



Actuarial & Life Insurance Component User Manual



| | | |
|---------|--|----|
| 1. | Introduction..... | 1 |
| 1.1. | Price of the insurance or present value of insurance | 1 |
| 1.2. | Mortality Tables | 1 |
| 1.3. | Interest | 2 |
| 1.4. | Costs..... | 3 |
| 1.5. | Premiums | 3 |
| 1.6. | Choice of mortality figures and interest rate in life insurance | 3 |
| 1.7. | Difference between assumed ciphers and reality | 3 |
| 1.8. | Expected deviations in mortality | 3 |
| 1.9. | Shifting the age of the person | 4 |
| 1.10. | Pure Endowment Insurance | 5 |
| 1.10.1. | Calculation Example | 5 |
| 1.10.2. | Split the duration of pure endowment insurance | 5 |
| 2. | Life Annuity | 7 |
| 2.1. | The Definition of Life Annuity | 7 |
| 2.2. | Present value of Life Annuity..... | 7 |
| 2.2.1. | Present value of whole life annuity. | 8 |
| 2.2.2. | Present value of the deferred whole life annuity. | 9 |
| 2.2.3. | Present value of the temporary life annuity | 10 |
| 2.2.4. | Present value of the deferred temporary life annuity..... | 11 |
| 2.3. | Safety margin..... | 11 |
| 2.4. | Not annual term of payments..... | 12 |
| 2.4.1. | Periodical payments | 12 |
| 2.4.2. | Continuous payments..... | 13 |
| 3. | Death Annuity or family benefit income | 14 |
| 3.1. | Present Value of Death Annuity | 14 |
| 4. | Death Insurance..... | 16 |
| 4.1. | Present value of 1-year death insurance. | 16 |
| 4.2. | Present value of whole life death insurance | 17 |
| 4.3. | Present value of temporary death insurance..... | 18 |
| 4.4. | Present value of deferred death insurance | 19 |
| 4.4.1. | Present value of deferred whole life death insurance..... | 19 |
| 4.4.2. | Present value of the deferred temporary death insurance. | 20 |
| 4.5. | Benefit payout moment..... | 21 |
| 5. | Present value of Combined Insurances | 22 |
| 5.1. | Endowment Insurance | 22 |
| 5.2. | Combinations of whole life death insurance | 23 |
| 5.2.1. | Whole life death insurance with pure endowment insurance | 23 |
| 5.2.2. | Whole life death insurance with a temporary higher assurance | 24 |
| 5.3. | Death Annuity in combination with Death Insurance..... | 24 |
| 5.4. | Pure Endowment Insurance combined with Present Value Refund after Death | 24 |
| 5.5. | Fixed Term Death Insurance | 25 |
| 5.6. | Fixed Term Endowment Insurance | 26 |
| 5.7. | Other Fixed Term Death Insurances | 26 |
| 6. | Increasing and Decreasing Annuities | 27 |
| 6.1. | Increasing Life Annuity with a simple rate..... | 27 |
| 6.1.1. | Whole Life Increasing Annuity with a simple rate..... | 27 |



| | | |
|--------|--|----|
| 6.1.2. | Whole Life Temporary Increasing Annuity..... | 28 |
| 6.1.3. | Temporary Life Annuity, increasing with a simple rate..... | 29 |
| 6.2. | Increasing Life Annuity with a compound rate | 29 |
| 6.3. | Decreasing Life Annuity..... | 30 |
| 6.4. | Not regularly increasing or decreasing annuity..... | 31 |
| 7. | Increasing and Decreasing Death Insurance | 33 |
| 7.1. | Regularly Increasing and Decreasing Death Insurance | 33 |
| 7.2. | Not regularly Decreasing Death Insurance | 33 |
| 8. | Premiums..... | 34 |
| 8.1. | Background..... | 34 |
| 8.2. | Gross value of insurance and gross premiums | 36 |
| 9. | Appendices..... | 37 |
| 9.1. | Mortality tables Xml Format..... | 37 |
| 9.1.1. | mortalitytables element..... | 37 |
| 9.1.2. | countries element | 38 |
| 9.1.3. | country element | 38 |
| 9.1.4. | mortalitytable element | 39 |
| 9.1.5. | mortalityrecord element | 40 |
| 9.2. | Class diagram..... | 41 |
| 9.3. | Common Dutch Translations | 42 |



1. Introduction

Life insurance calculation of single-person insurance using interest rate, death and life probabilities is a primary goal of this module. This chapter is meant as an introduction, where you learn basic terms used in the price calculation of insurance and premiums.

To make the first chapter very simple, only one type of insurance is discussed - pure endowment insurance. In Chapters 2 to 7, we introduce other basic and mixed types of insurance during the life and upon the death of the insured person. Lastly, Chapter 8 shows how to calculate premiums to be paid for an insurance policy.

After reading this manual, you will be able to calculate the premiums and price of any type of life insurance.

It is assumed that anyone reading this manual has a minimal knowledge of C# programming and a basic understanding of interest rate calculations (such as present or future values).

1.1. Price of the insurance or present value of insurance

Insurance policies like any other products on the market have their price. The insurer (the life insurance company) calculates the price of an insurance policy with the intention of funding claims to be paid and administrative costs, as well as making a profit.

Price of insurance is the value of the product at the present time, so it is often referred to as 'present value'.

The price of the insurance is determined using mortality tables, interest rates and costs. Because these are the major factors in calculations, we will explain them in the following sections in more detail.

1.2. Mortality Tables

Mortality tables are statistic-based tables showing expected annual mortality rates. It is possible to derive life expectancy from these mortality assumptions. In practice, these mortality tables are used in conjunction with the health and family history of the individual applying for a policy in order to determine premiums and insurability.

Latest mortality figures are periodically populated by government institutions (and internally by Figlo BV).

We assume that probabilities derived from those tables are valid for the whole duration of the insurance. Examples in this manual use mortality tables from the Netherlands (e.g. GBM 1990-1995). For every example, the code tells you which mortality table is used.

Examples use helper method **GetMortalityTable()** written in C# and the .Net component **Figlo.Actuarial.Mortality.Data.Sql** to retrieve mortality tables from the SQL database.



```
private static MortalityTable GetMortalityTable(string countryCode, Gender gender, string tableName)
{
    using (SqlMortalityData sqlMortalityData = SqlMortalityData.ConnectWithConnectionSection("mortality"))
    {
        MortalityTable mortalityTable = sqlMortalityData.GetMortalityTable(countryCode, gender, tableName);
        return mortalityTable;
    }
}
```

With the Figlo.Actuarial.Mortality.Data component, it is possible to read the mortality from:

- SQL database
- Xml file
- Xml resource
- Xml stream
- Online Mortality Web Service

There are also methods for writing Xml tables into xml files, which are useful when copying remote mortality tables (SQL, Online Mortality Web Service) to a local xml file (or resource).

For additional information on a format of the xml file with mortality tables, see the appendix, section §9.1.

Mortality table object exposes the following methods:

| Method | Description |
|----------|--|
| l_x | Number surviving to the age x . |
| d_x | Number dying between age x and $(x+1)$. |
| p_x | Probability of surviving (1 person). |
| q_x | Probability of dying (1 person). |
| e_x | Life expectation (1 person). |
| p_{xy} | Probability of surviving in the given scenario of two persons. |

These methods are actively used by life insurance components to calculate premiums and the price of insurance.

1.3. Interest

Because there is a delay between the receipt of premiums and eventual payouts (payments to insured persons), received premiums are usually invested by insurance companies and the interest rate is used to estimate the return on investment. By using interest rates, the present value of all payments is always less than future payments. The interest rate used in this manual, is a 'perunage' rate, which is 100 times less than 'percentage'. For example, 0.04 interest perunage is equal to 4% interest percentage.



1.4. Costs

Life Insurance Companies have costs such as administration, certifications, billing premiums and payouts, provisions to intermediate persons and so on. Such costs are usually added up into premiums. Premiums where no costs are taken into account are called net premiums. Premiums where costs are included are called gross premiums.

When we talk about the price of insurance, we say 'net present value of insurance' if no costs are included and 'gross present value of insurance' if all costs are included.

1.5. Premiums

Knowing the price of insurance, the holder of an insurance policy can buy it paying a lump sum at the beginning of the insurance term or he can spread the payment over the specified period of time. Payments at regular intervals of time are called 'premiums'. Most commonly used intervals of payments are twelve months, six months, three months or one month. In Chapter §8, you can find detailed information on how to calculate life insurance premiums.

1.6. Choice of mortality figures and interest rate in life insurance

It is very important for insurance companies to choose proper basic figures (parameters) with which life insurance is calculated. They want enough of a safety margin against possible loss in the future.

On the other hand, premiums cannot be too high because they will be unaffordable.

Care should be taken. Once premiums are fixed, it is usually impossible to change them during the term of the insurance contract.

1.7. Difference between assumed figures and reality

Because mortality, interest rate and costs are based on historical figures, it is likely that those figures will be different in the future during the term of the insurance contract. Such difference can bring loss or benefit to an insurance company.

Indeed, the more people who die with insurance against death (excess mortality), the more payouts have to be made to insured people, i.e. losses are incurred.

For insurance policies that pay out during the lifetime of the insured person, on the other hand, excess mortality means profit.

If more than the expected number of people survive (lower mortality), profits and losses change round. The more people who survive to the end of the insurance term, the more payouts have to be made for policies during their lifetime and fewer payouts have to be made for insurance policies against death.

1.8. Expected deviations in mortality

In life insurance, mortality tables are usually used for the entire population, taking no account of healthy and less healthy people.



For insurance policies that everyone has to have by law, insurance companies are not permitted to differentiate according to a person's health. An example of this insurance policy is basic retirement income insurance (pension or AOW in the Netherlands).

For insurance policies where the insured person can choose whether he is insured or not, insurance companies have to take the health of that person into account. Anyone who is very ill wants to insure himself against death. Healthy people are less likely to insure themselves against death. Therefore, it is a common practice in non-mandatory insurances that applications have to demonstrate the health of the insurance candidate. Extremely unhealthy candidates are charged a higher premium or sometimes rejected altogether.

The tendency of the insurance candidate to get the most out of an insurance policy for himself is called anti-selection.

These reasons make it almost impossible to calculate accurate life insurance premiums using only mortality tables for the entire population (GBM or GBV). Anti-selection results in calculations being performed for people who are on average less healthy. Compensatory factors are calculated on the basis of health statements issued by candidates (worst case scenario is handled).

