Lab 3b : ALM-analysis

Enterprise Risk Management
Autumn 2021

Supervised by:
Markus Ådahl

November 11, 2021

Umeå University
Department of Mathematics and Mathematical Statistics

Filip Forsgren (fifo0010@student.umu.se)

Contents

1 Task 1: Explore the data

1.1 Description

a) Plot the average, over the five years of data, of the mean population for each age for both men and women in the same figure.

b) Plot the average, over the five years of data, of the number of deaths for each age for both men and women in the same figure.

1.2 Solution

This Task was solved in Excel (see handed in .xlsx file). Arithmetic means for the period 2016-2020 are calculated using excel function `Average` for populations and deaths for men and women separately.

1.3 Result

In Figures 1 and 2 the average population and deaths for men and women in the period 2016-2020 are presented.

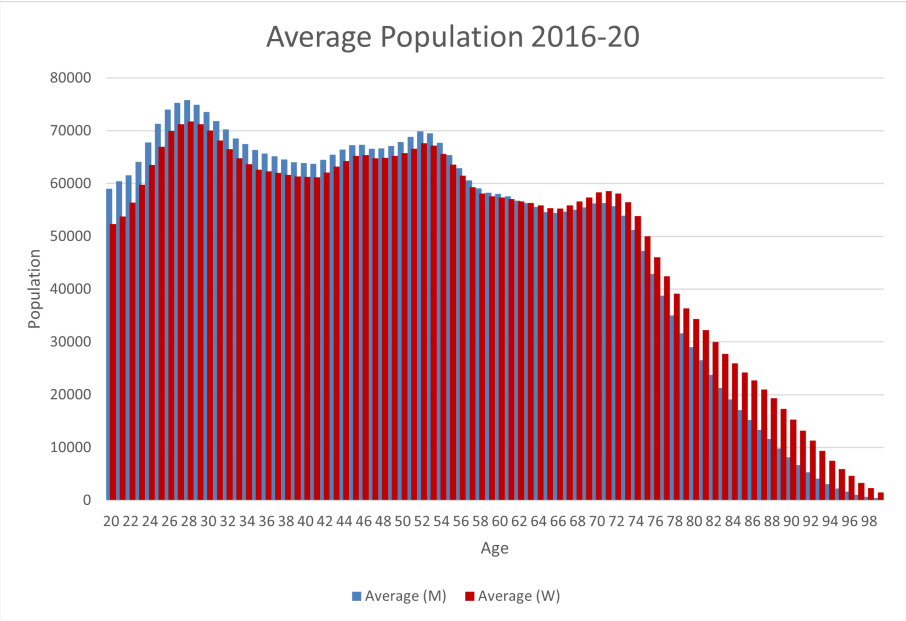

Figure 1: average population for men and women in the period 2016-2020.

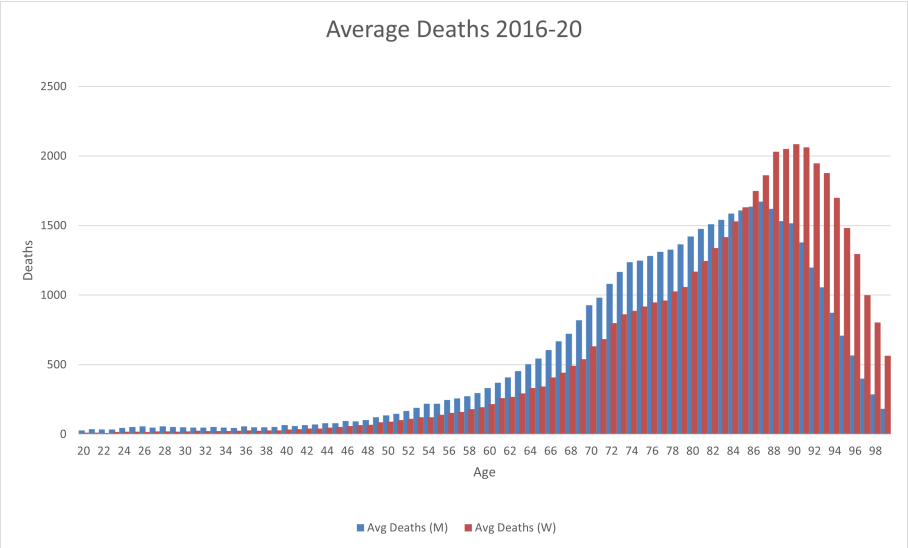

Figure 2: average number of deaths for men and women in the period 2016-2020.

