.CommutationFunctions(i=1.5, g=0, data_type='l', mt=list(table_manual_lx['GRM95'
    ]), perc=100, app_cont=False)
```

With the construction of the commutation table (ct), the class computes methods that are useful when computing actuarial evaluations, which are described in sections 7.3.1 to 7.3.7.

7.2 Class CommutationTableFrac

class *CommutationTableFrac*
This class instantiates, for a specific mortality table and interest rate, all the usual commutation functions: D_x, N_x, S_x, C_x, M_x and R_x, for ages $x + \dfrac{1}{frac} \in \left\{0, \dfrac{1}{frac}, \ldots, \omega + \dfrac{frac-1}{frac}\right\}$.

Usage

```
CommutationFunctions(i=None, g=0, data_type='q', mt=None, perc=100, frac=2, method='udd')
```

Description

Initializes the CommutationTableFrac class so that we can construct an actuarial table with the usual fields, for all indented fractional ages, considering that payments are made in the end of the fractional times.

Parameters

i	Interest Rate, in percentage. For instance, use 5 for 5%.
g	Rate of growing (in percentage), for capitals evolving geometrically. $g > 0$ for increasing capitals and $g < 0$ for decreasing capitals. For instance, use 2 for 2%.
data_type	Use 'l' for l_x, 'p' for p_x and 'q' for q_x.
mt	The mortality table, in array format, according to the data_type defined
perc	The percentage of q_x to use, e.g., use 50 for 50%.
frac	Number of fractional ages for each age x
method	Approximation method for non-integer ages. Use 'udd' for *Uniform Distribution of Death*, 'cfm' for *Constant Force of Mortality* and 'bal' for *Balducci approximation*.

Examples

```
from soa_tables import read_soa_table_xml as rst
from lifeActuary.commutation_table_frac import CommutationFunctionsFrac

```

```
# reads SOA table
soa = rst.SoaTable('soa_tables/' + 'TV7377' + '.xml')

# creates an actuarial table from qx of SOA table, for ages x+k*0.5, x=0,...,w, k=0,1.
tv7377_ct_f2 = CommutationFunctionsFrac(i=2, g=0, data_type='q', mt=soa.table_qx, perc=100,
    frac=2, method='udd')

# creates an actuarial table from qx of SOA table, for ages x+k*0.25, x=0,...,w, k=0,1,2,3
tv7377_ct_f4 = CommutationFunctionsFrac(i=2, g=0, data_type='q', mt=soa.table_qx, perc=100,
    frac=4, method='udd')

# creates an actuarial table from qx of SOA table, for ages x+k*1/6, x=0,...,w, k=0,1,...,5
tv7377_ct_f6 = CommutationFunctionsFrac(i=2, g=0, data_type='q', mt=soa.table_qx, perc=100,
    frac=6, method='udd')

# creates an actuarial table from qx of SOA table, for ages x+k*1/365, x=0,...,w,
     k=0,1,...,364
tv7377_ct_f365 = CommutationFunctionsFrac(i=2, g=0, data_type='q', mt=soa.table_qx, perc=100,
    frac=365, method='udd')

# creates an actuarial table from qx of SOA table, for ages x+k*0.5, x=0,...,w, k=0,1 with
    rate of growing of 1%
tv7377_ct_f2_g1 = CommutationFunctionsFrac(i=2, g=1, data_type='q', mt=soa.table_qx, perc=100,
     frac=2, method='udd')
```

With the construction of the fractional commutation table, the class computes methods that may be useful when computing actuarial evaluations, described in section 7.3.

Observation: Naturally, the *CommutationTableFrac* class with *frac*=1 produces the same results as the class *CommutationTable*.

7.2.1 Function age_to_index()

The *CommutationTableFrac* class produces, for each commutation symbol, a vector with $\omega + (frac - 1) \times \omega$ components. To simplify the access of each method to each fractional age, the user should use the following function:

Usage

```
age_to_index(age_int, age_frac)
```

Description: Returns the index of the vector position in a given method of an actuarial table, corresponding to the age_int+age_frac position.

Parameters

age_int	integer part of the age
age_frac	fractional part of the age

Examples

```
tv7377_ct_f2.age_to_index(50, 0.5)   # 101
tv7377_ct_f4.age_to_index(50, 0.75)  # 203
tv7377_ct_f6.age_to_index(50, 5/6)   # 305
```

7.3 Commutation Table Methods

In the following sections we detail the available methods in a commutation table object, which we will denote by *ct*, where we included the commutation symbols, as well as methods for evaluating standard life annuities and life insurance from commutation symbols.
Naturally, for life annuities and life insurance (for integer ages and terms) the results are coincident with the ones presented in the correspondent previous chapters, where life annuities and life insurance functions were presented and illustrated. In the case of life annuities paid m times per year, within the commutation table object, we follow the standard approximations as, for instance,

$$a_x^{(m)} = a_x + \frac{m-1}{2m} \qquad \text{and} \qquad \ddot{a}_x^{(m)} = a_x + \frac{m+1}{2m}$$

7.3.1 v

Actuarial Notation	v
Definition	$\frac{1}{1+i}$
Usage	ct.v
Example	tv7377_ct.v
Result	0.9803921568627451

7.3.2 Dx and Dx_frac

Actuarial Notation	D_x
Usage	ct.Dx[x]
Args	x: age as an integer
Example	tv7377_ct.Dx[50]
Result	34944.42647196618

As for the Dx_frac method, the following examples illustrate the use of the method:

Examples

```
x=50.5
index_age=tv7377_ct_f2.age_to_index(int(x), x-int(x))  #101
a=tv7377_ct_f2.Dx_frac[index_age]
print(f'For age {x} with index {index_age}, Dx={a}')
# For age 50.5, with index 101, Dx=34535.02547926754
```

7.3.3 Nx and Nx_frac

Actuarial Notation	N_x
Usage	ct.Nx[x]
Args	x: age as an integer
Example	tv7377_ct.Nx[50]
Result	788151.7176774722

As for the Nx_frac method, the following examples illustrates the use of the method:

Examples

```
x=35+1/6
index_age=tv7377_ct_f6.age_to_index(int(x), x-int(x))  # 211
a=tv7377_ct_f6.Nx_frac[index_age]
print(f'For age {x} with index {index_age}, Nx={a}')
# For age 35.166666666666664, with index 211, Nx=8333822.587495911


## Present value of an unitary whole life annuity due, paid quarterly to an individual aged
    x=65.25
x=65.25
index_age=tv7377_ct_f4.age_to_index(int(x), x-int(x))  # 261
a=tv7377_ct_f4.Nx_frac[index_age]/tv7377_ct_f4.Dx_frac[index_age]
print(f'For age {x} with index {index_age}, ax={a}')
# For age 65.25, with index 261, ax=56.9123257953868


## Present value of an unitary whole life annuity due, paid annually to an individual aged
    x=65.25
print(f'For age {x}, with index {index_age}, ax(4)={a/4}')
# For age 65.25, with index 261, ax(4)=14.2280814488467


## Present value of an unitary whole life annuity immediate, paid daily to an individual aged
    x=66+120/365
x=66+120/365
index_age=tv7377_ct_f365.age_to_index(int(x), x-int(x))  # 24210
a=tv7377_ct_f365.Nx_frac[index_age]/tv7377_ct_f365.Dx_frac[index_age]
print(f'For age {x} with index {index_age}, ax={a}')
# For age 66.32876712328768 with index 24210, ax=4931.626631895195


## Present value of an unitary whole life annuity immediate, paid annually to an individual
    aged x=66+120/365
print(f'For age {x}, with index {index_age}, ax(365)={a/365}')
# For age 66.32876712328768, with index 24210, ax(365)=13.511305840808753
```

7.3.4 Sx and Sx_frac

Actuarial Notation	S_x
Usage	ct.Sx[x]
Args	x: age as an integer
Example	tv7377_ct.Sx[50]
Result	12024274.4751688

Examples for using Sx_frac are similar to the previous methods and will be omitted.

In the following methods, the choice between payments made in the "end of the year" or in the "moment of death" must be performed when constructing the commutation table (False or True in the app_cont parameter, respectively). In that sense, the actuarial notation in each of the following methods is given for both scenarios.

As for the computation of these commutation symbols for non-integer ages, the reasoning is the same as the previous sub-sections: define a fractional actuarial table and use the methods for non-integer ages, according the defined fractions of the year. In that sense, examples will not the included.

7.3.5 Cx and Cx_frac

Actuarial Notation	C_x or $\overline{C}_x$
Usage	ct.Cx[x]
Args	x: age as an integer number
Example	tv7377_ct.Cx[50]
Result	128.94202849786421

7.3.6 Mx and Mx_frac

Actuarial Notation	M_x or $\overline{M}_x$
Usage	ct.Mx[x]
Args	x: age as an integer number
Example	tv7377_ct.Mx[50]
Result	19490.471223388264

7.3.7 Rx and Rx_frac

Actuarial Notation	R_x or $\overline{R}_x$
Usage	ct.Rx[x]
Args	x: age as an integer number
Example	tv7377_ct.Rx[50]
Result	552381.6299290637

Examples

```
# Actuarial present value of a unitary due whole life annuity for (50) paid annually
tv7377_ct.Nx[50]/tv7377_ct.Dx[50]   # 22.55443277

# Actuarial present value of a whole life insurance for (50) with 100.000 m.u. capital.
    Payment is made in the moment of death
tv7377_ct_md = ct.CommutationFunctions(2, 0, 'q', soa.table_qx, 100, True)
100000*tv7377_ct_md.Mx[50]/tv7377_ct_md.Dx[50]    # 56330.616995

# Present value of an unitary whole life annuity due, paid quarterly to an individual aged
     x=65.25
x = 65.25
index_age = tv7377_ct_f4.age_to_index(int(x), x - int(x))  # 261
a = tv7377_ct_f4.Nx_frac[index_age] / tv7377_ct_f4.Dx_frac[index_age]
print(f'For age {x}, with index {index_age}, ax(4)={a}')
# For age 65.25, with index 261, ax(4)=56.9123257953868

## Actuarial present Value of unitary annuity due, paid semiannually with 10 terms for (35.5)
x = 35.5
n=10/2
index_age = tv7377_ct_f2.age_to_index(int(x), x - int(x))
index_age_end = tv7377_ct_f2.age_to_index(int(x+n), x+n - int(x+n))
```

```python
b = (tv7377_ct_f2.Nx_frac[index_age] - tv7377_ct_f2.Nx_frac[index_age_end])/tv7377_ct_f2.
    Dx_frac[index_age]
print(f'For age {x}, with index {index_age}, with semiannual payments until age {x+n}, with
    index {index_age_end}, ax:n={b}')
# For age 35.5, with index 71, with semiannual payments until age 40.5, with index 81,
    ax:n=9.542714980644465

## Actuarial present value of a whole life insurance for (50.75) with 100.000 m.u. capital.
x=50.75
index_age=tv7377_ct_f4.age_to_index(int(x), x-int(x))  # 203
b=100000*tv7377_ct_f4.Mx_frac[index_age]/tv7377_ct_f4.Dx_frac[index_age]
print(f'For age {x}, with index {index_age}, the risk premium is Ax={b}')
# For age 50.75, with index 203, the risk premium is Ax=56909.96956118816