There is a third reason why mortality tables for the entire population cannot be used without corrections. Figures in these tables are based on historical events, which cannot be assumed to remain the same in the future. Recent research has shown that that mortality figures have always fluctuated considerably in the past, but that average lifetimes are getting longer and that there are more people with specific diseases, which has a major impact on survival/death probabilities.

1.9. Shifting the age of the person

Insurance companies take the above-mentioned influences and uncertainties into account as special margins in the mortality probabilities.

The safety margin is created if the probability of payout is made slightly higher in calculation.

For payouts during a person's lifetime, insurance companies assume a younger age in their calculations, so the probability of surviving is higher.

For the same reason, age is assumed older for insurances against death.

This shift in age can also be based on the health of the insured person.

When an insurance company uses age shifting, it is usually noticeable in the mortality figures. For example, GBM 1961-1965 – 2 means that a 2-year younger age is used in mortality table GBM 1961-1965.

Age shifting is not the only way to create a margin. Insurance companies can create a slight margin by choosing an interest rate at which the return on invested premium is probably slightly higher than expected.

In practice, if an insurance company makes a lot of profit from the built-in margin, it is often shared with insured people. Therefore, the burden of those margins for insured people is kept to a minimum.

1.10. Pure Endowment Insurance

We will finish this chapter with a practical example. For this example, we have chosen one of the simplest types of life insurance – Pure Endowment Insurance.

Capital Sum Assurance or Pure Endowment Assurance is a form of insurance where assured capital will be paid out at the end of the term of insurance on condition that the policyholder is still alive at the end of that term. It is sometimes called pure endowment insurance as opposed to normal endowment insurance (§5.1), where insured capital is also paid out after a person dies.

1.10.1. Sample Calculation

The present value of the insurance is calculated according to the example below.

Example:

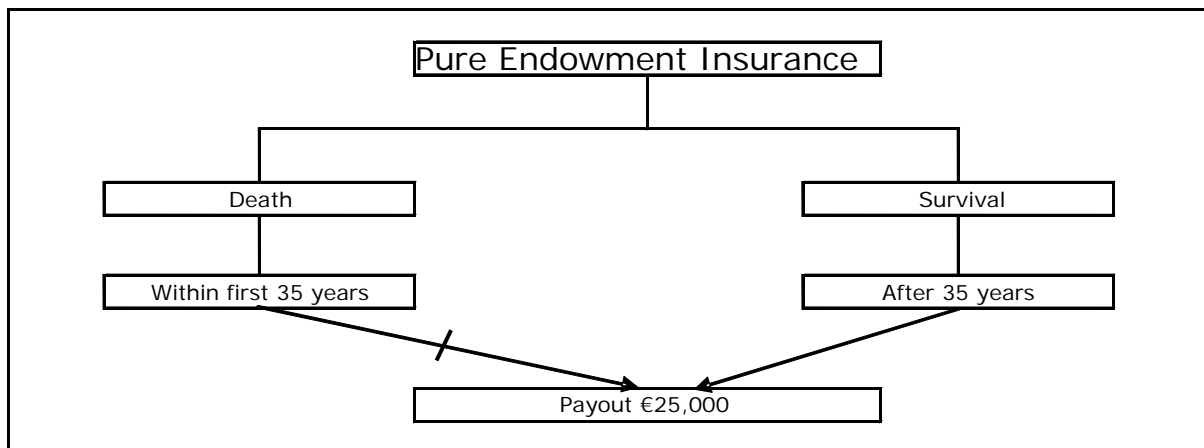
Age of the insured male person at the beginning of the insurance is 30.

Insurance term: 35 years ($n=35$).

Insured capital: €25,000

Mortality table: LX_9095

Interest rate: 0.04



Endowment function nEx (actuarial notation ${}_nE_x$) is used to calculate the net present value of the pure endowment insurance.

```

public void nEx_PureEndowment()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.nEx(lxTable, 0.04, 30, 25000, 0, 35);
    Console.WriteLine("Present value of the pure endowment insurance is {0:C}", pv);
    // Output: Present value of the pure endowment insurance is €5,254.42
}
  
```

1.10.2. Split the duration of pure endowment insurance

Example:



The insured capital of €25,000 in the pure endowment insurance with a duration of 30 years for a male person aged 40 has to be split into two insurances: the first runs for 20 years and the second for 10 years.

It can be shown that the present values of the complete insurance will be expressed through present values of the two insurances, the first one with a duration of 20 years and the second one is deferred insurance with a duration of 10 years and a delay of 20 years.

The formula for the present value of the complete insurance is:

$${}_{30}E_{40} = {}_{20}E_{40} * {}_{10}E_{60}$$

Or in generic form:

$${}_{n+m}E_x = {}_nE_x * {}_mE_{x+n}$$

Note that this formula is only valid for unit calculation, which is the calculation of the present value of insured capital with value 1. To calculate other values, simply multiply the result by the desired insurance capital:

$${}_{n+m}E_x = A * {}_nE_x * {}_mE_{x+n} \quad \text{Where A is insured capital.}$$

```
public void nEx_PureEndowment_Split()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double part1 = LifeInsuranceMath.nEx(lxTable, 0.04, 40, 1.00, 0, 20);
    double part2 = LifeInsuranceMath.nEx(lxTable, 0.04, 60, 1.00, 0, 10);
    double pv = 25000 * part1 * part2;
    Console.WriteLine("Present value of the pure endowment insurance is {0:C}", pv);
    // Output: Present value of the pure endowment insurance is €5,649.05

    // This should produce the same result:
    pv = LifeInsuranceMath.nEx(lxTable, 0.04, 40, 25000.00, 0, 30);
    Console.WriteLine("Present value of the pure endowment insurance is {0:C}", pv);
    // Output: Present value of the pure endowment insurance is €5,649.05
}
```

This form of splitting is used in practice if there is a need to use different mortality tables or different rates in the first and second parts of the insurance. In such cases, the result of the calculation as a product of two insurances will not be equal to the initial insurance with one complete term.



2. Life Annuity

In the previous chapter, we discussed the term 'present value' of the insurance. Only one type of insurance was discussed. In this chapter, you will learn how to calculate the present value of four other types of insurances.

2.1. The Definition of Life Annuity

Life annuities are mostly used to assure retirement income.

Life annuity is a contract that provides an income for life, payable annually or more frequently, on condition that the person receiving the payments is alive.

The simplest example of this kind of income is a pension. When a 40-year-old person wants to buy pension insurance from his 65th birthday until he dies, this is referred to as deferred whole life annuity insurance with a 25-year delay.

With whole life annuity, the insured person receives benefit until he dies. With temporary life annuity, the insured person receives benefit until he dies or until the annuity policy expires. The 'deferred' option in the annuity contract specifies the beginning (delay from now) of the annuity contract.

Depending on the start end dates of the annuity contract, there are four known types of life annuity:

- whole life annuity
- deferred whole life annuity
- temporary life annuity
- deferred temporary life annuity

Below, we show you how to calculate present values for all of these four types of life annuity.

2.2. Present value of Life Annuity

To calculate the present value of life annuity, we use the `ax()` method of the `LifeInsuranceMath` class. The name of this method comes from actuarial notation a_x , where 'a' means 'annuity' and 'x' means person of age x.

This is a definition of this method:

```
public static double ax(  
    MortalityTable lxTable,  
    double rate,  
    int x,  
    AnnuityDueDate due,  
    Annuity surviveAnnuity,  
    Annuity deathAnnuity  
)
```

Because this chapter only discusses life annuities, the last parameter should be zero. Death annuities are discussed in Chapter 3 of this manual.

The key parameter in this method is *surviveAnnuity*. It contains information on life annuity such as delay, duration and initial payment. Delay, which is greater than 0, is used for deferred annuity. Duration, which is greater than 0, is used for temporary annuity, otherwise it is a whole life annuity. Initial payment is used to specify the value of the payment. For simplicity's sake, we use constant annuity in our examples, which means that payments are the same every year. It is possible to define increasing or decreasing annuities, but this is discussed in other chapters (see §6).

Another important parameter of the *ax()* method is the *x* parameter, which indicates a person's current age. It is assumed that an annuity contract starts immediately for a person of a given age, except for deferred annuity where a delay is also specified.

2.2.1. Present value of whole life annuity.

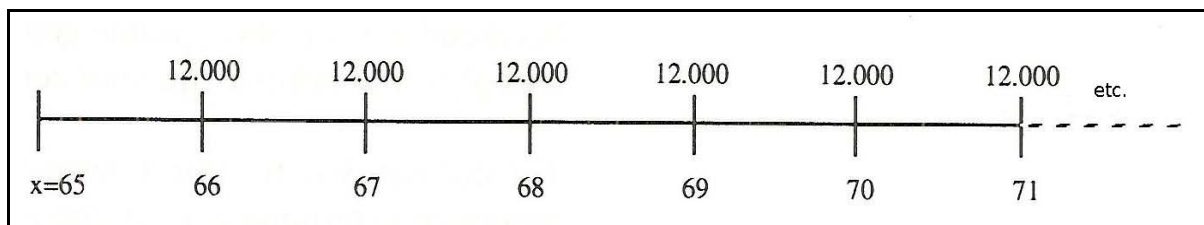
Whole life annuity is an annuity which begins immediately and lasts until the client dies.

Example:

A 65-year-old man signs a whole life annuity contract with a €12,000 benefit per year.

The first benefit is paid after 1 year, the second benefit is paid after 2 years and so on. The benefit is paid at the end of the each year (postnumerando).

Insurance ceases when the client dies. The last payment is made therefore before the client dies.