1.4 Comments

From the Result in Section 1.3 the conclusion can be drawn that males tend to die earlier than women. Also, women's intensity of mortality seems to peak higher (around the age of 90) compared to males, where the intensity is *spread out* in younger ages.

2 Task 2: Mortality estimation

2.1 Description

a) Estimate the force of mortality for men and women separately using the SCB-data and plot these estimates in the same figure. Do this for ages 20 – 99 and use data for all the years (2015 – 2019) for a more stable estimate.

b) Fit Makeham functions using only force of mortality estimates for ages 20 – 90 from above. Then plot the estimated forces of mortality, for ages 20 – 99, together with the Makeham functions in the same figure (both men and women). Include the parameters in the report.

c) Plot the survival functions for men and women, for ages 0 – 105, in the same figure (but not the same as in b).

d) Calculate life expectancy for both men and women, according to the lifetime distributions given by the Makeham functions

2.2 Solution

Also, this Task was solved in Excel (see handed in .xlsx file).
As for a), Arithmetic means for population and deaths from Task 1 (Section 1) are used to estimate force of mortality, $\hat{\mu}$, by dividing average number of deaths for one age group with the average population for the same age group, $\hat{\mu} = \frac{AverageDeaths_{age}}{AveragePopulation_{age}}$.

b) the *Makeham Function* are defined as $\hat{\mu}(x) \approx \alpha + \beta e^{\gamma x}$, where α, β and γ are parameters to be optimized and x is the age group. This function is defined in one cell for every age group. The squared error is defined in one cell as $(\frac{AverageDeaths_{age}}{AveragePopulation_{age}} - (\alpha + \beta e^{\gamma x}))^2$. Using Python function `optimize.minimize` from `SciPy` package (See Section 5.1 for explicit implementation), the minimum square error is found by adjusting α, β and γ. Hence, deriving optimal parameters $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$.

c) Using $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ from b) we can estimate the survival function from age 0-105 as $e^{-\hat{\alpha}x - \frac{\hat{\beta}}{\hat{\gamma}}\left(e^{\hat{\gamma}x}-1\right)}$, where x is the age.

d) The life expectancy, $E[T] = \int_0^\infty e^{-\mu x}$ is estimated using Riemann sums for ages 0-140.

2.3 Result

In Figure 3 the estimated Force of Mortality is presented. In Figure 4 the estimated Force of Mortality together with the Makeham Function is presented and in Table 1 the estimated parameters are presented. In Figure 5 the estimated Survival Function is presented. In Table 2 the life expectancy is presented.

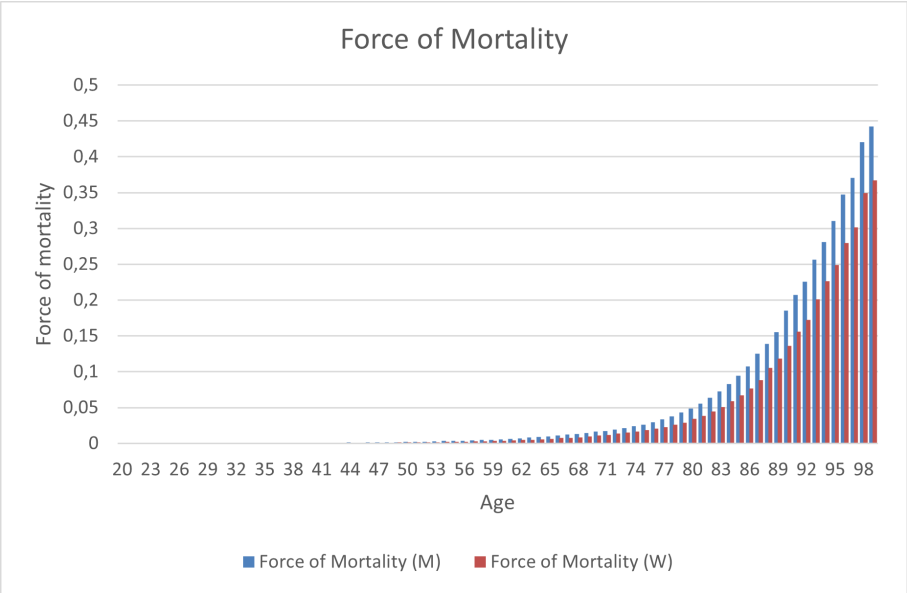

Figure 3: Estimated Force of Mortality.