## Actuarial present Value of a whole life annuity paid semiannually to an individual aged
     x = 35.5. Payments have a growth rate of 1%
x = 35.5
index_age = tv7377_ct_f2_g1.age_to_index(int(x), x - int(x))  # 151
a = tv7377_ct_f2_g1.Nx_frac[index_age] / tv7377_ct_f2_g1.Dx_frac[index_age]
print(f'For age {x}, with index {index_age}, Gax(2)={a}')
# For age 35.5, with index 71, Gax(2)=70.31380781464682
```

7.4 Life Annuities using Commutation Tables

Considering a Commutation Table *ct*, the following life annuities are defined as methods and the syntax for each type is described in Tables 16 (constant term life annuities) and 17 (increasing/decreasing life annuities).

Table 16: Actuarial Notation and Syntax Formula for Life Annuities in Commutation Tables

Notation	**Description**	**Syntax**
a_x	whole life annuity	ax(x,1)
$\ddot{a}_x$	whole life annuity due	aax(x,1)
${}_{t\vert}a_x$	t years deferred whole life annuity	t_ax(x,1,t)
${}_{t\vert}\ddot{a}_x$	t years deferred whole life annuity due	t_aax(x,1,t)
$a_x^{(m)}$	whole life annuity payable m times per year	ax(x,m)
$\ddot{a}_x^{(m)}$	whole life annuity due payable m times per year	aax(x,m)
${}_{t\vert}a_x^{(m)}$	t years deferred whole life annuity payable m times per year	t_ax(x,m,t)
${}_{t\vert}\ddot{a}_x^{(m)}$	t years deferred whole life annuity due payable m times per year	t_aax(x,m,t)
$a_{x:\overline{n}\vert}$	n year temporary life annuity	nax(x,n,1)
$\ddot{a}_{x:\overline{n}\vert}$	n year temporary life annuity due	naax(x,n,1)
${}_{t\vert}a_{x:\overline{n}\vert}$	t year deferred n year temporary life annuity	t_nax(x,n,1,t)
${}_{t\vert}\ddot{a}_{x:\overline{n}\vert}$	t year deferred n year temporary life annuity due	t_naax(x,n,1,t)
$a_{x:\overline{n}\vert}^{(m)}$	n year temporary life annuity payable m times per year	nax(x,n,m)
$\ddot{a}_{x:\overline{n}\vert}^{(m)}$	n year temporary life annuity due payable m times per year	naax(x,n,m)
${}_{t\vert}a_{x:\overline{n}\vert}^{(m)}$	t year deferred n year temporary life annuity payable m times per year	t_nax(x,n,m,t)
${}_{t\vert}\ddot{a}_{x:\overline{n}\vert}^{(m)}$	t year deferred n year temporary life annuity due payable m times per year	t_naax(x,n,m,t)

Table 17: Actuarial Notation and Syntax for Increasing/Decreasing Life Annuities in Commutation Tables

Notation	Description	Syntax
${}_{t\|}(Ia)^{(m)r}_{x:\overline{n}\|}$	t-years deferred n-year temporary increasing life annuity, payable m times per year. First payment C and increasing/decreasing amount r	t_nIax(x,n,m,t,C,r)
${}_{t\|}(I\ddot{a})^{(m)r}_{x:\overline{n}\|}$	t-years deferred n-year temporary increasing life annuity, payable m times per year. First payment C and increasing/decreasing amount r	t_nIaax(x,n,m,t,C,r)

For life annuities with terms varying geometrically, the Actuarial Table must be built with a growth rate g as defined in section 7.1, and the functions from Table 16 are applied.

Important Remark:
When using *CommutationTableFrac*, all the above methods for the computation of life annuities are available. However, since the acturial table is already fractional, parameter m should always be set to 1.

7.4.1 Examples

In this section, we illustrate the use of life annuities methods included in *CommutationTable* class.

Constant Term Whole Life Annuities

```
# Immediate Life Annuities
tv7377_ct.ax(x=50, m=1)  # 21.554432773700235
tv7377_ct.ax(x=50, m=4)  # 21.929432773700235

# Due Life Annuities
tv7377_ct.aax(x=50, m=1)  # 22.55443277370024
tv7377_ct.aax(x=50, m=4)  # 22.17943277370024

# Deferred Life Annuities
tv7377_ct.t_ax(x=50, m=1, defer=5)  # 16.899196591768252
tv7377_ct.t_ax(x=50, m=14, defer=5)  # 17.231374204075433

tv7377_ct.t_aax(x=50, m=1, defer=5)  # 17.78500355792074
tv7377_ct.t_aax(x=50, m=4, defer=5)  # 17.45282594561355
```

Remarks:

- ct.t_ax(x, m, defer=0) = ct.ax(x, m)
- ct.t_aax(x, m, defer=0) = ct.aax(x, m)

Constant Term Temporary Life Annuities

```
# Immediate Life Annuities
tv7377_ct.nax(x=50, n=10, m=1)  # 8.756215803256639
tv7377_ct.nax(x=50, n=10, m=4)  # 8.839775242816884

# Due Life Annuities
tv7377_ct.naax(x=50, n=10, m=1)  # 8.979040975417291
tv7377_ct.naax(x=50, n=10, m=4)  # 8.895481535857046

```

```python
# Deferred Life Annuities
tv7377_ct.t_nax(x=50, n=10, m=1, defer=5)  # 7.670292001795834
tv7377_ct.t_nax(x=50, n=10, m=4, defer=5)  # 7.750622055449898

tv7377_ct.t_naax(x=50, n=10, m=1, defer=5)  # 7.8845054782066715
tv7377_ct.t_naax(x=50, n=10, m=4, defer=5)  # 7.804175424552608
```

Remarks:

- ct.t_nax(x, m, defer=0) = ct.nax(x, m)
- ct.t_naax(x, m, defer=0) = ct.naax(x, m)

Life Annuities with Variable Terms - Increasing/Decreasing in Arithmetic Progression

```python
# Immediate
tv7377_ct.t_nIax(x=50, n=10, m=1, defer=0, first_amount=1, increase_amount=1)    # 46.33017
tv7377_ct.t_nIax(x=50, n=10, m=1, defer=0, first_amount=1, increase_amount=5)    # 196.62599
tv7377_ct.t_nIax(x=50, n=10, m=1, defer=0, first_amount=100, increase_amount=-5) # 687.75180

# Due
tv7377_ct.t_nIaax(x=50, n=10, m=1, defer=0, first_amount=1, increase_amount=1)    # 47.5374643
tv7377_ct.t_nIaax(x=50, n=10, m=1, defer=0, first_amount=1, increase_amount=5)    # 201.771158
tv7377_ct.t_nIaax(x=50, n=10, m=1, defer=0, first_amount=100, increase_amount=-5) # 705.111980
```

Remarks:

- ct.t_nIax(x, n, m, defer, 1, increase_amount=0) = ct.t_nax(x, n, m, defer)
- ct.t_nIaax(x, n, m, defer, 1, increase_amount=0) = ct.t_naax(x, n, m, defer)

Geometric Life Annuities

```python
from lifeActuary import commutation_table as ct

# reads SOA table
soa = rst.SoaTable('/soa_tables/' + 'TV7377' + '.xml')

# creates an actuarial table from qx of SOA table with geometric increase of 5% on payments
tv7377_ctg_inc = ct.CommutationFunctions(i=2, g=5, data_type='q', mt=soa.table_qx, perc=100,
    app_cont=False)


# creates an actuarial table from qx of SOA table with geometric decrease of 5% on payments
tv7377_ctg_dec = ct.CommutationFunctions(i=2, g=-5, data_type='q', mt=soa.table_qx, perc=100,
    app_cont=False)

# actuarial present value of a geometrically evolving life annuity for a 50 years old
    indidivual, for 10 years period
tv7377_ctg_inc.naax(50,10,1)  # 11.18091822195998
tv7377_ctg_dec.naax(50,10,1)  # 7.281682932595854
```

Present Value Function

```
# Present value of a series of cash-flows c=(100, -25, 120, 300, -50) paid to a age=35
    individual.
# Spot rates for periods of payments are given by (1.2%, 1.4%, 1.8%, 1.6%, 1.9%).

# 1 - Payments are made upon the survival of the individual. Survival probabilities are set by
     the chosen mortality table
pv1 = tv7377_ct.present_value(probs=None, age=35,
      spot_rates=[1.2, 1.4, 1.8, 1.6, 1.9], capital=[100, -25, 120, 300, -50])
print('Present Value:', pv1)
# 424.2408517830521

# 2 - Payments are certain
pv2 = tv7377_ct.present_value(probs=1, age=None,
      spot_rates=[1.2, 1.4, 1.8, 1.6, 1.9], capital=[100, -25, 120, 300, -50])
print('Present Value:', pv2)
# 425.750701233034

# 3 - Survival Probabilities are updated
x = 50
survprobs = [1-1.5*tv7377_ct.qx[x], 1-1.55*tv7377_ct.qx[x + 1], 1-1.6*tv7377_ct.qx[x + 2],
    1-1.65*tv7377_ct.qx[x + 3], 1-1.7*tv7377_ct.qx[x + 4]]
print('SurvProbs:', survprobs)
# [0.9943544272, 0.993714114965, 0.99293628512, 0.992097190855, 0.99121180988]

pv3 = tv7377_ct.present_value(probs=survprobs, age=None, spot_rates=[1.2, 1.4, 1.8, 1.6, 1.9],
     capital=[100, -25, 120, 300, -50])
print('Present Value:', pv3)
# 422.7070503910042
```

7.5 Life Insurance using Commutation Tables

Tables 18 and 19 resumes the available methods for evaluating life insurance, from a CommutationTable class. As usual in the actuarial notation, the capital letters with bar refer to payments due in the "moment of death" and the absense of bar refers to payments due in the end of the year in which the death occurs.
For life insurance with terms varying geometrically, the Actuarial Table must be built with a growth rate g, as defined in section 7.1, and the functions from Table 18 are applied.

7.5.1 Examples

In this section, examples on the use of methods for pricing life insurance, available in the *CommutationTable* class are illustrated.

Payments in the end of periods

```
# Pure Endowment Insurance
tv7377_ct.nEx(x=50, n=5)  # 0.8858069661524854
tv7377_ct.nEx(x=50, n=10) # 0.7771748278393479
tv7377_ct.nEx(x=80, n=10) # 0.2283081320230277