To create a constant whole life annuity we use the following statement:

```
new Annuity(12000)
```

This is equal to the annuity with delay 0 and duration 0:

```
new Annuity(12000, 0, 0).
```

The present value of this annuity is calculated with the following code:

```
public void ax_WholeLife()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 65, AnnuityDueDate.EndOfPeriod,
        new Annuity(12000), zero);
    Console.WriteLine("Present value of the whole life annuity is {0:C}", pv);
    // Output: Present value of the whole life annuity is €119,482.72
}
```

As mentioned above, in this example we calculate the present value of the postnumerando annuity. Parameter due date gives you the possibility to specify other payment moments, such as at the beginning of the year (praenumerando) or continuous payment (independent of the moment), e.g. for payment at the beginning of the year, calculation will be as follows:

```
public void ax_WholeLife_praenumerando()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 65, AnnuityDueDate.BegOfPeriod,
        new Annuity(12000), zero);
    Console.WriteLine("Present value of the praenumerando whole life annuity is {0:C}",
        pv);
    // Output: Present value of the whole life annuity is €131,482.72
}
```

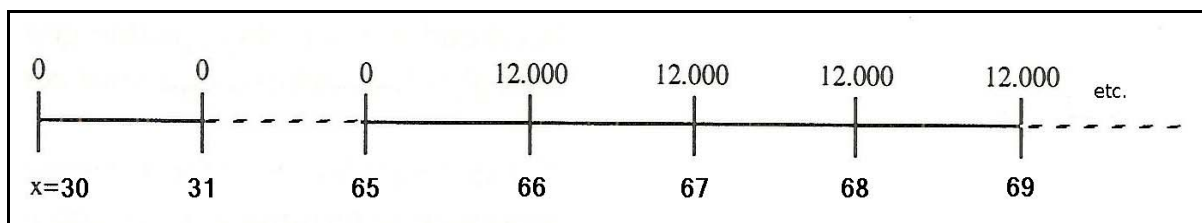
Note that in the last example, the present value is exactly €12,000 higher. That is because one more payment is added at the beginning of the first year.

2.2.2. Present value of the deferred whole life annuity.

If the benefit from life annuity does not start immediately, but after a specified number of years, we have to provide a delay parameter in the annuity object. The rest of the code looks very similar to the code for whole life annuity calculation (§2.2.1).

We modify the example in §2.2.1 as follows:

Age of the male person is 30 years, but annuity begins when client is 65-year-old:



Note that we create this annuity with an added delay parameter (35):

```
new Annuity(12000, 35)
```

The age parameter (x) of the person is 30 instead of 65. This means that the first benefits will be paid when the client is $31+35 = 66$ years old on condition that he is still alive (+1 is because payments are made at the end of the period, i.e. postnumerando).



```
public void ax_DeferedWholeLife()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 30, AnnuityDueDate.EndOfPeriod,
        new Annuity(12000, 35), zero);
    Console.WriteLine("Present value of the deferred whole life annuity is {0:C}", pv);
    // Output: Present value of the defered whole life annuity is €25,112.50
}
```

You probably also noted that the present value of the deferred life annuity is significantly lower. There are two reasons for this. The first is because the present value of delayed payments is lower than without delay. The second is that there is a probability of the client dying before he even reaches the date of the first benefit.

2.2.3. Present value of the temporary life annuity

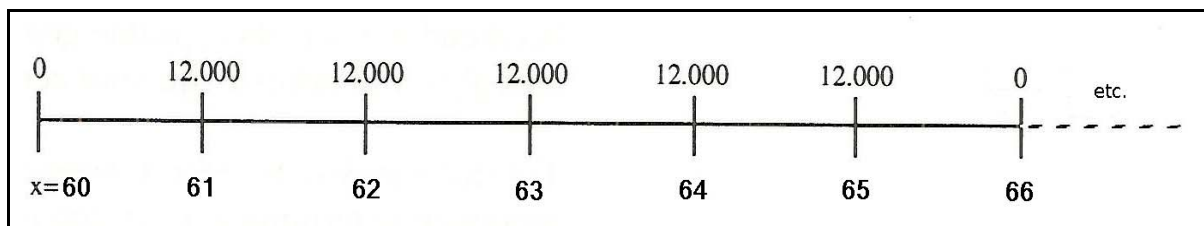
If the benefit from life annuity is defined for only a specified number of years, then it is temporary life annuity. It ends after a specified number of years or when the client dies, whichever is earlier.

To specify temporary life annuity, set *delay* parameter to 0 and *duration* parameter of the annuity object to a value other than 0.

Example:

A 60-year-old man wants to stop working at age 60. A normal pension starts at 65. Therefore he needs an income for between 60 and 65. The desired income is €12,000 a year.

To assure this benefit, the client has to sign a life annuity contract with a duration of 5 years.



To create this kind of temporary annuity, we use the following statement:

```
new Annuity(12000, 0, 5)
```

To calculate the present value of the annuity, we use the following code:

```

public void ax_Temporary()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

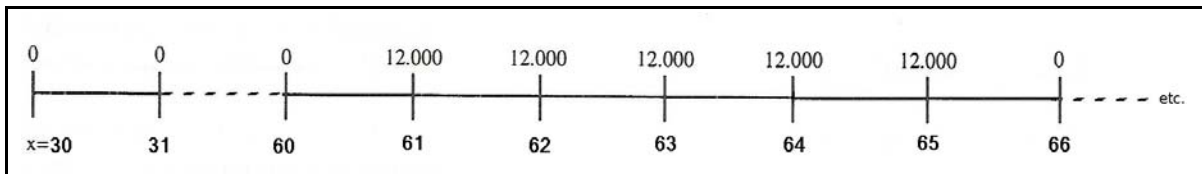
    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 60, AnnuityDueDate.EndOfPeriod,
        new Annuity(12000, 0, 5), zero);
    Console.WriteLine("Present value of the temporary life annuity is {0:C}", pv);
    // Output: Present value of the temporary life annuity is €51,239.71
}

```

2.2.4. Present value of the deferred temporary life annuity.

If the temporary life annuity starts after a specified number of years, then the calculation is very similar to the example in §2.2.3, except that the delay parameter is larger than 0.

We modify the previous example in such a way that the client is 30 years old and he wants to have an income of €12,000 a year during the early retirement period from 60 to 65 years old.



The annuity object is then created with the following statement:

```
new Annuity(12000, 30, 5)
```

And the present value of this annuity is calculated with the following code:

```

public void ax_DeferredTemporary()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 30, AnnuityDueDate.EndOfPeriod,
        new Annuity(12000, 30, 5), zero);
    Console.WriteLine("Present value of the deferred temporary life annuity is {0:C}",
        pv);
    // Output: Present value of the deferred temporary life annuity is €14,161.89
}

```

2.3. Safety margin

Lower mortality and anti-selection (see §1.7) produce a negative effect for life annuity, resulting in most cases as a loss to the insurer. To secure against such losses, the insurer introduces a safety margin. The most common safety margin is created by reducing the actual age of the insured person.

2.4. Non-annual term of payment

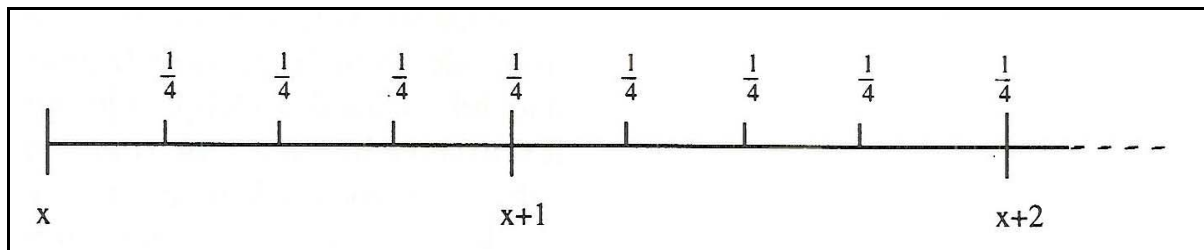
2.4.1. Periodical payments

In practice, benefit payments are not usually paid annually, but every six months, quarter or month. The most common situation is payment at the end of each month or quarter, if the policyholder is still alive.

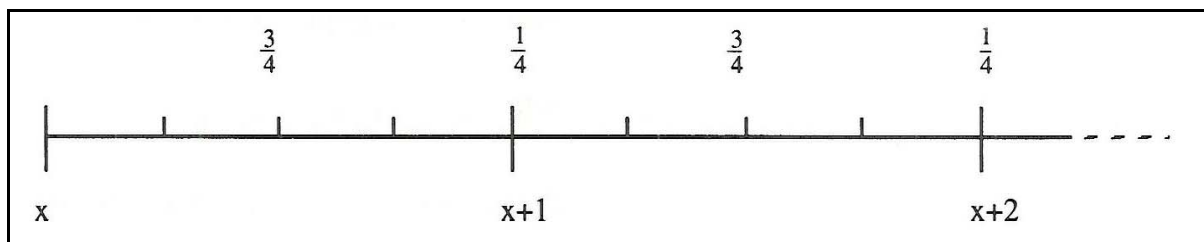
In actuarial notations, annuity with terms of payments every quarter is written as $a_x^{(4)}$, where (4) in brackets means 4 payments a year.

As an example, we will show you how to calculate postnumerando $a_x^{(4)}$, using a combination of annual postnumerando and praenumerando payments.

Instead of a single annual benefit, we have $\frac{1}{4}$ of the annual benefits, which is paid at the end of each quarter.



One of the $\frac{1}{4}$ annual payments takes place at the end of the year. For simplicity's sake, the other three benefits are merged in the middle of the year (they are symmetrically allocated around the middle of the year).



The present value of the annuity in the middle of the year is estimated as an average of postnumerando and praenumerando benefits, such as $\frac{3}{4} \times \frac{a_x + \ddot{a}_x}{2}$, where \ddot{a}_x means praenumerando annuity.

Together with the benefit at the end of the year, the total present value of the annuity will be:

$$\frac{3}{4} \times \frac{a_x + \ddot{a}_x}{2} + \frac{1}{4} a_x$$



Or, after simplifying:

$$\frac{5}{8} \times a_x + \frac{3}{8} \times \ddot{a}_x$$

Parameter *due* of the `LifeInsuranceMath.ax()` method allows to specify payment moments, such as praenumerando (beginning of the year) or postnumerando (end of the year).

2.4.2. Continuous payments

Sometimes, payments are defined for very small intervals, e.g. daily. In this case, we talk about continuous payments. The present value of such payments is estimated as an average of postnumerando and praenumerando annual payments (actuarial notation is \bar{a}_x).

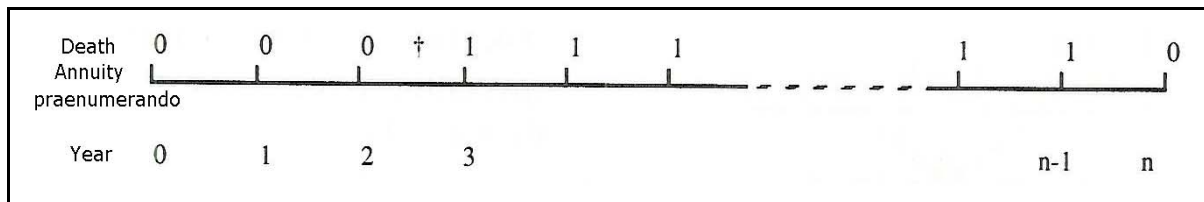
Using the *'due'* parameter in the `ax()` method set to `AnnuityDueDate.Continuous` will automatically calculate the present value of the annuity based on continuous payments.