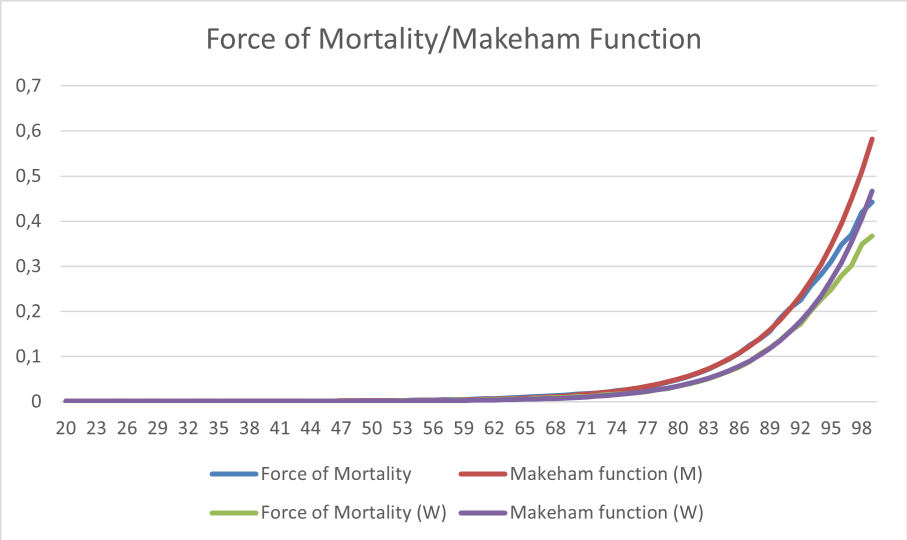

Figure 4: Estimated Force of Mortality and Makeham Function.

Gender	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\gamma}$
Male	0.0012028	0.000001459	0.13024904
Female	0.0008721	0.000000546	0.13793626

Table 1: Estimated parameters.

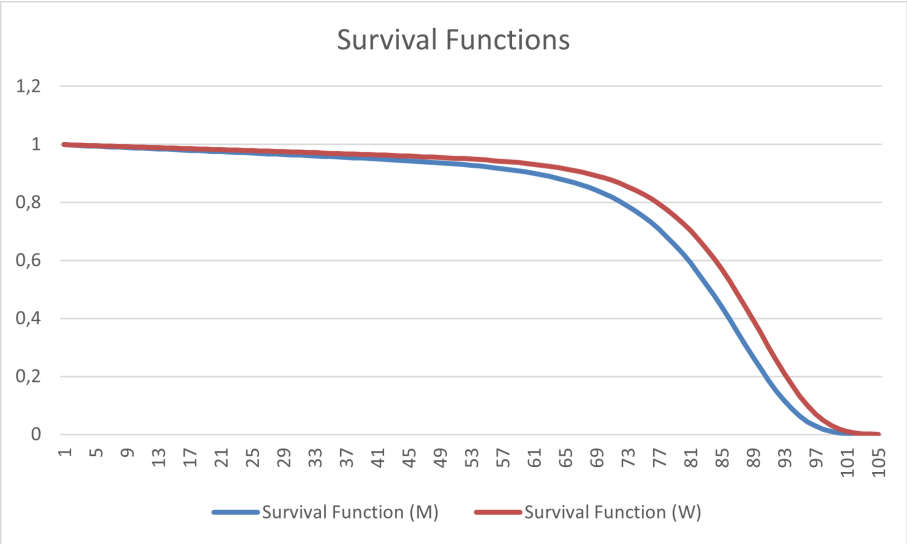

Figure 5: Estimated Survival Function.

Gender	**Life Expectancy**
Male	82.82
Female	79.02

Table 2: Estimated parameters.

3 Task 3: The state model

3.1 Description

Implement the state model for the policy and plot the populations (PP, PO, TR, DE) for each time period 30 years (360 months) into the future. Use the area plot to visualize the size of the populations over time.