# Whole Life Insurance
tv7377_ct.Ax(x=50)                # 0.5577562201235239
tv7377_ct.t_Ax(x=50, defer=2)     # 0.550183040772438

```

Table 18: Actuarial Notation and Syntax Formula for Life Insurances - fixed capitals

Notation	Description	Syntax
${}_nE_x$	pure endowment	nEx(x,n)
A_x	whole life insurance (end of the year)	Ax(x)
$\bar{A}_x$	whole life insurance (moment of death)	Ax_(x)
${}_{t\|}A_x$	t years deferred whole life insurance (end of the year)	t_Ax(x,t)
${}_{t\|}\bar{A}_x$	t years deferred whole life insurance (moment of death)	t_Ax_(x,t)
$A^1_{x:\overline{n}\|}$	term life insurance (end of the year)	nAx(x,n)
$\bar{A}^1_{x:\overline{n}\|}$	term life insurance (moment of death)	nAx_(x,n)
${}_{t\|}A^1_{x:\overline{n}\|}$	t years deferred term life insurance (end of the year)	t_nAx(x,n,t)
${}_{t\|}\bar{A}^1_{x:\overline{n}\|}$	t years deferred term life insurance (moment of death)	t_nAx_(x,n,t)
$A_{x:\overline{n}\|}$	endowment insurance (end of the year)	nAEx(x,n)
$\bar{A}_{x:\overline{n}\|}$	endowment insurance (moment of death)	nAEx_(x,n)
${}_{t\|}A_{x:\overline{n}\|}$	t-years deferred endowment insurance (end of the year)	t_nAEx(x,n,t)
${}_{t\|}\bar{A}_{x:\overline{n}\|}$	t-years deferred endowment insurance (moment of death)	t_nAEx_(x,n,t)

Table 19: Actuarial Notation and Syntax Formula for Life Insurances - variable capitals

Notation	Description	Syntax
$(IA)_x$	whole life insurance with arithmetically increasing capitals (end of the year)	IAx(x)
$(I\bar{A})_x$	whole life insurance with arithmetically increasing capitals (moment of death)	IAx_(x)
$(IA)_{x:\overline{n}\|}$	term life insurance with arithmetically increasing capitals (end of the year)	nIAx(x,n)
$(I\bar{A})_{x:\overline{n}\|}$	term life insurance with arithmetically increasing capitals (moment of death)	nIAx_(x,n)
${}_{t\|}(IA)^r_{x:\overline{n}\|}$	t-years deferred term life insurance with capitals evolving arithmetically (increasing or decreasing). First Capital C and increase amount r (end of the year)	nIArx(x,n,t,C,r)
${}_{t\|}(I\bar{A})^r_{x:\overline{n}\|}$	t-years deferred term life insurance with capitals evolving arithmetically (increasing or decreasing). First Capital C and increase amount r (moment of death)	nIArx_(x,n,t,C,r)

```
# Temporary Life Insurance
tv7377_ct.nAx(x=50, n=10)            # 0.046765545191685375
tv7377_ct.t_nAx(x=50, n=10, defer=5) # 0.059615329779334834

# Endowment Insurance
tv7377_ct.nAEx(x=50, n=10)             # 0.8239403730310333
tv7377_ct.t_nAEx(x=50, n=10, defer=2)  # 0.7863040688470341
```

```
# Temporary Life Insurance with variable Capitals
# (arithmetic progression with first amount = rate of the progression)
tv7377_ct.IAx(x=50)          # 15.807431562003355
tv7377_ct.nIAx(x=50, n=10)  # 0.2751855520152587
tv7377_ct.nIAx(x=50, n=50)  # 15.702602747043013

# Temporary Life Insurance with variable Capitals (arithmetic progression)
# Increasing Capitals
tv7377_ct.nIArx(x=50, n=10, defer=0, first_amount=1000, increase_amount=50)
# 58.18654553286392
tv7377_ct.nIArx(x=50, n=10, defer=10,first_amount=1000, increase_amount=50)
#133.3402470815534

# Decreasing Capitals
tv7377_ct.nIArx(x=50, n=10, defer=0, first_amount=1000, increase_amount=-50)
# 35.344544850506836
tv7377_ct.nIArx(x=50, n=10, defer=10, first_amount=1000,increase_amount=-50)
# 28.027981923486493
```

Remarks:

- ct.t_Ax(x, defer=0) = ct.Ax(x)
- ct.t_nAx(x, n, defer=0) = ct.nAx(x, n)
- ct.nIAx(x, n) = ct.nIArx(x, n, defer=0,first_amount=1, increase_amount=1)

Payments in the middle of the year ("moment of death")

```
# Whole Life Insurance
tv7377_ct.Ax_(x=50)              # 0.5633061699539695
tv7377_ct.t_Ax_(x=50, defer=2)  # 0.5556576337284301

# Temporary Life Insurance
tv7377_ct.nAx_(x=50, n=10)             # 0.047230885460862126
tv7377_ct.t_nAx_(x=50, n=10, defer=5)  # 0.060208531750847685

# Endowment Insurance
tv7377_ct.nAEx_(x=50, n=10)               # 0.8244057133002101
tv7377_ct.t_nAEx_(x= 50, n=10, defer=2)  # 0.7868146552887256
tv7377_ct.t_nAEx_(x=50, n=10, defer=10)  # 0.6442926524583354

# Temporary Life Insurance with variable Capitals
tv7377_ct.IAx_(x=50)          # 15.964723312327344
tv7377_ct.nIAx_(x=50, n=10)  # 0.2779237841543988
tv7377_ct.nIAx_(x=50, n=50)  # 15.858851398889882

tv7377_ct.nIArx_(x=50, n=10, defer=0, first_amount=1000, increase_amount=50)
# 58.765530395538875
tv7377_ct.nIArx_(x=50, n=10, defer=10, first_amount=1000, increase_amount=50)
# 134.66704838825683
tv7377_ct.nIArx_(x=50, n=10, defer=0, first_amount=1000, increase_amount=-50)
# 35.69624052618538
tv7377_ct.nIArx_(x=50, n=10, defer=10, first_amount=1000, increase_amount=-50)
# 28.306874184857506
```

Remarks:

- ct.t_Ax_(x, defer=0) = ct.Ax_(x)
- ct.t_nAx_(x, n, defer=0) = ct.nAx_(x, n)
- ct.nIAx_(x, n) = ct.nIArx_(x, n, defer=0,first_amount=1, increase_amount=1)

7.6 Export Actuarial Table to Excel

Actuarial Tables may be exported to Excel files, using the standard Python functions.
For example, the following instruction produces a xlsx file with actuarial commutation symbols for all integer and half year ages, for TV7377 mortality table:

```
tv7377_ct_f2.df_commutation_table_frac().to_excel(excel_writer='frac2' + '.xlsx',
     sheet_name='frac2', index=False, freeze_panes=(1, 1))
```

Figure 1 illustrates an excerpt of the the produced excel file for TV7377 actuarial table for years fractioned in semesters.

x	lx	dx	qx	px	Dx	Nx	Sx	Cx	Mx	Rx
0	100000	584	0,00584	0,99416	100000	7794804	464818252,1	578,2462	23202,03	3215202
0,5	99416	584	0,005874	0,994126	98436,51	7694804	457023448,5	572,549	22623,79	3192000
1	98832	48	0,000486	0,999514	96894,12	7596367	449328644,9	46,59518	22051,24	3169376
1,5	98784	48	0,000486	0,999514	95892,88	7499473	441732277,7	46,1361	22004,64	3147325
2	98736	29,5	0,000299	0,999701	94901,96	7403580	434232804,7	28,07512	21958,51	3125320
2,5	98706,5	29,5	0,000299	0,999701	93938,87	7308678	426829224,6	27,79851	21930,43	3103362
3	98677	23	0,000233	0,999767	92985,54	7214739	419520546,4	21,45988	21902,63	3081431
3,5	98654	23	0,000233	0,999767	92047,95	7121754	412305807,1	21,24845	21881,17	3059529
...	...	...	...	...	...	...	...	...	...	...
104	12	4	0,333333	0,666667	1,530254	3,533177	7,265558967	0,505059	1,495443	3,461593
104,5	8	4	0,5	0,5	1,010118	2,002923	3,732382231	0,500083	0,990384	1,96615
105	4	1,5	0,375	0,625	0,500083	0,992805	1,729458998	0,185683	0,490301	0,975766
105,5	2,5	1,5	0,6	0,4	0,309472	0,492723	0,736653597	0,183854	0,304618	0,485465
106	1	0,5	0,5	0,5	0,122569	0,18325	0,24393104	0,060681	0,120764	0,180847
106,5	0,5	0,5	1	0	0,060681	0,060681	0,060680858	0,060083	0,060083	0,060083
107	0	0	1	0	0	0	0	0	0	0

Figure 1: Excerpt of TV7377 semiannually fractional actuarial table

8 Financial Annuities

8.1 Class Annuities_Certain

---

class *Annuities_Certain*
This class instantiates the methods for the computation of financial annuities, for a given interest rate and a chosen frequency of payments m (in each period of the interest rate).

Usage

```
AnnuitiesCertain(interest_rate, m=1)
```

Description

Initializes the AnnuitiesCertain class so that we can compute the present value of financial annuities.

Parameters

interest_rate	interest rate, in percentage (e.g. use 5 for 5%)
m	frequency of payments, in each period of the interest rate

Examples

```
from lifeActuary import annuities_certain as ac

# instantiates a methods for computing financial annuities with a 5% annual interest rate,
    with annual payments
i1=ac.Annuities_Certain(5,1)

# instantiates a methods for computing financial annuities with a 5% annual interest rate,
    with quarterly payments
i4=ac.Annuities_Certain(5,4)
```

The following methods are available after instantiating the class:

8.1.1 im

Actuarial Notation	i_m
Definition	$m\left[(1+i)^{1/m}-1\right]$
Usage	irate.im
Example	i4.im
Result	0.04908893771615741

8.1.2 vm

Actuarial Notation	v_m
Definition	$\frac{1}{1+\frac{i_m}{m}}$
Usage	irate.vm
Example	i4.vm
Result	0.9878765474230741

8.1.3 dm

Actuarial Notation	d_m
Definition	$\left(1 - \frac{i}{1+i}\right)^{1/m} - 1$
Usage	irate.dm
Example	i4.dm
Result	0.0484938103077039

8.2 Constant Terms Financial Annuities

8.2.1 an

Actuarial Notation: $a_{\overline{n}|}$ and $a_{\overline{n}|}^{(m)}$ or $a_{\overline{\infty}|}$ and $a_{\overline{\infty}|}^{(m)}$

Usage

```
an(terms)
```

Description: Returns the present value of an immediate n term financial annuity with payments equal to 1. Payments are made in the end of the periods. In fractional annuities, payments of $1/m$ are made m times per year at the end of the periods.

Parameters

terms	number of periods (measured in periods of the interest rate) (terms=None) or (terms=0) returns the present value of the perpetual annuity

Examples

```
i1.an(10) # 7.721734929184813
i4.an(10) # 7.86504586209782

i1.an(None) # 19.999999999999982
i4.an(0) # 20.371188429095998
```

8.2.2 aan

Actuarial Notation: $\ddot{a}_{\overline{n}|}$ and $\ddot{a}_{\overline{n}|}^{(m)}$ or $\ddot{a}_{\overline{\infty}|}$ and $\ddot{a}_{\overline{\infty}|}^{(m)}$

Usage

```
aan(terms)
```

Description: Returns the present value of a due n term financial annuity with payments equal to 1. Payments are made in the beginning of periods. In fractional annuities, payments of $1/m$ are made m times per year at the end of the periods.

Parameters

terms	number of periods (measured in periods of the interest rate) (terms=None) or (terms=0) returns the present value of the perpetual annuity

Examples