3. Death Annuity or family benefit income

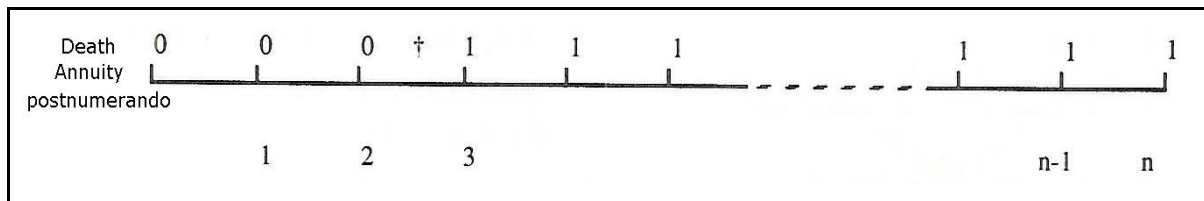
Death annuity is a series of payments which starts immediately after a person's death and stops at the end of the insurance term.

For example, given a death annuity insurance policy with a duration of 20 years, if the insured person dies 5 years after the beginning of the policy, the insurer is obligated to pay the assured amount for the remaining 15 years.

Annuity benefit can be paid at the end of the year (postnumerando) or at the beginning of the year (praenumerando), and can be also split into smaller intervals than one year.



1st benefit is after 3^d year, last benefit is after (n-1) years.



1st benefit is after 3rd year, last benefit is after n years.

When comparing praenumerando and postnumerando death annuities with the same duration for the same person, you can see that the number of benefit payments is different. In both cases, the first payment moment is at the beginning of the payment period after the death occurs. The last payment is one year before the end of the insurance term for praenumerando payments and exactly at the end of the insurance term for postnumerando payments.

Therefore, the present value of the praenumerando death annuity will be equal to a postnumerando death annuity with a 1-year shorter duration.

3.1. Present Value of Death Annuity

In the previous chapters, we have showed that the present value of life annuity can be calculated with the `LifeInsuranceMath.ax()` method. In this chapter, we will show that the present value of death annuity can be calculated using the same `LifeInsuranceMath.ax()` method.

The `LifeInsuranceMath.ax()` method has parameters `surviveAnnuity` and `deathAnnuity`. Because this chapter only discusses death annuity, we use only the `deathAnnuity` parameter and set the other one to zero.



The annuity object provided as a *deathAnnuity* parameter is created in the same way as it was created for Life Annuities (see §2.2).

Example:

A 30-year-old man has an annual income of €12,000. He wants to assure his family of the same income if he dies before he is 65.

An *Annuity* object with a duration of 35 years at €12,000 benefit a year is created with the following statement:

```
new Annuity(12000, 0, 35)
```

The present value of this annuity is calculated with the following code:

```
public void ax_DeathAnnuity()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 30, AnnuityDueDate.EndOfPeriod, zero,
        new Annuity(12000, 0, 35));
    Console.WriteLine("Present value of the death annuity is {0:C}", pv);
    // Output: Present value of the death annuity is €7,246.42
}
```

4. Death Insurance

Calculation of the present value of death insurance is based on the mortality probability and interest rate in the same manner as the present value of life and death annuity.

To calculate the present value of death insurance, we use the `Ax()` method of `LifeInsuranceMath` class. The name comes from actuarial notation A_x , where A means the present value of the risky capital and x means the age of the person.

Death insurance which is paid out only after the client dies is also frequently called 'pure term life assurance'. For simplicity's sake, we use the term 'death insurance' in this chapter.

The definition of the `Ax` method is as follows:

```
public static double Ax(
    MortalityTable lxTable,
    double rate,
    int x,
    PayoutDueDate payoutDue,
    Insurance deathPayout,
    double survivePayout
)
```

In this chapter, we discuss death insurance which is paid out only after the death of the person (pure term life assurance), so the `survivePayout` parameter is set to 0 in examples in this chapter.

The key parameter in the `Ax()` method is `deathPayout`, containing the duration, delay and value of the risky capital. When the delay is longer than 0, then it is deferred death insurance, i.e. death insurance with delayed risk. When the duration is longer than 0, it is temporary death insurance. Otherwise, it is whole life death insurance.

In this manual, we will show you examples of how to calculate the present values of:

- One-year death insurance
- Whole life death insurance
- Temporary death insurance
- Deferred death insurance

Examples:

After the death, the cost of the funeral has to be paid, which is usually quite an expensive matter. This risk is covered by funeral assurance in the form of whole life death insurance.

A family comprises a mother, father and two children. The father is an income earner and wants to take care of study costs for his children after his death, so he signs a temporary death insurance contract.

4.1. Present value of one-year death insurance.

In one-year death insurance, assured capital is paid out if the client dies within one year. One-year death insurance is created by setting the duration parameter of the insurance object to 1.

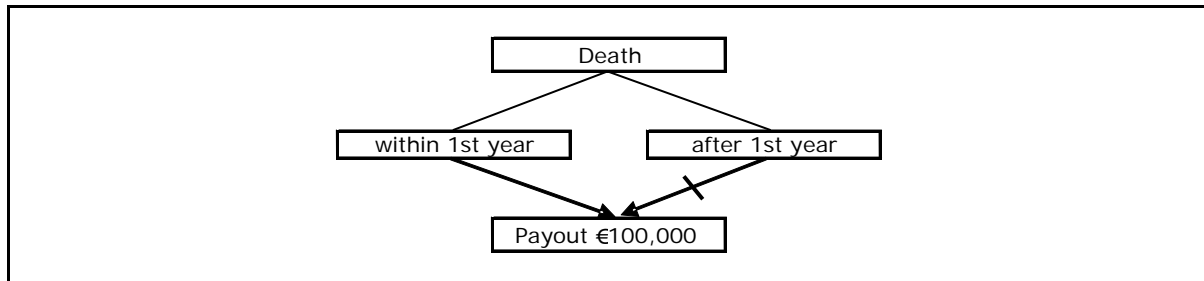
You can compare this type of insurance to temporary death insurance, which is technically the same; except the duration of the insurance is fixed at one year.

Example:

One-year death insurance with assured capital €100,000

Client age: 30

Mortality table: LX_9095 (the Netherlands, years 1990-1995)



The insurance object for this example is created with the following statement:

```
new Insurance(100000, 0, 1)
```

The present value of this kind of insurance is calculated with the following code:

```
public void Ax_1YearDeathInsurance()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.Ax(lxTable, 0.04, 30, PayoutDueDate.DirectOnDeath,
        new Insurance(100000, 0, 1), 0);
    Console.WriteLine("Present value of the 1-year death insurance is {0:C}", pv);
    // Output: Present value of the 1-year death insurance is €82.66
}
```

4.2. Present value of whole life death insurance

Whole life death insurance is a type of insurance in which a single benefit is paid out to a client after his death, no matter when this death occurs.

It is certain that the benefit will be paid one day, but it is not known when.

Example:

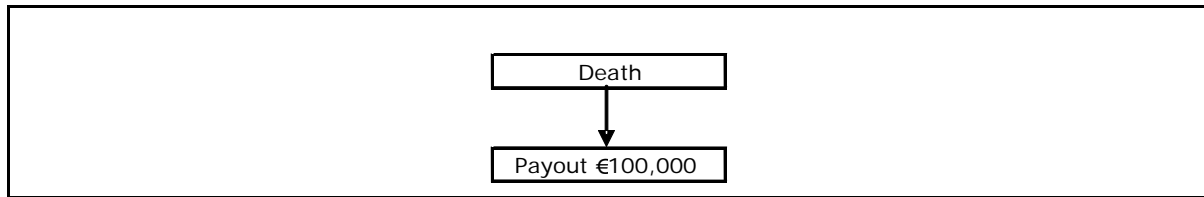
Calculate the present value of the whole life death insurance for the following input:

Age of the person: 30

Gender: Male

Assured capital: €100,000

Mortality Table: LX_9095 (the Netherlands, years 1990-1995).



The insurance object is created with the following statement:

```
new Insurance(100000)
```

Or equivalent:

```
new Insurance(100000, 0, 0)
```

The present value of this kind of insurance is calculated with the following code:

```
public void Ax_WholeLifeDeathInsurance()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.Ax(lxTable, 0.04, 30, PayoutDueDate.DirectOnDeath,
        new Insurance(100000), 0);
    Console.WriteLine("Present value of the whole life death insurance is {0:C}", pv);
    // Output: Present value of the whole life death insurance is €19,009.72
}
```

4.3. Present value of temporary death insurance

Temporary death insurance is a type of insurance where a single benefit is paid out to a client after his death, if the death occurs within a specified period of time.

Setting the 'duration' parameter of the insurance object to a value larger than 0 changes the insurance type from whole life to temporary.

You can compare this type of insurance to a one-year death insurance, which is technically the same except the duration of the insurance is not fixed at one year.

Example:

As an example, we will show you how to calculate the present value of temporary death insurance for a man aged 46. The benefit of €100,000 should be paid out if the client dies within the first four years:

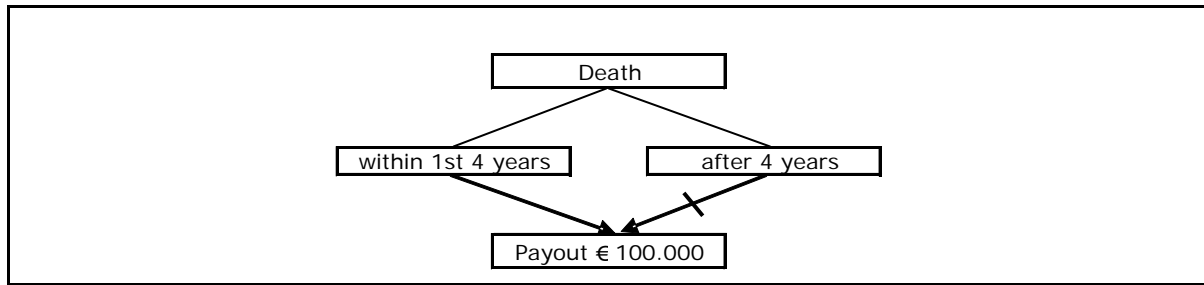
Age of the client: 46

Insurance duration: 4 years

Gender: Male

Assured capital: €100,000

Mortality Table: LX_9095 (the Netherlands, years 1990-1995).