3.2 Solution

The probability of death in a time period (1 month), $P_m(death)$, is calculated using:

$$q(x) = 1 - \frac{l\left(x + \frac{1}{12}\right)}{l(x)}$$

where $l(x)$ is

$$e^{-\hat{\alpha}x - \frac{\hat{\beta}}{\hat{\gamma}}\left(e^{\hat{\gamma}x} - 1\right)}$$

where x is the age in the beginning of the period and $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ are the estimate from Task 2 (Section 2)

The probability of transfer in a time period (1 month), $P(transfer)$, derived from the annual transfer probability P_a by $P_m(transfer) = 1 - (1 - P_a)^{1/12}$.

using a transition matrix with above transition probabilities the populations throug time can be derived. See Section 5.2 for explicit python implementation.

3.3 Result

In Figure 6 the evolution of the population through time is presented.

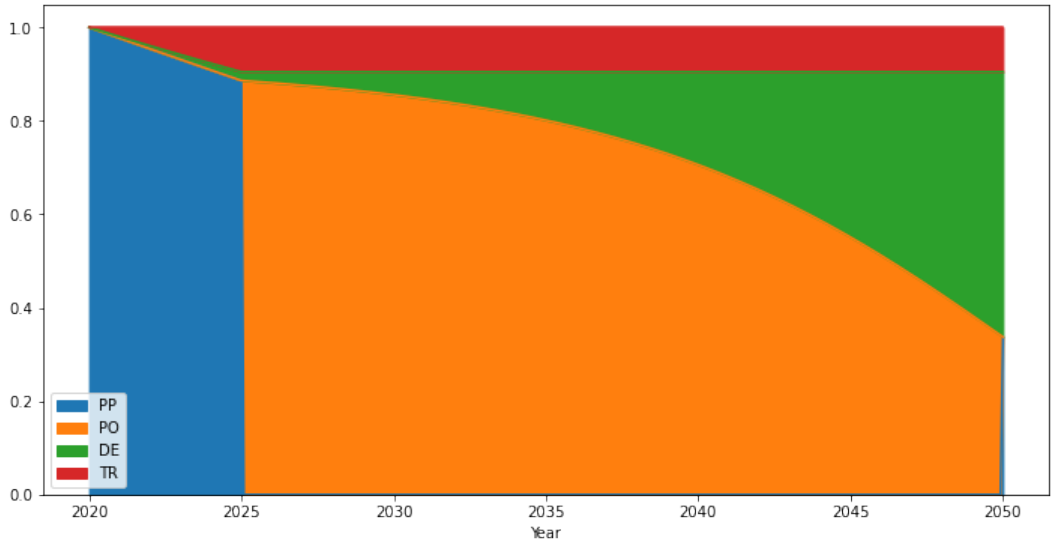

Figure 6: Evolution of the population. PP: Premium Paying, PO: Premium Out, DE: dead, TR: Transfered

4 Task 4: 1st order payout and reserve

4.1 Description

Calculate the guaranteed payout amount and project the 1st order reserve for the policy.

a) Determine the yearly payout amount, b.

b) Plot the evolution of the reserve 30 years into the future.

c) Include a table with the value of the reserve for every 10 years.

4.2 Solution

For a), b is calculated using formula:

$$b = \frac{G(x) + \frac{p}{12} \sum_{t=1}^{12u} \frac{l(x+t)}{l(x)} e^{-r \cdot \frac{t}{12}}}{(1/12) \sum_{t=12u+1}^{12(u+s)} \frac{l\left(x+\frac{t}{12}\right)}{l(x)} \cdot e^{-r\frac{t}{12}}}$$

where x is the age at the beginning of 2020, P is the yearly premium, u is the time of pension, s is the last date for payout (assumed to be 120) and r is the yearly interest rate.

For b) and c), is calculated by recursively calculate $G(t+1) = G(t) * e^r + premium_m + riskpremium_m$ where G(t) is the first order reserve in time t, r is the guarantied monthly interest rate, $premium_m$ is the monthly premium and $riskpremium_m$ is the monthly risk premium, until t+1=2025. Where the *premium* is replaced by a payout of size b from a). For explicit Python implementation see Section 5.3.

4.3 Result

The yearly payout amount, b is 168 506.24.
In Figure 7 the evolution of the first order reserve trough 2020-2050 is presented.
In Table 3 the first order reserve trough 2020-2050 for every 10 year is presented.