```
i1.aan(10) # 8.107821675644054
i4.aan(10) # 7.9615675487126305

i1.aan(None) # 20.999999999999982
i4.aan(0) # 20.621188429095998
```

8.3 Variable Terms Financial Annuities

In this section we present functions that allow the computation of the present value of financial annuities with variable terms, for the particular cases where the terms evolve in arithmetic or geometric progressions.
For both cases, functions are defined in a general way such that the first term may differ from the rate of growth and the same function allows for increasing and decreasing terms.
For annuities with payments more frequent than the interest rate period, two approaches are considered and corresponding functions are developed: (1) payments level within each interest period and increase/decrease from one interest period to the next and (2) payments increase in each payment period.

Annuities with terms evolving Arithmetically

8.3.1 Ian

Actuarial Notation: $(Ia)_{\overline{n}|}$ and $(Ia)_{\overline{n}|}^{(m)}$ or $(Da)_{\overline{n}|}$ and $(Da)_{\overline{n}|}^{(m)}$

Usage

```
Ian(terms, payment=1, increase=1)
```

Description: Returns the present value of an immediate n term financial annuity with payments increasing/decreasing arithmetically. Payments are made in the end of the periods. First payment and increase amount may differ. In fractional annuities, payments level within each interest period and increase/decrease from one interest period to the next.

Parameters

terms	number of periods (measured in periods of the interest rate)
payment	amount of the first payment
increase	increase amount of payments (>0 for increasing annuities and <0 for decreasing annuities)

Actuarial Formula
For C - first payment , r - rate of increasing/decreasing, i-annual interest rate, n - number of terms, m-frequency of payments

$$_{(C,r)}(Ia)_{\overline{n}|}^{(m)} = C\, a_{\overline{n}|}^{(m)} + r\, \frac{a_{\overline{n}|i} - nv^n}{i^{(m)}}$$

Examples

```python
a4 = ac.Annuities_Certain(interest_rate=5, m=12)
r4 = a4.Ian(terms=20, payment=2000 * 12, increase=400 * 12)
print(r4)
#789369.5624059099
```

8.3.2 Iaan

Actuarial Notation: $I\ddot{a}_{\overline{n|}}$ and $I\ddot{a}_{\overline{n|}}^{(m)}$ or $D\ddot{a}_{\overline{n|}}$ and $D\ddot{a}_{\overline{n|}}^{(m)}$

Usage

```python
Iaan(terms, payment=1, increase=1)
```

Description: Returns the present value of a due n term financial annuity with payments increasing/decreasing arithmetically. Payments are made in the beginning of the periods. First payment and increase amount may differ. In fractional annuities, payments level within each interest period and increase/decrease from one interest period to the next.

Parameters

terms	number of periods (measured in periods of the interest rate)
payment	amount of the first payment
increase	increase amount of payments (> 0 for increasing annuities and < 0 for decreasing annuities)

Examples

```python
a5 = ac.Annuities_Certain(interest_rate=2, m=2)
r5 = a5.Iaan(terms=2, payment=1, increase=1)
print(r5)
# 2.946198813622495
```

8.3.3 Iman

Actuarial Notation: $(I^{(m)}a)_{\overline{n|}}$ and $(I^{(m)}a)_{\overline{n|}}^{(m)}$ or $(D^{(m)}a)_{\overline{n|}}$ and $(D^{(m)}a)_{\overline{n|}}^{(m)}$

Usage

```python
Iman(terms, payment=1, increase=1)
```

Description: Returns the present value of an immediate n-term financial annuity with payments increasing/decreasing arithmetically. Payments are made in the end of the periods. First payment and increase amount may differ. In fractional annuities, payments increase in each payment period.

Parameters

terms	number of periods (measured in periods of the interest rate)
payment	amount of the first payment
increase	increase amount of payments (> 0 for increasing annuities and < 0 for decreasing annuities)

Actuarial Formula

For C - first payment , r - rate of increasing/decreasing, i-annual interest rate, n - number of terms, m-frequency of payments

$$_{(C,r)}(I^{(m)}a)^{(m)}_{\overline{n}|} = C\, a^{(m)}_{\overline{n}|} + rm\,\frac{a^{(m)}_{\overline{n}|i} - nv^n}{i^{(m)}}$$

Examples

```python
a3 = ac.Annuities_Certain(interest_rate=3.3, m=12)
r3 = a3.Iman(terms=8, payment=25 * 12, increase=2 * 12)
print(r3)
# 9781.284321297218
```

Annuities with terms evolving Geometrically

For computing the present value of geometric financial annuities, let us consider the interest rate i_g which reflects the annual interest rate of the annuity, i, and the growth rate g. For these cases, let us define

$$i_g = \frac{1-g}{1+i} \qquad \text{and} \qquad v_g = \frac{1+g}{1+i}.$$

8.3.4 Gan

Actuarial Notation: $_g(Ga)_{\overline{n}|}$ and $_g(Ga)^{(m)}_{\overline{n}|}$

Usage

```python
Gan(terms,payment=1, grow=0)
```

Description: Returns the present value of an immediate n term financial annuity with payments increasing/decreasing geometrically. Payments are made in the end of the periods. In fractional annuities, payments level within each interest period and increase/decrease from one interest period to the next.

Parameters

terms	number of periods (measured in periods of the interest rate)
payment	amount of the first payment
grow	rate of growing of payments, in percentage

Actuarial Formula

For C - first payment , g - rate of increasing/decreasing, i-annual interest rate, n - number of terms, m-frequency of payments

$$ {}_{(C,g)}(Ga)^{(m)}_{\overline{n}|} = \begin{cases} \frac{C}{m}\,\frac{1}{(1+g)^{1/m}}\,a^{(m)}_{\overline{n}|\,i_g} & , \quad i \neq g \\ \\ Cnv^{1/m} & , \quad i = g \end{cases} $$

Examples

```
g1=ac.Annuities_Certain(interest_rate=5, m=2)
g1.Gan(terms=5,payment=10,grow=10)
# 108.60048398210647
```

8.3.5 Gaan

Actuarial Notation: ${}_{g}(G\ddot{a})^{(m)}_{\overline{n}|}$

Usage

```
Gaan(terms,payment=1, grow=0)
```

Description: Returns the present value of an immediate n term financial annuity with payments increasing/decreasing geometrically. Payments are made in the end of the periods. In fractional annuities, payments level within each interest period and increase/decrease from one interest period to the next.

Parameters

terms	number of periods (measured in periods of the interest rate)
payment	amount of the first payment
grow	rate of growing of payments, in percentage

Examples

```
g2=ac.Annuities_Certain(interest_rate=5, m=4)
g2.Gaan(terms=5,payment=100,grow=10)
# 2185.9539086346845
```

8.3.6 Gman

Actuarial Notation: ${}_{g}(G^{(m)}a)_{\overline{n}|}$ or ${}_{g}(G^{(m)}a)^{(m)}_{\overline{n}|}$

Usage

```
Gman(terms,payment=1, grow=0)
```

Description: Returns the present value of an immediate n term financial annuity with payments increasing/decreasing geometrically. Payments are made in the end of the periods. In fractional annuities, payments increase in each payment period.

Parameters

terms	number of periods (measured in periods of the interest rate)
payment	amount of the first payment
grow	rate of growing of payments, in percentage

Actuarial Formula

For C - first payment , g - rate of increasing/decreasing, i-annual interest rate, n - number of terms, m-frequency of payments

$$_{(C,g)}(Ga)^{(m)}_{\overline{n}|} = \begin{cases} C a^{(m)}_{\overline{1}|\,i} \times \ddot{a}_{\overline{n}|\,i_g} & , \quad i \neq g \\ \\ C\, n\, a^{(m)}_{\overline{1}|\,i} & , \quad i = g \end{cases}$$

Examples

```
g3=ac.Annuities_Certain(interest_rate=5, m=2)
g3.Gman(terms=5,payment=10,grow=10)
# 53.022051853437205
```

8.3.7 Gmaan

Actuarial Notation: $_g(G^{(m)}\ddot{a})_{\overline{n}|}$ or $_g(G^{(m)}\ddot{a})_{\overline{n}|}$

Usage

```
Gmaan(terms,payment=1, grow=0)
```

Description: Returns the present value of an immediate n term financial annuity with payments increasing/decreasing geometrically. Payments are made in the beginning of the periods. In fractional annuities, payments increase in each payment period.

Parameters

terms	number of periods (measured in periods of the interest rate)
payment	amount of the first payment
grow	rate of growing of payments, in percentage

Examples

```
g4=ac.Annuities_Certain(interest_rate=5, m=2)
g4.Gmaan(terms=5,payment=10,grow=10)
#  51.80299115517991
```

9 Examples

In this section we present some interesting and more complete examples of application, for which the use of lifeActuary package is helpful and provides "easy to use" solutions and functions.

9.1 Survival Probabilities

Example 1

Consider a 50 year old individual and the TV7377 mortality table.

1. Determine the probabilities of (50) being alive in the end of each month of the following ten years, considering the Uniform Distribution of Death approximation.

2. Determine the probabilities of (50) not surviving up to the end of each month of the following ten years, considering the Uniform Distribution of Death approximation.

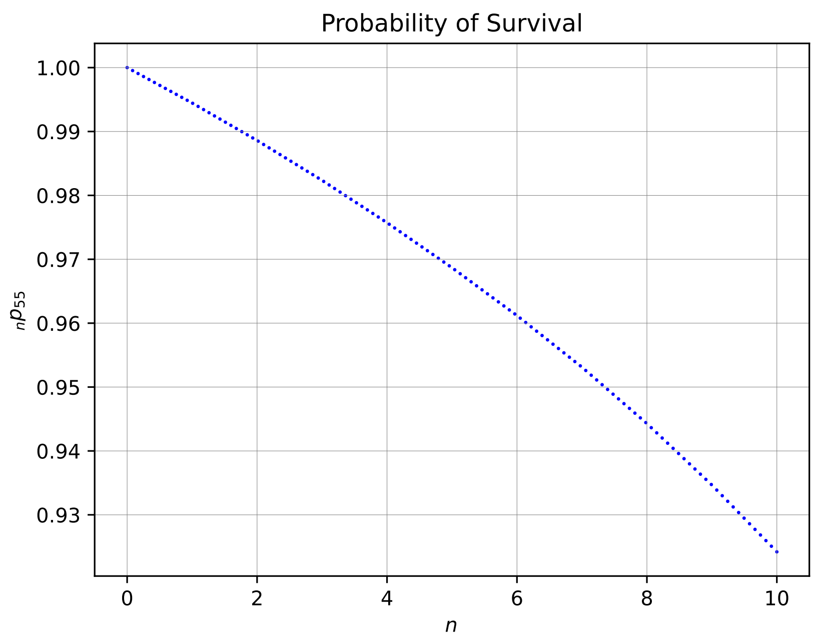
3. Build a dataframe with ages and estimated probabilities.

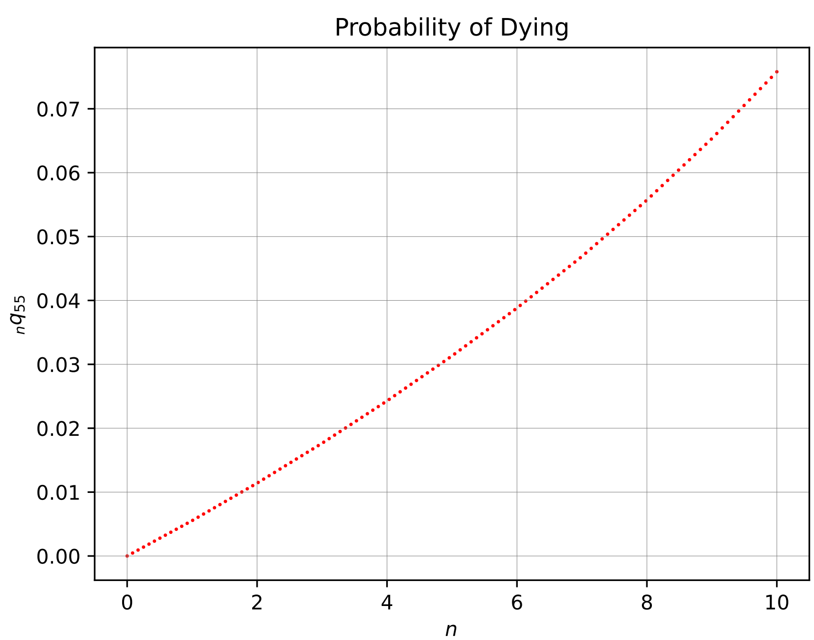
4. Export data to an Excel file.

5. Plot the estimated probabilities in a scatterplot.