The insurance object is created with the following statement:

```
new Insurance(100000, 0, 4)
```

The present value of this insurance is calculated with the following code:

```
public void Ax_TemporaryDeathInsurance()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.Ax(lxTable, 0.04, 30, PayoutDueDate.DirectOnDeath,
        new Insurance(100000, 0, 4), 0);
    Console.WriteLine("Present value of the temporary death insurance is {0:C}", pv);
    // Output: Present value of the temporary death insurance is €339.40
}
```

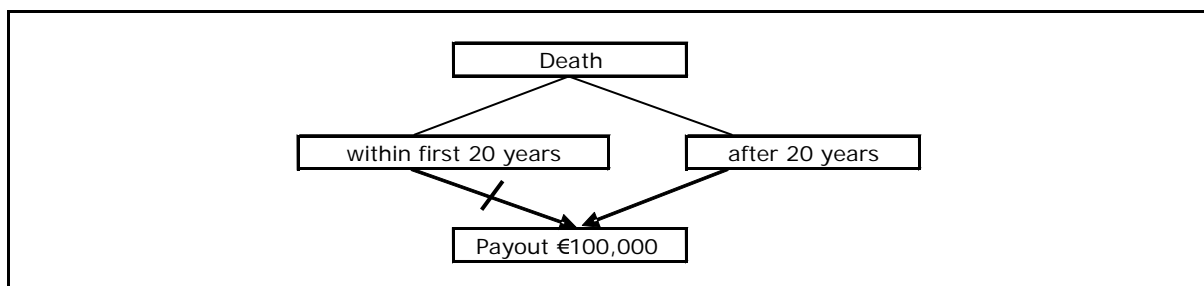
4.4. Present value of deferred death insurance

It is possible to calculate the present value of death insurance where the benefit is defined just after a certain number of years. This number of years defines the delay for the insurance. Only after this number of years has elapsed does the insurance become valid. Insurance with a delayed start date is called deferred insurance.

4.4.1. Present value of deferred whole life death insurance.

Example:

A 30-year-old male client wants to take out whole life death insurance, but only if he dies after 20 years. The delay of the insurance is then 20 years.



The insurance object is created with the following statement:



```
new Insurance(100000, 20, 0)
```

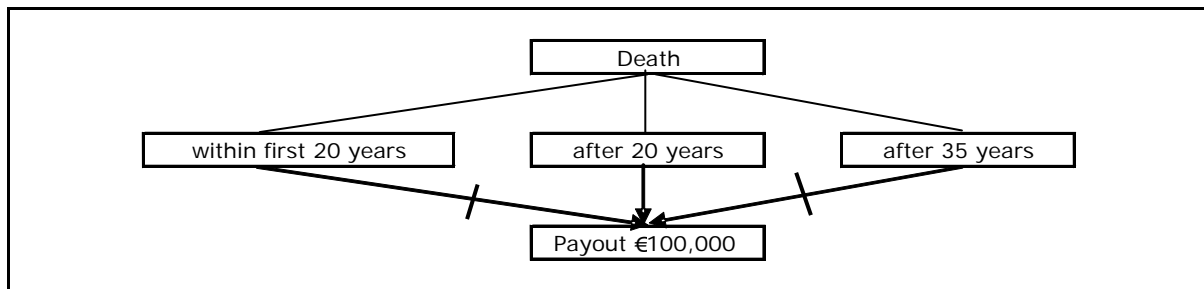
The present value of this insurance is calculated with the following code:

```
public void Ax_DeferredWholeLifeDeathInsurance()  
{  
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");  
  
    double pv = LifeInsuranceMath.Ax(lxTable, 0.04, 30, PayoutDueDate.DirectOnDeath,  
        new Insurance(100000, 20, 0), 0);  
    Console.WriteLine("Present value of the deferred whole life death insurance is {0:C}",  
        pv);  
    // Output: Present value of the deferred whole life death insurance is €16,702.95  
}
```

4.4.2. Present value of the deferred temporary death insurance.

Example:

A 30-year-old male client wants to take out death insurance, but only if he dies when he is between 50 and 65 years old. The delay of the insurance is then 20 years and the duration 15 years.



```
new Insurance(100000, 20, 15)
```

The present value of this insurance is calculated with the following code:

```
public void Ax_DeferredTemporaryDeathInsurance()  
{  
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");  
  
    double pv = LifeInsuranceMath.Ax(lxTable, 0.04, 30, PayoutDueDate.DirectOnDeath,  
        new Insurance(100000, 20, 15), 0);  
    Console.WriteLine("Present value of the deferred temporary death insurance is {0:C}",  
        pv);  
    // Output: Present value of the deferred temporary death insurance is €4,301.69  
}
```



4.5. Benefit payout moment

In the previous examples, we assumed that a single benefit is paid out immediately after a person dies. This is the most common case. However, other payment moments are sometimes used.

The payout moment is specified with the *payoutDue* parameter of the `LifeInsuranceMath.Ax()` method. In this section, we briefly describe possible values of this parameter.

| | |
|-----------------------------|---|
| <code>DirectOnDeath</code> | Single benefit is paid out immediately after a person's death. |
| <code>EndOfDeathYear</code> | Single benefit is paid out at the end of the year in which death occurred. |
| <code>EndOfInsurance</code> | Single benefit is paid out at the end of the insurance term if the person dies during the insurance term. This type of payout forms another so-called 'Fixed Term Death Insurance'. It will be discussed later in this manual (§5.7). |

5. Present value of Combined Insurances

In the previous chapters, we showed you how to calculate the present values of basic life and death insurances. In this chapter, we will show you how to calculate mixed-type insurance policies containing both life and death insurances in one.

In theory, mixed-type insurances are always a set of separate basic life or death insurances. However, due to the fact that some types of these insurances are very frequently used in practice, this chapter will show you how to calculate them without much effort.

5.1. Endowment Insurance

The official definition of endowment insurance is:

‘Under an endowment assurance, the assurers agree to pay the sum assured at the end of a fixed term of years or earlier if the life assured shall die meanwhile’.

In other words, a single benefit will be paid immediately if the insured person dies within the duration of insurance policy or if the insured person survives to the end of the insurance policy.

Endowment assurance is commonly used in practice. It assures a person that in addition to death insurance, a certain sum will be built up and paid out at the end of the insurance policy. An example is mortgage coverage, which is typically valid for the duration of the mortgage. If a person dies, the mortgage is paid off from the death insurance benefit. Furthermore, if the person survives, enough capital is built up to pay off the rest of the mortgage.

With this type of insurance, it is certain that assured capital will be paid one day, which is at the end of the insurance term or earlier.

Technically, this insurance is made up of a combination of a pure endowment insurance (see §1.6) and temporary death insurance, both with the same duration. That is why it is called mixed-type insurance.

In the case of typical endowment insurance, the benefits after death and after survival are the same, but in many cases they can be different.

The same `LifeInsuranceMath.Ax()` method can be used to calculate Endowment Insurance. In this case, in addition to the death insurance, with the `survivePayout` parameter, we have to provide an amount of benefit after survival. Note that with basic death insurances, this parameter was always 0. As shown below, endowment insurance can also be calculated as a sum of two basic insurances.

Example:

A 30-year-old man signs an insurance policy. With this policy, a €100,000 single benefit is assured if the person dies during the first 35 years. A €50,000 single benefit is also assured if the person survives 35 years.

The following code shows how to calculate the present value of endowment insurance in two different ways. In the first way, we use a single call to `Ax()` method. In the second way, we



split insurance into two basic insurances: death insurance (\$4.3) and pure endowment insurance (\$1.6). The result should be the same:

```
public void Ax_EndowmentInsurance()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.Ax(lxTable, 0.04, 30, PayoutDueDate.DirectOnDeath,
        new Insurance(100000, 0, 35), 50000);
    Console.WriteLine("Present value of the endowment insurance is {0:C}", pv);
    // Output: Present value of the endowment insurance is €17,117.31

    // This should produce the same result:
    double part1 = LifeInsuranceMath.Ax(lxTable, 0.04, 30, PayoutDueDate.DirectOnDeath,
        new Insurance(100000, 0, 35), 0);
    double part2 = LifeInsuranceMath.nEx(lxTable, 0.04, 30, 50000, 0, 35);
    pv = part1 + part2;
    Console.WriteLine("Present value of the endowment insurance is {0:C}", pv);
    // Output: Present value of the endowment insurance is €17,117.31
}
```

5.2. Combinations of whole life death insurance

5.2.1. Whole life death insurance with pure endowment insurance

This insurance type is very similar to endowment insurance, but instead of temporary death insurance, whole life death insurance is used.

Example:

A 30-year-old man signs an insurance policy. This policy assures a €10,000 single benefit if the person dies, to cover his funeral costs. The policy also assures a €50,000 single benefit if the person survives 35 years, to ensure a good start to the pension years.

The present value of such insurance can be calculated as a sum of the present values of two basic insurances:

```
public void Ax_WholeLifeDeathAndPureEndowmentInsurance()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double part1 = LifeInsuranceMath.Ax(lxTable, 0.04, 30, PayoutDueDate.DirectOnDeath,
        new Insurance(10000), 0);
    double part2 = LifeInsuranceMath.nEx(lxTable, 0.04, 30, 50000, 0, 35);
    double pv = part1 + part2;
    Console.WriteLine("Present value of the combination of whole life death insurance
        and pure endowment insurance is {0:C}", pv);
    // Output: Present value of the combination of whole life death insurance
    //         and pure endowment insurance is €12,409.81
}
```

5.2.2. Whole life death insurance with a temporary higher assurance

In this type of whole life death insurance, the death benefit is higher at the beginning of the policy for a certain number of years.

This insurance consists of two basic death insurances: Temporary death insurance with higher assured capital and deferred whole life death insurance with lower assured capital.

Example:

A 35-year-old man is insured for €100,000 if he dies before he turns 65. If he dies after 65, a single benefit of €10,000 will be paid.

The present value of this insurance is calculated with the following code:

```
public void Ax_WholeLifeDeathTemporaryHigherInsurance()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double part1 = LifeInsuranceMath.Ax(lxTable, 0.04, 35, PayoutDueDate.DirectOnDeath,
        new Insurance(100000, 0, 30), 0);
    double part2 = LifeInsuranceMath.Ax(lxTable, 0.04, 35, PayoutDueDate.DirectOnDeath,
        new Insurance(10000, 30, 0), 0);
    double pv = part1 + part2;
    Console.WriteLine("Present value of the whole life death insurance with
        temporary higher assured capital is {0:C}", pv);
    // Output: Present value of the whole life death insurance with
    //         temporary higher assured capital is €9,070.23
}
```

5.3. Death Annuity in combination with Death Insurance

In practice, death annuities are mostly combined with death insurances. It is common practice to define death annuity as a percentage of the assured capital in death insurance.

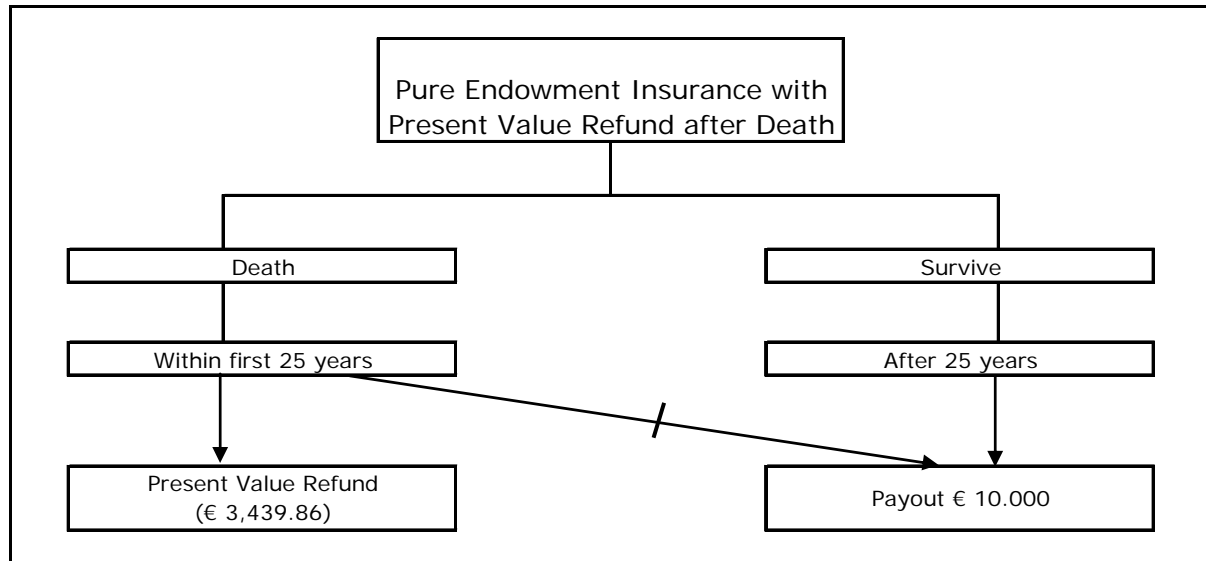
5.4. Pure Endowment Insurance combined with Present Value Refund after Death

This type of insurance consists of two parts:

- Pure Endowment Insurance. Assured capital will be paid if a person survives to the end of the insurance policy (§1.6).
- Temporary Death Insurance. The present value of this insurance will be paid if a person dies before the end of the insurance policy.

Example:

A 40-year-old man has assured capital of €10,000 if he survives 25 years. If he dies within 25 years, he will receive the net present value of this insurance.



To calculate the present value of this insurance, use the following code:

```

public void PureEndowmentWithPresentValueRefund()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double part1 = LifeInsuranceMath.nEx(lxTable, 0.04, 40, 10000.00, 0.0, 25);
    double part2 = LifeInsuranceMath.Ax(lxTable, 0.04, 40, PayoutDueDate.DirectOnDeath,
        new Insurance(1.0, 0, 25), 0.0);
    double pv = part1 / (1 - part2);
    Console.WriteLine("Present value of the pure endowment insurance is {0:C}", part1);
    Console.WriteLine("Present value of the refund after death is {0:C}", pv * part2);
    Console.WriteLine("Present value of the pure endowment insurance combined with
        present value refund after death is {0:C}", pv);
    // Output: Present value of the pure endowment insurance is €3,146.94
    // Output: Present value of the refund after death is €292.92
    // Output: Present value of the pure endowment insurance combined with
    // present value refund after death is €3,439.86
}
  
```

From the above example, we can see that the present value of the insurance is a sum of two present values:

| | |
|--|------------|
| Present value of pure endowment insurance: | € 3,146.94 |
| Present value of death insurance: | € 292.92 |
| Total present value: | € 3,439.86 |

5.5. Fixed Term Death Insurance

In this type of insurance, a single benefit is only paid out at the end of the insurance policy if the insured person dies before the end of the insurance policy.

The difference between this and other death insurances is that benefit is not paid out immediately after the death but at the end of the insurance policy.



Example:

A family of two have a child. The father is the principal earner and wants to ensure that after his death, his child has enough money to cover study costs (€10,000) at age 18.

So he takes out fixed term death insurance with a duration of 18 years.

The present value of this insurance can be calculated with the following code:

```
public void Ax_FixedTermDeathInsurance()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.nEx(lxTable, 0.04, 20, 0, 10000, 18);
    Console.WriteLine("Present value of the fixed term death insurance is {0:C}", pv);
    // Output: Present value of the fixed term death insurance is €76.93
}
```

5.6. Fixed Term Endowment Insurance

In this type of insurance, a single benefit is paid out at the end of the insurance policy. The benefit can be independent of whether the person survived or died, or it can be a different amount if the insured person dies or stays alive. One thing remains the same, however: the death and life benefits are always paid out at the end of the insurance policy, no matter when/if the insured person dies.

Example:

In the previous example, the father provided his child with capital to cover study costs after his death. If he survives, he has to finance these costs from his current income.

We will modify the previous example in such a way that the money for studying will always be available no matter if the father dies or survives. We build up this kind of capital by means of fixed term endowment insurance.

The present value of this kind of insurance is calculated with the following code:

```
public void Ax_FixedTermEndowmentInsurance()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.nEx(lxTable, 0.04, 20, 10000, 10000, 18);
    Console.WriteLine("Present value of the fixed term endowment insurance is {0:C}", pv);
    // Output: Present value of the fixed term endowment insurance is €4,936.28
}
```

5.7. Other Fixed Term Death Insurances

For other fixed term death insurances which cannot be calculated with the methods above, you can use the `LifeInsuranceMath.Ax()` method, setting the parameter `payoutDue` to the `PayoutDueDate.EndOfInsurance` value. This method covers a large set of other fixed term insurances such as deferred, increasing, decreasing capital assurances. However, the `Ax()` method is also slower. Use the `nEx()` method where possible.

6. Increasing and Decreasing Annuities

This chapter looks at a slightly different form of annuity. Life annuities are mostly used to assure retirement income. The cost of living increases from year to year. Income also has to increase to cover all costs.

Another example is a decreasing death annuity to cover mortgage payoffs.

There are a few possibilities for increasing the benefits from annuity:

- Increasing life annuity linearly with a simple rate. Benefit increases annually by a constant amount or a percentage of the first benefit.
- Increasing life annuity exponentially with a compound rate. Benefit increases annually by a percentage of the benefit in the previous year.
- Any combination of the simple and compound rate.

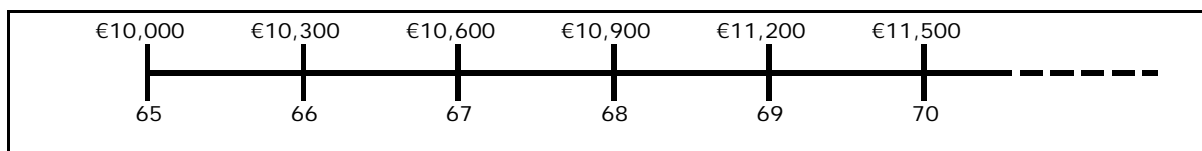
To define Decreasing or Increasing Annuity, more parameters have to be provided in the constructor of the annuity object. The following sections discuss how to use these extra parameters to create different types of increasing and decreasing annuities.

6.1. Increasing Life Annuity with a simple rate

6.1.1. Whole Life Increasing Annuity with a simple rate

Example:

A 65-year-old man has a life annuity for the rest of his life. He receives an annual benefit at the beginning of each year. The first amount he receives is €10,000. Every subsequent amount is increased by 3% of the initial value.



With this information, we can construct the annuity object with the following statement:

```
new Annuity(10000.00, 0, 0, 300.00, 0.0, 1, 0)
```

Do not forget to set the *changeDelay* parameter to 1 instead of 0, otherwise the first change will be applied to the first payment.

In this annuity, the first payment is €10,000, the second is €10,300, the third payment is €10,600 and so on.

The present value of this kind of annuity is calculated with the following code:

```
public void ax_WholeLifeIncreasing()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 65, AnnuityDueDate.BegOfPeriod,
        new Annuity(10000.00, 0, 0, 300.00, 0.0, 1, 0), zero);
    Console.WriteLine("Present value of the whole life increasing annuity is {0:C}", pv);
    // Output: Present value of the whole life increasing annuity is €133,808.19
}
```

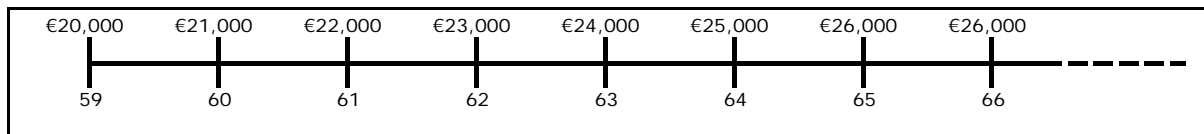
6.1.2. Whole Life Temporary Increasing Annuity

Sometimes increasing annuity is limited to a specified number of years. When the specified number of years elapses, the annuity becomes constant.

Using the *changeDuration* parameter of the annuity constructor, you can specify the duration of increasing or decreasing annuity.