```
from soa_tables import read_soa_table_xml as rst
from lifeActuary import mortality_table

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

mt = rst.SoaTable('/soa_tables/TV7377' + '.xml')
lt = mtable.MortalityTable(mt=mt.table_qx)

# Question 1
x = 55
n = np.linspace(0, 10, 10*12)
sprobs = [lt.npx(x=x, n=i, method='udd') for i in n]
dprobs = [lt.nqx(x=x, n=i, method='udd') for i in n]
ages = x + n
df = pd.DataFrame.from_dict({'n': n, 'x': ages, 'npx': sprobs, 'nqx': dprobs})

# Question 2
df.to_excel(excel_writer='example1.xlsx', sheet_name='example1', index=False, freeze_panes=(1,
    1))

# Question 3
plt.scatter(n, sprobs, s=.5, color='blue')

plt.xlabel(r'$n$')
plt.ylabel(r'${}_{n}p_{55}$')
plt.title('Probability of Survival')
plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
plt.savefig('example1s' + '.eps', format='eps', dpi=3600)
plt.show()
```

```
plt.scatter(n, dprobs, s=.5, color='red')

plt.xlabel(r'$n$')
plt.ylabel(r'${}_{n}q_{55}$')
plt.title('Probability of Dying')
plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
plt.savefig('example1d' + '.eps', format='eps', dpi=3600)
plt.show()
```





9.2 Life Tables and Life Annuities

Example 2

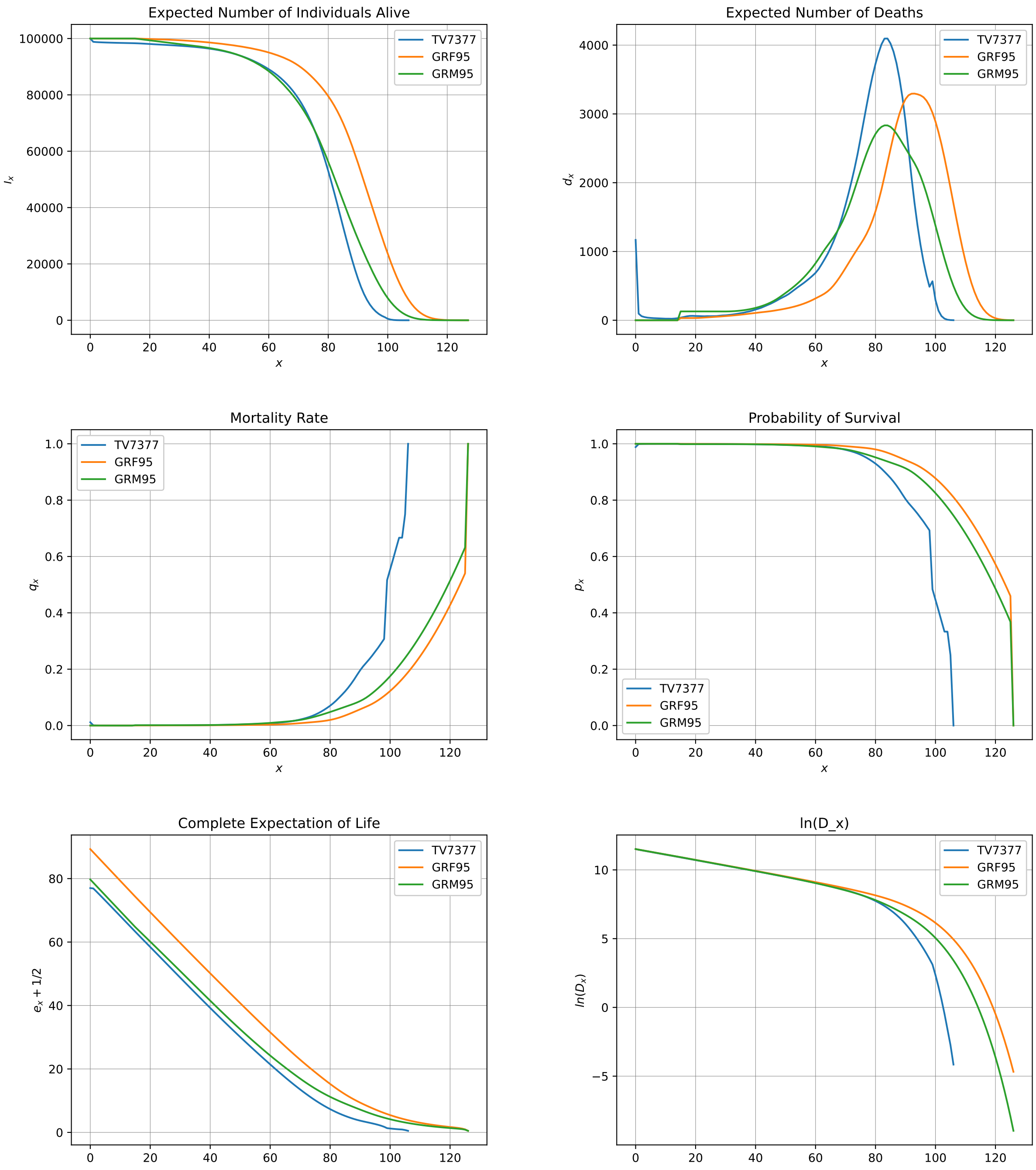
In this example, let us develop the Python code for answering the following questions:

1. Import mortality tables TV7377, GRF95 and GRM95, from SOA Mortality Tables, with the columns x, l_x, p_x, q_x, e_x.
2. Construct an actuarial table considering a technical rate of interest of 4% *per annum*, and append the columns D_x and N_x to the tables produced in the previous question.
3. Plot the l_x, q_x, p_x, e_x and $\ln(D_x)$, comparing, in the same graph, the values of each mortality table.
4. Determine the net single premium (risk single premium) of a whole life annuity immediate, if someone 55 years old today, wants to receive 1000€ per year considering that:
   (a) The contract is paid at single premium.
   (b) The contract is paid at level premiums during 5 years.
5. Determine the net single premium (risk single premium) of a 10 years temporary due life annuity, if someone 55 years old today, wants to receive 1000 m.u. per year.

```
from soa_tables import read_soa_table_xml as rst
from lifeActuary import mortality_table as mt, commutation_table as ct

import numpy as np
import os
import sys
import matplotlib.pyplot as plt

this_py = os.path.split(sys.argv[0])[-1][:-3]

def parse_table_name(name):
    return name.replace(' ', '').replace('/', '')

# Question 1
table_names = ['TV7377', 'GRF95', 'GRM95']
mt_lst = [rst.SoaTable('soa_tables/' + name + '.xml') for name in table_names]
lt_lst = [mtable.MortalityTable(mt=mt.table_qx) for mt in mt_lst]


# Question 2
interest_rate = 4
ct_lst = [ct.CommutationFunctions(i=interest_rate, g=0, mt=mt.table_qx) for mt in mt_lst]

for idx, lt in enumerate(lt_lst):
    name = parse_table_name(mt_lst[idx].name)
    lt.df_life_table().to_excel(excel_writer=name + '.xlsx', sheet_name=name, index=False,
    freeze_panes=(1, 1))
    ct_lst[idx].df_commutation_table().to_excel(excel_writer=name + '_comm' + '.xlsx',
    sheet_name=name, index=False, freeze_panes=(1, 1))

```

```
# Question 3
'''
Plot lx
'''
fig, axes = plt.subplots()
for idx, lt in enumerate(lt_lst):
    ages = np.arange(0, lt.w + 2)
    plt.plot(ages, lt.lx, label=table_names[idx])
plt.xlabel(r'$x$')
plt.ylabel(r'$l_x$')
plt.title('Expected Number of Individuals Alive')
plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
plt.legend()
plt.savefig(this_py + 'lx' + '.eps', format='eps', dpi=3600)
plt.show()

'''
Plot dx
'''
fig, axes = plt.subplots()
for idx, lt in enumerate(lt_lst):
    ages = np.arange(0, lt.w + 1)
    plt.plot(ages, lt.dx, label=table_names[idx])
plt.xlabel(r'$x$')
plt.ylabel(r'$d_x$')
plt.title('Expected Number of Deaths')
plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
plt.legend()
plt.savefig(this_py + 'dx' +'.eps', format='eps', dpi=3600)
plt.show()

'''
Plot qx
'''
fig, axes = plt.subplots()
for idx, lt in enumerate(lt_lst):
    ages = np.arange(0, lt.w + 1)
    plt.plot(ages, lt.qx, label=table_names[idx])
plt.xlabel(r'$x$')
plt.ylabel(r'$q_x$')
plt.title('Mortality Rate')
plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
plt.legend()
plt.savefig(this_py + 'qx' +'.eps', format='eps', dpi=3600)
plt.show()

'''
Plot px
'''
fig, axes = plt.subplots()
for idx, lt in enumerate(lt_lst):
    ages = np.arange(0, lt.w + 1)
    plt.plot(ages, lt.px, label=table_names[idx])
plt.xlabel(r'$x$')
plt.ylabel(r'$p_x$')
plt.title('Probability of Survival')
plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
```

```
plt.legend()
plt.savefig(this_py + 'px' +'.eps', format='eps', dpi=3600)
plt.show()

'''
Plot ex
'''
fig, axes = plt.subplots()
for idx, lt in enumerate(lt_lst):
    ages = np.arange(0, lt.w + 1)
    plt.plot(ages, lt.ex, label=table_names[idx])
plt.xlabel(r'$x$')
plt.ylabel(r'${e}_{x}+1/2$')
plt.title('Complete Expectation of Life')
plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
plt.legend()
plt.savefig(this_py + 'ex' + '.eps', format='eps', dpi=3600)
plt.show()