Example:

In this example, we calculate the present value of the whole life annuity, where the first benefit is €20,000, the second is €21,000, and so on until €26,000. After that, it remains the same:



Age of the person: 59
 Initial benefit: €20,000
 Change amount: €1,000
 Change duration: 6
 Change delay: 1

The annuity object is created with the following statement:

```
new Annuity(20000.00, 0, 0, 1000.00, 0.0, 1, 6),
```

The present value of this annuity is calculated with the following code:

```
public void ax_WholeLifeTemporaryIncreasing()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 59, AnnuityDueDate.BegOfPeriod,
        new Annuity(20000.00, 0, 0, 1000.00, 0.0, 1, 6), zero);
    Console.WriteLine("Present value of the whole life temporary increasing annuity is
{0:C}",
        pv);
    // Output: Present value of the whole life temporary increasing annuity is €324,182.81
}
```

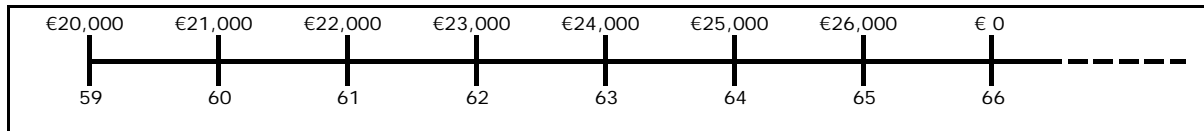
6.1.3. Temporary Life Annuity increasing with a simple rate

With a temporary life annuity, the benefit change is also limited to the duration of the insurance policy or earlier. For example, if the duration of the insurance policy is 30 years and the change duration is 40 years, the change is only used for the first 30 years.

In cases where annuity changes throughout the entire duration of the insurance policy, you can set the *changeDuration* parameter to 0. In such cases, the duration of the annuity will be used as the change duration.

Example:

In this example, we calculate the present value of the temporary life annuity, where the first benefit is €20,000, the second is €21,000 and so on until €26,000. After that, the annuity stops.



Age of the person: 59
 Initial benefit: €20,000
 Annuity Duration: 7
 Change amount: €1,000
 Change duration: 6
 Change delay: 1

The annuity object is created with the following statement:

```
new Annuity(20000.00, 0, 0, 1000.00, 0.0, 1, 6)
```

The present value of this annuity is calculated with the following code:

```
public void ax_TemporaryWholeLifeIncreasing()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 59, AnnuityDueDate.BegOfPeriod,
        new Annuity(20000.00, 0, 7, 1000.00, 0.0, 1, 6), zero);
    Console.WriteLine("Present value of the temporary annuity increasing
        with a simple rate is {0:C}", pv);
    // Output: Present value of the temporary annuity increasing
    //         with a simple rate is €136,973.42
}
```

6.2. Increasing Life Annuity with a compound rate

When increasing life annuity with a compound rate, every next payment is increased by a percentage of the previous value instead of a constant amount every year.

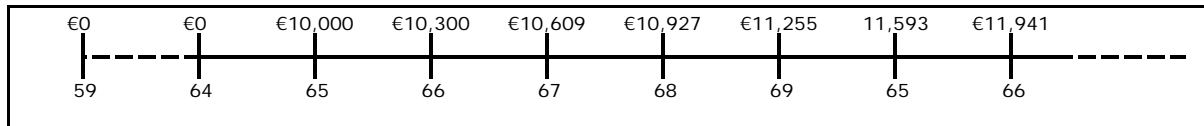


For example, given a change rate of 3% and an initial value of €1,000, we can calculate a series of payments as follows: €1,000.00, €1,030.00, €1,060.90, €1,092.73, etc.

To create such an annuity, the *changeRate* parameter is used instead of the change amount. The change rate is a 'perunage' rate, which is 100th of a 'percentage' rate, so a percentage rate of 3 is a perunage rate of 0.03.

Example:

A 59-year-old man takes out an annuity policy which starts when he is 65 and lasts for the rest of his life (whole life deferred annuity). The benefit starts at €10,000 and increases by 3% a year to cover inflation costs.



Age of the person: 59

Duration of annuity: 0 (whole life)

Initial Amount: €1,000.00

Insurance delay: 6 years

Initial amount: €10,000

Change rate: 0.03

Change delay: 1 (start at the second payment).

Change duration: 0 (whole life)

The annuity object is created with the following statement:

```
new Annuity(10000.00, 0, 0, 0.00, 0.03, 1, 0)
```

The present value of this annuity is calculated with the following code:

```
public void ax_WholeLifeCompoundIncreasing()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 59, AnnuityDueDate.BegOfPeriod,
        new Annuity(10000.00, 0, 0, 0.00, 0.03, 1, 0), zero);
    Console.WriteLine("Present value of the whole life deferred annuity increasing
        with a compound rate is {0:C}", pv);
    // Output: Present value of the whole life deferred annuity increasing
    //         with a compound rate is €176,448.06
}
```

6.3. Decreasing Life Annuity

Decreasing Life Annuity is calculated with the same method as Increasing Annuity, except that the *changeAmount* parameter is set to a negative number.

Example:

In this example, we show how to create decreasing annuity, which decreases linearly from €10,000 to €0 over a period of 10 years (€1,000 a year).



Initial amount: €10,000
Duration: 10 years
Change amount: €1,000
Change delay: 1
Change duration: 0 (the same duration as the annuity)

This annuity can be created with the following statement:

```
new Annuity(10000.00, 0, 10, 1000.00, 0.0, 1, 0)
```

6.4. Non-regularly increasing or decreasing annuity.

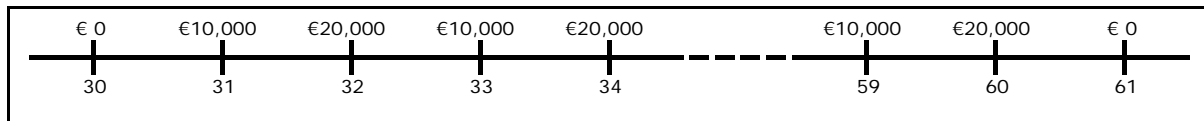
There are a couple of situations when the annuity class cannot be used to describe the annuity:

- The insurer provides its own custom list of payments per year
- Annuity changing is more complex and cannot be modelled at a simple or compound rate.

It is still possible to calculate the present value of this kind of annuity. You can use a version of the `ax()` method which allows a wide range of annual payments.

Example:

Calculate the value of insurance with a death annuity for a 30-year-old man with a duration of 30 years. The €10,000 benefit in this annuity is paid at the end of every year and another €10,000 is paid at the end of every second year:



To create this annuity, use the following code:

```
double[] payments = new double[30];  
for (int i = 0; i < 30; i++)  
    payments[i] = 10000;  
for (int i = 1; i < 30; i += 2)  
    payments[i] += 10000;
```

To calculate the present value of such an annuity assurance, use the following code:



```
public void ax_Irregular()
{
    MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

    double[] payments = new double[30];
    for (int i = 0; i < 30; i++)
        payments[i] = 10000;
    for (int i = 1; i < 30; i += 2)
        payments[i] += 10000;

    double pv = LifeInsuranceMath.ax(lxTable, 0.04, 30, AnnuityDueDate.EndOfPeriod,
        zero, payments);
    Console.WriteLine("Present value of irregular annuity is {0:C}", pv);
    // Output: Present value of irregular annuity is €6,252.97
}
```

7. Increasing and Decreasing Death Insurance

Increasing and decreasing death insurance policies are insurances in which the capital sum assurance against death (paid as a single amount after death) increases or decreases every year.

Decreasing death insurance is used directly in real life to cover loans. Increasing death insurance is mostly applied to a special self-managed 'pension' to cover the financial consequences of death before retirement age. This is why increasing and decreasing death insurances are very important.

The most common use of decreasing insurance is to cover a mortgage, while non-regularly decreasing insurance is used to cover a constant-payment mortgage.

7.1. Regularly Increasing and Decreasing Death Insurance

The calculation of regularly increasing and decreasing death insurances is very similar to the calculation of increasing and decreasing annuities, except that we use the $A_x()$ method instead of the $a_x()$ method and insurance is defined with the `Insurance` object instead of the `Annuity` object.

7.2. Non-regularly Decreasing Death Insurance

The calculation of the present value of non-regularly decreasing death insurance can be done in the same way as non-regularly decreasing annuity (§6.4).

8. Premiums

8.1. Background

The previous chapters discussed how to calculate the net value of the insurance policy. Leaving costs aside, you now know how much it can be bought or sold for.

As we discussed in §1.5, there are two ways to pay for insurance. You can buy it directly and pay at once, or you can spread payment over time. Spread payments are called premiums.

Of course, the insurer wants to receive the same total amount, no matter which method is used. In the case of premiums, because they are usually spread over a long period of time, you also have to take interest rates and the likelihood of the policyholder dying into account.

Fortunately, there is a very simple method for calculating the value of premiums. Technically, premiums are the same as a Life Annuity, which is discussed in §2. They are paid while the policyholder is alive. We already know how to calculate the value of a Life Annuity. The same method is applied to premiums.

Once again, we have to make sure that the present value of all premiums is the same as the present value of the insurance.

The calculation steps described in this section are used for background purposes. You can use these steps to calculate premiums for virtually any type of life insurance. In the following sections, we will show you practical examples of premium calculations for the most frequently used insurance types using helper methods such as $P_{Ax}()$ and $\overline{P}_{Ax}()$ in a single step.

Most premium calculations follow the same pattern:

- First, you have to decide how premiums are to be paid (e.g. whole life annuity, deferred annuity, temporary annuity, etc.).
- Choose an initial value for premiums. Since we do not know what it is at this stage, we choose €1.00. Calculate the present value of premiums using the chosen initial value. Knowing the present value of premiums with an initial value of €1.00, we can see how much higher or lower it is than the present value of insurance. We can use this value to calculate the scale factor for premiums.
- Multiply initial premiums by a scale factor.

We will demonstrate this calculation pattern with a simple example.

A 30-year-old man has death insurance with assured capital of €10,000 for the next 3 years. He wants to spread payment of this insurance equally over 3 years. How much does he have to pay?

Step 1. Calculate the value of the insurance.

Using information from §4.3, the present value of the insurance can be calculated with the following code:



```
MortalityTable lxTable = GetMortalityTable("NL", Gender.Male, "LX_9095");

double insurancePv = LifeInsuranceMath.Ax(lxTable, 0.04, 30, PayoutDueDate.DirectOnDeath,
    new Insurance(10000, 0, 3), 0);
Console.WriteLine("Present value of the death insurance is {0:C}", insurancePv);
// Output: Present value of the death insurance is €25.21
```

Step 2. Calculate the value of premiums with an initial value of €1.00
using information from §2.2.3

```
double premiumsPv = LifeInsuranceMath.ax(lxTable, 0.04, 30, AnnuityDueDate.BegOfPeriod,
    new Annuity(1.00, 0, 3), zero);
Console.WriteLine("Present value of premiums, not scaled is {0:C}", premiumsPv);
// Output: Present value of premiums, not scaled is €2.88
```

Step 3. Calculate the scale factor.

```
double scaleFactor = insurancePv / premiumsPv;
Console.WriteLine("Scale factor for premiums is {0:0.00}", scaleFactor);
// Output: Scale factor for premiums is 8.74
```

Step 4. Scale premiums.