'''
Plot ln(Dx)
'''
fig, axes = plt.subplots()
for idx, lt in enumerate(ct_lst):
    ages = np.arange(0, lt.w + 1)
    plt.plot(ages, np.log(lt.Dx), label=table_names[idx])
plt.xlabel(r'$x$')
plt.ylabel(r'$ln(D_x)$')
plt.title(r'ln(D_x)')
plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
plt.legend()
plt.savefig(this_py +'lnDx' + '.eps', format='eps', dpi=3600)
plt.show()

# Question 4
for idx, ct in enumerate(ct_lst):
    print(table_names[idx] + ": " + f'{round(1000 * ct.ax(x=55, m=1), 2):,}')
print()

# Question 5
for idx, ct in enumerate(ct_lst):
    print(table_names[idx]+":"+f'{round(1000 * ct.ax(x=55, m=1) / ct.naax(x=55, n=5, m=1), 2)
    :,}')

# Consult the values used each computation (Nx, Dx)
for idx, ct in enumerate(ct_lst):
    print(ct.msn)
    print()

# Consult the values used in the computation, only for TV7377
ct_lst[0].msn
```



Question 4 a): $1000a_{55} = 1000\frac{N_{56}}{D_{55}}$

TV7377: 14,979.28 **GRF95:** 18,019.96 **GRM95:** 15,531.28

Question 4 b): $P\ \ddot{a}_{55:\overline{5}|} = 1000\frac{N_{56}}{D_{55}}$

TV7377: 3,272.3 **GRF95:** 3,910.88 **GRM95:** 3,398.66

Question 5: $1000a_{55:\overline{10}|} = 1000\frac{N_{56}-N_{66}}{D_{55}}$

TV7377: 3,272.3 **GRF95:** 3,910.88 **GRM95:** 3,398.66

9.3 Life Tables and Life Insurance

Example 3

Consider a Pure Endowment insurance with duration 10 years, if someone 55 years old today, subscribes 1000€ , considering $i = 4\%/\text{year}$. Considering the mortatily tables TV7377, GRF95 and GRM95 and using the Commutation Table functions:

1. Determine the net single premium (risk single premium).
2. Determine the annual premium paid during 5 years if the insured is alive.
3. Evaluate the price of contract, including the refund of all premiums paid at the end of the year of death and at the end of the term.

```python
from soa_tables import read_soa_table_xml as rst
from lifeActuary import mortality_table as mtable, commutation_table as ct

import numpy as np

table_names = ['TV7377', 'GRF95', 'GRM95']
interest_rate = 4
mt_lst = [rst.SoaTable('soa_tables/' + name + '.xml') for name in table_names]
lt_lst = [mtable.MortalityTable(mt=mt.table_qx) for mt in mt_lst]
ct_lst = [ct.CommutationFunctions(i=interest_rate, g=0, mt=mt.table_qx) for mt in mt_lst]

# General Information
x = 55
capital = 1000
term = 10
term_annuity = 5
# pure endowment
pureEndow = [ct.nEx(x=x, n=term) for ct in ct_lst]
# temporary annuity due
tad = [ct.naax(x=x, n=term_annuity, m=1) for ct in ct_lst]

### Question 1
print('\nnet single premium')
for idx, ct in enumerate(ct_lst):
    print(table_names[idx] + ": " + f'{round(capital * pureEndow[idx], 5):,}')

### Question 2
print('\nlevel premium')
for idx, ct in enumerate(ct_lst):
    print(table_names[idx] + ":" + f'{round(capital * pureEndow[idx] / tad[idx], 5):,}')

# show the annuities
print('\nannuities')
for idx, ct in enumerate(ct_lst):
    print(table_names[idx] + ":" + f'{round(tad[idx], 5):,}')

### Question 3

## Refund of Net Single Premium
```

```python
print('\nSingle Net Risk Premium Refund at End of the Year of Death')
termLifeInsurance = [ct.nAx(x=x, n=term) for ct in ct_lst]
pureEndow_refund = [ct.nEx(x=x, n=term) / (1 - ct.nAx(x=x, n=term)) for ct in ct_lst]

print('\nTerm Life Insurance')
for idx, ct in enumerate(ct_lst):
    print(table_names[idx] + ":" + f'{round(termLifeInsurance[idx], 5):,}')

print('\nSingle Net Premium Refund Cost at End of the Year of Death')
for idx, ct in enumerate(ct_lst):
    print(table_names[idx] + ":" + f'{round(capital * pureEndow[idx] / (1 - termLifeInsurance[idx]), 5):,}')

print('Refund Cost at End of the Year of Death')
for idx, ct in enumerate(ct_lst):
    print(table_names[idx] + ":" + f'{round(capital * (pureEndow_refund[idx] - pureEndow[idx]), 5):,}')

print('\nSingle Net Risk Premium Refund at End of the Term')
pureEndow_refund_eot = [ct.nEx(x=x, n=term) / (1 - (1 + interest_rate / 100) ** (-term) + ct.nEx(x=x, n=term))
    for ct in ct_lst]

for idx, ct in enumerate(ct_lst):
    print(table_names[idx] + ":" + f'{round(capital * pureEndow_refund_eot[idx], 5):,}')

print('Refund Cost at End of the the Term')
for idx, ct in enumerate(ct_lst):
    print(table_names[idx] + ":" + f'{round(capital * (pureEndow_refund_eot[idx] - pureEndow[idx]), 5):,}')


## Refund of Net Level Premiums

print('\nLeveled Net Risk Premium Refund at End of the Year of Death')
tli_increasing = [ct.nIAx(x=x, n=term_annuity) for ct in ct_lst]
tli_deferred = [ct.t_nAx(x=x, n=term - term_annuity, defer=term_annuity) for ct in ct_lst]
pureEndow_leveled_refund = [
    pureEndow[idx_ct] / (tad[idx_ct] - tli_increasing[idx_ct] - term_annuity * tli_deferred[idx_ct])
    for idx_ct, ct in enumerate(ct_lst)]

for idx, ct in enumerate(ct_lst):
    print(table_names[idx] + ":" + f'{round(capital * pureEndow_leveled_refund[idx], 5):,}')
```

Solutions:

Question 1:

$$PP = 1000\,{}_{10}E_{55}$$

TV7377: 624.36092 **GRF95:** 653.67485 **GRM95:** 615.65987

Question 2:

$$P\ddot{a}_{55} = 1000\, \frac{{}_{10}E_{55}}{\ddot{a}_{55:\overline{10}|}}$$

TV7377: 136.39495 **GRF95:** 141.86738 **GRM95:** 134.72276

Question 3:

Single Premium with Refund paid at the end of the Year of Death

$$P = 1000\, \frac{{}_{10}E_{55}}{1 - A^{1}_{55:\overline{10}|}}$$

TV7377: 664.2747 **GRF95:** 670.91998 **GRM95:** 662.20316

Single Premium with Refund paid at the end of the contract

$$P = 1000\, \frac{{}_{10}E_{55}}{1 - v^{10}\ {}_{10}q_{55}}$$

TV7377: 658.0555 **GRF95:** 668.30356 **GRM95:** 654.89063

Level Premium with Refund paid at the end of the year of death

$$P = 1000\, \frac{{}_{10}E_{55}}{\ddot{a}_{55:\overline{10}|} - (IA)_{55:\overline{10}|}}$$

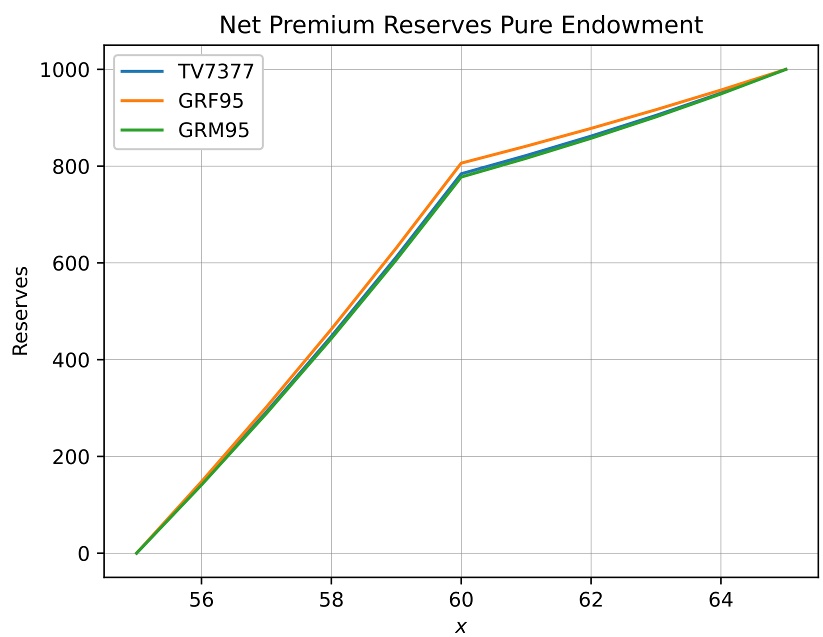
TV7377: 144.14453 **GRF95:** 145.18891 **GRM95:** 143.8195

Example 4

For the contract in Example 3.2, estimate the net premium reserves until the end of the contract, export the estimates to and EXcel file and plot the evolution of the reserves in a graph.

```

## Net premium reserves path

l0 = 1000
reserves_dict = {'table': [], 'x': [], 'insurer': [], 'insured': [], 'reserve': []}
fund_dict = {'lx': [], 'claim': [], 'premium': [], 'fund': []}

# Expected reserves value, that is, considering the survivorship of the group
expected_reserve_dict = {'insurer_exp': [], 'insured_exp': [], 'reserve_exp': []}
ages = range(x, x + term + 1)
print('\n\n Net Premium reserves \n\n')
for idx_clt, clt in enumerate(ct_lst):
    premium_unit = pureEndow[idx_clt]
    premium_capital = capital * premium_unit
    premium_unit_leveled = premium_unit / tad[idx_clt]
    premium_leveled = premium_unit_leveled * capital
    for age in ages:
        # reserves
        reserves_dict['table'].append(table_names[idx_clt])
        reserves_dict['x'].append(age)
        insurer_liability = clt.nEx(x=age, n=term - (age - x)) * \
                            capital
        reserves_dict['insurer'].append(insurer_liability)
        tad2 = clt.naax(x=age, n=term_annuity - (age - x))
        insured_liability = premium_leveled * tad2
        reserves_dict['insured'].append(insured_liability)
        reserve = insurer_liability - insured_liability
        reserves_dict['reserve'].append(reserve)

        prob_survival = clt.npx(x=x, n=age - x)
        lx = l0 * prob_survival
        expected_reserve_dict['insurer_exp'].append(insurer_liability*lx)
        expected_reserve_dict['insured_exp'].append(insured_liability*lx)
        expected_reserve_dict['reserve_exp'].append(reserve*prob_survival*lx)