Since we have chosen €1.00 a year as an initial premium, we can use the scale factor to calculate premiums as $8.74 \times €1.00 = €8.74$ per year:

```
double premiumPerYear = scaleFactor * 1.00;
Console.WriteLine("Premium per year is {0:C}", premiumPerYear);
// Output: Premium per year is €8.74
```

In the above example, premiums are equal every year and we only scaled the first premium. For increasing and decreasing premiums, we have to scale initial premiums for every year. We can use the `Annuity.ToArray()` method to get an array of premiums for each year:

```
Annuity premiumsAnnuity = new Annuity(1.00, 0, 3);
double[] premiumsArray = premiumsAnnuity.ToArray(3);
for (int i = 0; i < premiumsArray.Length; i++)
{
    premiumsArray[i] = scaleFactor * premiumsArray[i];
    Console.WriteLine("Premium in the year {0} is {1:C}", i+1, premiumsArray[i]);
}
// Output: Premium in the year 1 is €8.74
// Output: Premium in the year 2 is €8.74
// Output: Premium in the year 3 is €8.74
```

Note that we can also create scaled premiums with following statements:



```
Annuity premiumsAnnuity = new Annuity(scaleFactor, 0, 3);  
double[] premiumsArray = premiumsAnnuity.ToArray(3);
```

All these steps are combined in a single `PAX()` method:

```
double[] premiumsArray = LifeInsuranceMath.PAX(lxTable, 0.04, 30,  
AnnuityDueDate.BegOfPeriod,  
PayoutDueDate.DirectOnDeath, new Insurance(10000, 0, 3), 0,  
new Annuity(1.00, 0, 3), false);  
  
for (int i = 0; i < premiumsArray.Length; i++)  
    Console.WriteLine("Premium in the year {0} is {1:C}", i + 1, premiumsArray[i]);  
// Output: Premium in the year 1 is €8.74  
// Output: Premium in the year 2 is €8.74  
// Output: Premium in the year 3 is €8.74
```

8.2. Gross value of insurance and gross premiums

Costs are divided into two categories:

Fixed costs

Fixed costs are costs which are paid only once for each insurance policy. These are costs which the insurer incurs when creating a policy, such as:

- Commission costs to intermediary
- Inspections
- Salaries, consultants
- Advertisements
- Product development
- Costs to create the policy
- Administration

Recurrent day-to-day expenses

These costs are paid regularly during the lifetime of the policy, such as:

- Money collection (for periodic payments)
- Payouts
- Administration

It is sometimes not very clear to which category you have to assign costs (e.g. Administration and Salaries).

If we look at the basic figures on which the calculation of insurance is based ($\$1$), we can see that insurers can only compete with costs. An efficient insurer can lower costs and therefore charge lower gross premiums.



9. Appendices

9.1. Mortality tables in Xml Format

Mortality tables in Xml format contain mortality figures for one or more populations. It can be used by a client application or for data exchange between online figures such as the Online Mortality Web Service, the SQL data source and the locally hosted xml resource.

The location of the xml file is defined by the application itself. It can be a standalone xml file, compiled resource or any other streamable data source.

Figlo.Actuarial.Mortality.Data component defines methods for reading (and possibly writing) those data sources.

This is a snapshot of an xml file containing mortality tables:

```
<mortalitytables version="1.0.0.0">
  <countries>
    <country code="NL" name="The Netherlands" />
    <!-- other countries -->
  </countries>
  <mortalitytable countrycode="NL" name="LX_0005" gender="Male" description=""
    year="2001" yearcount="5" version="AAAAAAAAAHg=">
    <mortalityrecord age="0" lx="10000000" />
    <mortalityrecord age="1" lx="9945646" />
    <!-- other mortality records -->
  </mortalitytable>
  <mortalitytable countrycode="NL" name="LX_0005" gender="Female" description=""
    year="2001" yearcount="5" version="AAAAAAAAAHg=">
    <mortalityrecord age="0" lx="10000000" />
    <mortalityrecord age="1" lx="9945646" />
    <!-- other mortality records -->
  </mortalitytable>
  <!-- other mortality tables -->
</mortalitytables>
```

9.1.1. mortalitytables element

This is a root element required in every xml file.

```
<mortalitytables version="1.0.0.0">
  <!-- Country codes -->
  <!-- Mortality tables -->
</mortalitytables>
```

Attributes

| | |
|----------------|--|
| version | Required attribute. Specifies the version of Mortality format. Currently, this attribute must be set to '1.0.0.0'. |
|----------------|--|



Child Elements

- countries** Specifies a collection of countries in the xml file with country codes and country names. This is an optional element; it is useful but can be omitted if only country codes are used.
- mortalitytable** Specifies the mortality table entry with specific name, country code and gender.

9.1.2. countries element

This element specifies a collection of countries in the xml file with country codes and country names. This is an optional element; it is useful but can be omitted if only country codes are used.

```
<countries>
  <!-- Country elements -->
</countries>
```

Child Elements

- country** Country element, containing the country code and the full name of the country.

Parent Elements

- mortalitytables** Root element of the xml file.

9.1.3. country element

The country element contains the country code and the full name of the country.

```
<country code="NL" name="The Netherlands" />
```

Attributes

- code** Country code. This is a unique country identifier.
- name** The full name of the country.

Parent Elements

- countries** A collection of countries in the xml file with country codes and country names.



9.1.4. mortalitytable element

Defines the mortality table with a specific name, gender and country code.

```
<mortalitytable countrycode="NL" name="LX_0005" gender="Male" description=""
  year="2001" yearcount="5" version="AAAAAAAAAHg=">
  <!--Mortality records -->
</mortalitytable>
```

Attributes

| | |
|--------------------|---|
| countrycode | Required attribute. Specifies the country code for this mortality table. |
| name | Required attribute. Specifies the name of the mortality table. The name must be unique within the same country code and gender. This means that a maximum of two mortality tables with the same name can exist in an xml file for the same country code. One table is for male and the other one is for female. |
| gender | The gender of the person. Possible values are 'Male' and 'Female'. |
| description | Optional description of the mortality table. |
| year | The year that identifies the start period for the mortality statistics contained in this table. This attribute is used together with the yearcount attribute to identify the total duration of the statistical data. |
| yearcount | Number of years which identifies the total duration of the statistical data. It is used together with the year attribute. |
| version | Optional version of the mortality table. It can be used to synchronise mortality tables in a locally stored xml file with online mortality tables. The SQL server generates versions automatically with a timestamp data field. This kind of version is an array of 8 bytes which is Base64-encoded in the xml file. To decode the version back into an array of bytes, the following method from .Net framework can be used: |

```
byte[] vbytes = Convert.FromBase64String(attr.Value)
```

The following method from .Net framework can be used to convert the byte array back into a string:

```
string vstring = Convert.ToBase64String(vbytes)
```

Child Elements

| | |
|------------------------|---|
| mortalityrecord | Mortality record containing the number of living people of a given age. |
|------------------------|---|



Parent Elements

mortalitytables Root element of the xml file.

9.1.5. mortalityrecord element

The mortality record contains the number of living people of a given age.

```
<mortalityrecord age="0" lx="10000000" />
```

Attributes

age Age of the person.

lx Number of people who survive to this age (usually, but not necessarily scaled to a population size of 100,000).

Parent Elements

mortalitytable Specifies the mortality table entry with a specific name, country code and gender.

9.2. Class diagram

LifeInsuranceMath
Class

Methods

- ax (+ 1 overload)
- Ax (+ 2 overloads)
- axy (+ 1 overload)
- Axy (+ 2 overloads)
- nEx (+ 1 overload)
- nExy (+ 1 overload)
- Pax
- PAX (+ 2 overloads)
- Paxy
- PAXy (+ 2 overloads)
- Paxy_OneYearYSurviverPremium

Nested Types

Annuity
Class

Fields

- changeAmount
- changeDelay
- changeDuration
- changeRate
- delay
- duration
- initialPayment

Methods

- Annuity (+ 4 overloads)
- ToArray

AnnuityDueDate
Enum

- BegOfPeriod
- EndOfPeriod
- Continous

Insurance
Class

Fields

- changeAmount
- changeDelay
- changeDuration
- changeRate
- delay
- duration
- initialCapital

Methods

- Insurance (+ 4 overloads)
- ToArray

AnnuityScenarios
Class

Properties

- BothDie
- BothSurvive
- OnlyFirstSurvive
- OnlySecondSurvive
- UsedScenarios

Methods

- AnnuityScenarios (+ 2 overloads)

Scenarios_xy
Enum

- None
- OnlyFirstSurvive
- OnlySecondSurvive
- BothSurvive
- BothDie
- AtLeastOneSurvive
- OnlyOneSurvive
- AtLeastOneDie

PayoutDueDate
Enum

- DirectOnDeath
- EndOfDeathYear
- EndOfInsurance

AnnuityScenario
Class

Properties

- Annuity
- Scenarios_xy

Methods

- AnnuityScenario

9.3. Common Dutch Translations

| English | Dutch |
|--|--|
| Benefit payout | Uitkering |
| Death benefit payout | Overlijdensuitkering |
| Pure Endowment Assurance | Kapitaalverzekering bij leven |
| Annuity | Rente, Annuiteit |
| Life Annuity | Lijfrente |
| Death Annuity | Erfrente, ideaalrente, opvoedingsrente, gezinsrente |
| Whole Life Annuity | Levenslange Lijfrente |
| Deferred Whole Life Annuity | Uitgestelde Levenslange Lijfrente |
| Temporary Life Annuity | Dadelijk Ingaande Tijdelijke Lijfrente |
| Deferred Temporary Life Annuity | Uitgestelde Tijdelijke Lijfrente |
| Death Insurance; Capital Sum Assurance | Kapitaalverzekering Bij Overlijden; Risicoverzekering |
| 1-year Death Insurance | Éénjarige risicoverzekering |
| Whole Life Death Insurance | Levenslange kapitaalverzekering bij overlijden |
| Temporary Death Insurance | Tijdelijke kapitaalverzekering bij overlijden |
| Deferred Death Insurance | Uitgestelde kapitaalverzekering bij overlijden |