        # fund
        fund_dict['lx'].append(lx)
        qx_1 = clt.nqx(x=age, n=1)
        claim = 0
        if age == x + term:
            claim = capital * lx
        fund_dict['claim'].append(claim)
        premium = 0
        if tad2 > 0:
            premium = premium_leveled*lx
        fund_dict['premium'].append(premium)
        if age == x:
            fund = lx * premium_leveled
        else:
            fund = fund_dict['fund'][-1] * (1 + interest_rate / 100) - claim + premium
        fund_dict['fund'].append(fund)
```

```

reserves_df = pd.DataFrame(reserves_dict)
expected_reserve_df = pd.DataFrame(expected_reserve_dict)
fund_df = pd.DataFrame(fund_dict)
name = 'pureEndowment_55_1'
reserves_df.to_excel(excel_writer=name + '_netReserves' + '.xlsx', sheet_name=name, index=
    False, freeze_panes=(1, 1))

'''
plot the reserves
'''
for idx_clt, clt in enumerate(ct_lst):
    plt.plot(ages, reserves_df.loc[reserves_df['table'] == table_names[idx_clt]]['reserve'],
             label=table_names[idx_clt])

plt.xlabel(r'$x$')
plt.ylabel('Reserves')
plt.title('Net Premium Reserves Pure Endowment')
plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
plt.legend()
plt.show()

# save the graph
plt.savefig(this_py + '.eps', format='eps', dpi=3600)
```



9.4 Life Annuities and Life Insurance

Example 5

Robert and his wife, Silvia, will turn 66 this year. During the last years, they have invested in retirement savings products having each accumulated 200,000 m.u. in their fund (they own individual and independent funds). At the end of this year, they intend to invest the amount of each fund in the purchase of whole life annuities. Each of them has very different options regarding the way in which they intend to receive income:

– Silvia intends to buy a whole life annuity, paid quarterly, in which the terms of even order are the double of the odd-numbered terms, with payments made at the end of each trimester.

– Robert intends to acquire a whole life annuity, paid once a year, with the first term paid immediately, whose terms grow by 200 m.u. each year. Additionally, to celebrate his 70-th birthday, Robert wants to double the amount of that year. Additionally, Robert wants to buy a 10 years term life insurance, with a capital equal to the first amount of his life annuity.

Each income will be paid individually to each member of the couple.

Determine the amount of the first term of Robert and Silvia life annuity, considering a TV7377 for Robert and GRF95 for Silvia, both with a rate of 1%.

```python
from soa_tables import read_soa_table_xml as rst
from lifeActuary import mortality_table as mtable, annuities as la, mortality_insurance as
    lins

# reads soa table TV7377
soaTV7377 = rst.SoaTable('soa_tables/TV7377.xml')
soaGRF95 = rst.SoaTable('soa_tables/GRF95.xml')
#soa = rst.SoaTable('soa_tables/' + 'TV7377' + '.xml')

# creates a mortality table
#tv7377 = mt.MortalityTable(data_type='q', mt=soa.table_qx, perc=100, last_q=1)
grf95 = mtable.MortalityTable(mt=soaGRF95.table_qx)
tv7377 = mtable.MortalityTable(mt=soaTV7377.table_qx)

## Silvia
age = 66
premium = 200000
a4_66 = la.ax(mt=grf95, x=age, i=1, g=0, m=4)
a2_66 = la.ax(mt=grf95, x=age, i=1, g=0, m=2)
T = premium/(4 * a4_66 + 2 * a2_66)
print(T)

## Robert
age = 66
premium = 200000
termIa=grf95.w
Ia_66 = la.nIax(mt=tv7377, x=age, n=termIa, i=1, m=1, first_amount=1, increase_amount=1)
E466 = la.nEx(mt=tv7377, x=age, i=1, g=0, n=4)
ad66 = la.aax(mt=tv7377, x=age, i=1, g=0, m=1)
A66_10 = lins.nAx_(mt=tv7377, x=age, n=10, i=1, g=0)

T = (premium -50 * Ia_66 - 800 * E466)/(ad66 + E466 + A66_10)
print(T)
```

Solutions:

Silvia:

$$
\begin{aligned}
200\ 000 &= 4T\, a_{66}^{(4)} + 2T\, a_{66}^{(2)} \Leftrightarrow \\
\Leftrightarrow T &= \frac{200\ 000}{4\ a_{66}^{(4)} + 2\ a_{66}^{(2)}} = 1\ 480.91\ m.u.
\end{aligned}
$$

Robert:

$$
\begin{aligned}
200\ 000 &= T\, \ddot{a}_{66} + 200\,(Ia)_{66} + (T+800)\,{}_4E_{66} + T\bar{A}_{66:\overline{10}|} \Leftrightarrow \\
\Leftrightarrow T &= \frac{200\ 000 - 200\,(Ia)_{66} - 800\,{}_4E_{66}}{\ddot{a}_{66} + {}_4E_{66} + \bar{A}_{66:\overline{10}|}} = 6\ 878.47\ m.u.
\end{aligned}
$$

Example 6

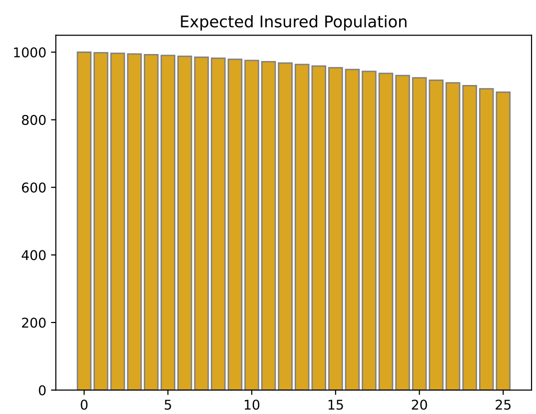
Consider that an Endowment Insurance with claim capital of 100 000 m.u. is sold to a group of 1000 insureds aged $x = 40$. The capital in case of life is paid at age $r = 65$ and premiums are paid during p =15 years. Using the TV7377 mortality table with an interest rate of 2%, determine:

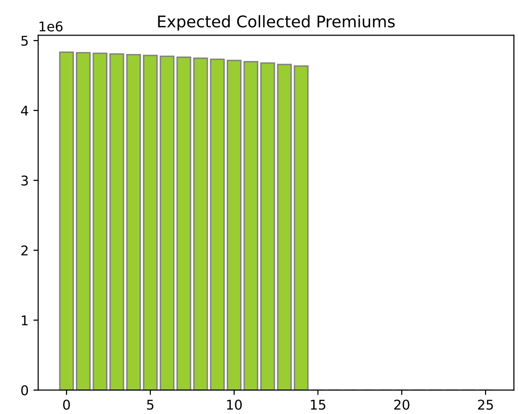
1. The risk premium of the contract.
2. The level risk premium.
3. For each year $t = 0, \ldots, r - x$, obtain:
   (a) The evolution of the expected number of insureds still alive at the beginning of year t.
   (b) The expected amount of collected premiums, in the beginning of year t.
   (c) The expected amount of claims paid at the end of year t.
   (d) The evolution of the fund of the contract.
4. Export the estimated amounts to an Excel file.
5. Save the estimated amounts in a dataframe.
6. Represent the amounts obtained in questions 3a, 3b, 3c and 3d in separated barplots.
7. Represent the amounts obtained in questions 3b, 3c and 3d in a single barplot.
8. Consider $x = 60$, $p = 5$ and $r = 80$ and compare.

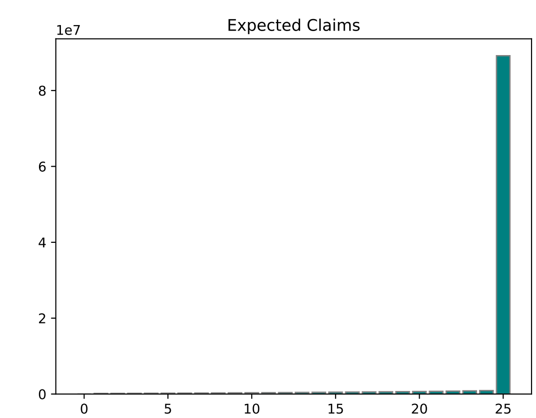
```python
from soa_tables import read_soa_table_xml as rst
from lifeActuary import mortality_table as mtable, annuities as la, \
    mortality_insurance as lins

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Reading the Mortality Table
mt = rst.SoaTable('soa_tables/TV7377' + '.xml')
lt = mtable.MortalityTable(mt=mt.table_qx)

# Data
x = 40
r = 65
n_premiums = 15
i = 2
g=0
capital = 100000
l0 = 1000

# QUESTION 1
risk_premium_1 = lins.nAEx(mt=lt, x=x, n=r - x, i=i, g=g)
risk_premium = risk_premium_1 * capital
print(f'risk_premium: {risk_premium}')

# QUESTION 2
aax_r = la.naax(mt=lt, x=x, n=n_premiums, i=i, g=g, m=1)
premium_leveled = risk_premium / aax_r
print(f'premium_leveled: {premium_leveled: ,.2f}')

# QUESTION 3
# lx, Premiums, Claims and Actuarial Liabilities
for idx_ages, ages in enumerate(range(x, r + 1)):
    path_liabilities['t'].append(idx_ages)
    path_liabilities['age'].append(ages)
    npx = lt.npx(x=x, n=idx_ages)
    nqx = lt.nqx(x=x + idx_ages, n=1)
    path_liabilities['px'].append(npx)
    path_liabilities['qx'].append(nqx)
    if idx_ages <= n_premiums - 1:
        path_liabilities['premium'].append(premium_leveled)
        path_liabilities['al'].append(lins.nAEx(mt=lt, x=x + idx_ages, n=r - x - idx_ages, i=i, g=g) * capital -
                                      premium_leveled *
                                      la.naax(mt=lt, x=x + idx_ages, n=n_premiums - idx_ages, i=i, g=g, m=1))
    else:
        path_liabilities['premium'].append(0)
        path_liabilities['al'].append(lins.nAEx(mt=lt, x=x + idx_ages, n=r - x - idx_ages, i=i, g=g) * capital)

    # EVOLUTION OF THE FUND
    path_fund['t'].append(idx_ages)
    path_fund['lx'].append(l0 * npx)
    path_fund['in'].append(l0 * npx * path_liabilities['premium'][idx_ages])
    if idx_ages == 0:
```

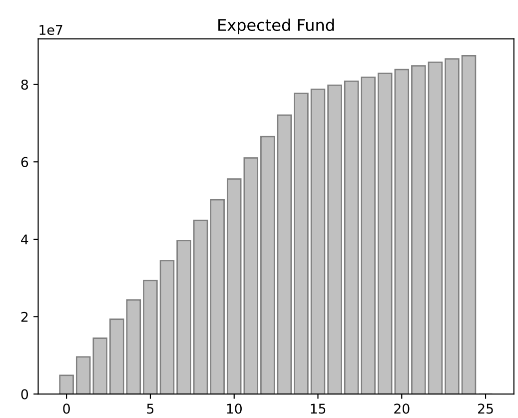
```
        path_fund['fund'].append(l0 * path_liabilities['premium'][idx_ages])
        path_fund['out'].append(0)
    else:
        if ages < r:
            path_fund['out'].append(l0 * path_liabilities['px'][idx_ages-1] *
                                    path_liabilities['qx'][idx_ages-1] * capital)
        else:
            path_fund['out'].append(l0 * path_liabilities['px'][idx_ages - 1] *
                                    path_liabilities['qx'][idx_ages - 1] * capital+
                                    l0*path_liabilities['px'][idx_ages]*capital)
        path_fund['fund'].append(path_fund['fund'][-1] * (1 + i / 100) + path_fund['in'][-1] -
    path_fund['out'][-1])

# QUESTION 4
(pd.DataFrame(path_liabilities)).to_excel(excel_writer='liabilities' + '.xlsx', sheet_name='
    liabilities',
                                          index=False, freeze_panes=(1, 1))

(pd.DataFrame(path_fund)).to_excel(excel_writer='fund' + '.xlsx', sheet_name='fund',
                                   index=False, freeze_panes=(1, 1))

# QUESTION 5
df_liabilities = pd.DataFrame(path_liabilities)
df_fund = pd.DataFrame(path_fund)

# QUESTION 6
# lx
fig, axes = plt.subplots()
barWidth = .1
br1 = np.arange(len(df_fund['lx']))
plt.bar(br1, df_fund['lx'], edgecolor='gray', color='goldenrod')
plt.title('Expected Insured Population')
plt.savefig('example6lx' + '.eps', format='eps', dpi=3600)
plt.show()

# Premiums
fig, axes = plt.subplots()
barWidth = .1
br1 = np.arange(len(df_fund['in']))
plt.bar(br1, df_fund['in'], edgecolor='gray', color='yellowgreen')
plt.title('Expected Collected Premiums')
plt.savefig('example6premiums' + '.eps', format='eps', dpi=3600)
plt.show()

# Claims
fig, axes = plt.subplots()
barWidth = .1
br1 = np.arange(len(df_fund['out']))
plt.bar(br1, df_fund['out'], edgecolor='gray', color='teal')
plt.title('Expected Claims')
plt.savefig('example6claims' + '.eps', format='eps', dpi=3600)
plt.show()

# Fund
fig, axes = plt.subplots()
barWidth = .1
br1 = np.arange(len(df_fund['fund']))
```

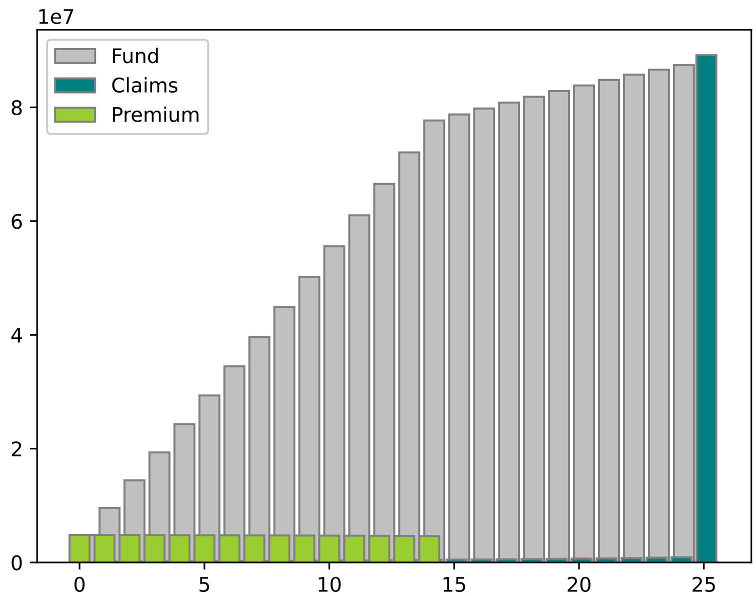
```python
plt.bar(br1, df_fund['fund'], edgecolor='gray', color='silver')
plt.title('Expected Fund')
plt.savefig('example6fund' + '.eps', format='eps', dpi=3600)
plt.show()

# QUESTION 7
fig, axes = plt.subplots()
barWidth = .1
br1 = np.arange(len(df_fund['in']))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]
plt.bar(br3, df_fund['fund'], label='Fund', edgecolor='gray', color='silver')
plt.bar(br2, df_fund['out'], label='Claims', edgecolor='gray', color='teal') #lightblue
plt.bar(br1, df_fund['in'], label='Premium', edgecolor='gray', color='yellowgreen')
plt.legend()
plt.savefig('example6all' + '.eps', format='eps', dpi=3600)
plt.show()
```



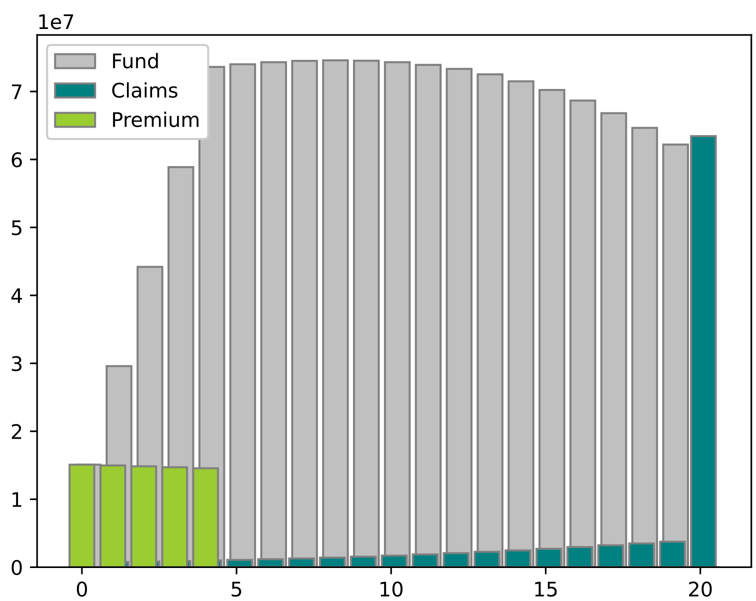








The result for the data of Question 8 is given by



9.5 Multiple Lifes Annuities and Insurance

Example 7

Marc and John, two business partners, both aged 50 years old, bought a life insurance with the following conditions:

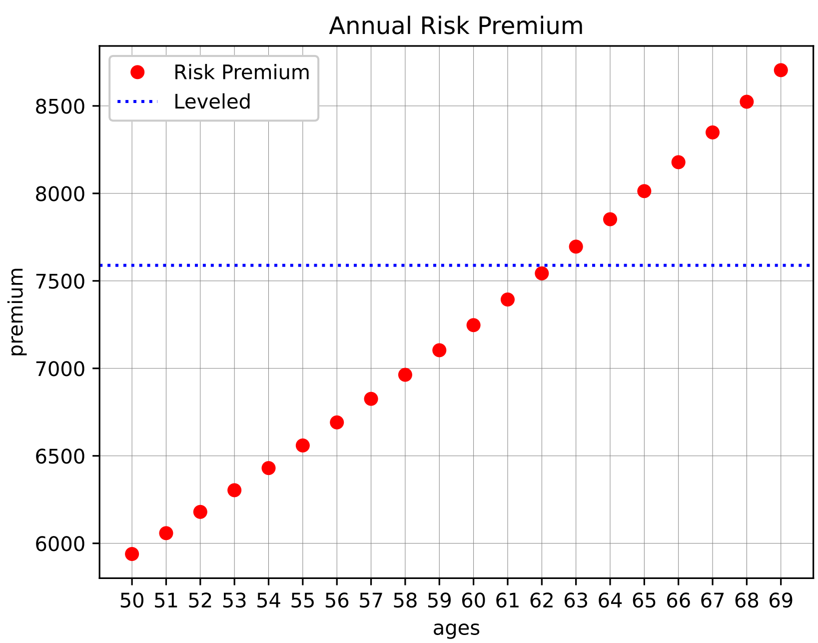
- Payment of 500 m.u., paid monthly, over the next 20 years, as long as one of them is alive. The terms of the life annuity increase 2% each year.
- Upon the death of the second, if the event occurs in the following 20 years, payment of 1.000.000 m.u.

Considering the TV7377 with a rate of 2% as the actuarial table:

1. Obtain the level premium of the contract, considering that premiums are paid semi-annually, during 20 years, as long as both of them is alive.
2. Considering that each year, the group pays the risk premium for the following year, obtain the risk premium of the contract, for each of the following 20 years and represent them graphically. Include the leveled premium in the same plot.

```python
# reads soa table TV7377
soaTV7377 = rst.SoaTable('soa_tables/TV7377.xml')
tv7377 = mtable.MortalityTable(mt=soaTV7377.table_qx)

x=50
y=50
n=20
i=2

# QUESTION 1
a12xy = l2h.naxy(mtx=tv7377, mty=tv7377, x=x, y=y, n=n, i=i, g=2, m=12,status='last-survivor')

Axy = l2h.nAxy_(mtx=tv7377, mty=tv7377, x=x, y=y, n=n, i=i, g=0, status='last-survivor')

aa2xy = l2h.naaxy(mtx=tv7377, mty=tv7377, x=x, y=y, n=n, i=i, g=0, m=2, status='joint-life')

premium_leveled = (500 * 12 * a12xy + 1000000 * Axy) / (2 * aa2xy)

print(f'a12xy:{a12xy}')
print(f'Axy:{Axy}')
print(f'aa2xy:{aa2xy}')
print(f'Leveled Premium:{premium_leveled}')

# QUESTION 2
# Risk Premium
g=2
premium_annual = [l2h.naxy(mtx=tv7377, mty=tv7377, x=x + a, y=y + a, n=1, i=i, g=0, m=12,
    status='last-survivor', method='udd') * 500 * 12 * (1+g/100) ** a +
    l2h.nAxy_(mtx=tv7377, mty=tv7377, x=x + a, y=y + a, n=1, i=i, g=0, status='last-survivor',
     method='udd') * 1000000
for a in range(n)]

```

```python
# Plot
fig, axes = plt.subplots()
ages = range(x, x + n)
plt.xticks(ages)
plt.plot(ages, premium_annual, 'ro', label='Risk Premium')

plt.xlabel('ages')
plt.ylabel('premium')
plt.title('Annual Risk Premium')
plt.grid(visible=True, which='both', axis='both', color='grey', linestyle='-', linewidth=.1)
plt.legend()
plt.savefig(this_py + 'arp' + '.eps', format='eps', dpi=3600)
plt.show()
```



Solutions:

Question 1:

$$P\,\ddot{a}^{(2)}_{50:50:\overline{20}|} = 500 \times 12\, a^{(12)}_{50:50:\overline{20}|} + 1.000.000\, \bar{A}_{50:50:\overline{20}|}$$

$$P = 3794.41\, m.u.$$

Question 2:

$$P_k = 500 \times 12\, a^{(12)}_{\overline{50+k:50+k:1}|} + 1.000.000\, \bar{A}_{\overline{50+k:50+k:1}|} \quad , \quad k = 0, 1, \ldots, 19$$

or

$$P_k = 500 \times 12\, a^{(12)}_{\overline{50+k:50+k:1}|} + 1.000.000 \times (1+i)^{-0.5}\, q_{\overline{50+k:50+k}} \quad , \quad k = 0, 1, \ldots, 19$$