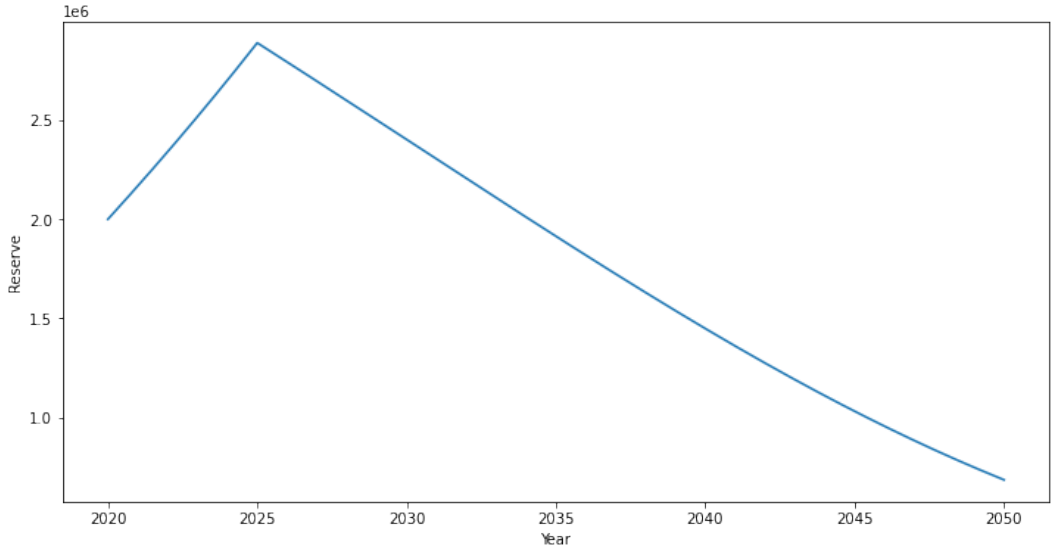

Figure 7: Evolution of the first order reserve trough 2020-2050

Year	**Reserve Amount**
2020	2 000 000.00
2030	2 401 997.38
2040	1 449 002.49
2050	686 397.27

Table 3: 10 year reserves.

5 Appendix

5.1 Task 2

```
class makeham(object):

    def __init__(self, estFOM, ages, gamma=0.1378):
        self.ages = ages
        self.estFOM = estFOM
        self.parameters = self.optParams(self.estFOM, gamma)

    def sqrError(self,gamma):
        m = np.sum(self.estFOM)
        g = 0
        f1 = 0
        f2 = 0

        for i in range(len(self.ages)):
            g += self.estFOM[i] * np.exp(gamma*self.ages[i])
            f1 += np.exp(gamma*self.ages[i])
            f2 += np.exp(2*gamma*self.ages[i])

        alpha = (m*f2 - g*f1)/(len(self.ages)*f2-np.square(f1))
        beta = (m*f1 - len(self.ages)*g)/(np.square(f1)-len(self.ages)*f2)

        sumSqrError = 0
        for i in range(len(self.estFOM)):
            sumSqrError += np.square(self.estFOM[i] - (alpha +
            beta*np.exp(gamma*(self.ages[i]))))
        return sumSqrError

    def optParams(self, estFOM, gamma):

        opt = optimize.minimize(self.sqrError, gamma)
        gamma = opt.x[0]

        m = np.sum(self.estFOM)
        g = 0
        f1 = 0
        f2 = 0

        for i in range(len(self.ages)):
            g += self.estFOM[i] * np.exp(gamma*self.ages[i])
            f1 += np.exp(gamma*self.ages[i])
            f2 += np.exp(2*gamma*self.ages[i])
```

```
        alpha = (m*f2 - g*f1)/(len(self.ages)*f2-np.square(f1))
        beta = (m*f1 - len(self.ages)*g)/(np.square(f1)-len(self.ages)*f2)

        parameters = [alpha, beta, gamma]
        return parameters


femaleMakeham = makeham(estFOMFemale, ages)
femaleParameters = femaleMakeham.parameters

maleMakeham = makeham(estFOMMale, ages)
maleParameters = maleMakeham.parameters

print(femaleParameters, maleParameters)
```

5.2 Task 3

```
def transitionMatrix(age, parameters, moveProb):
    '''
    gives transition matrix for population of age=age for 1 month
    '''
    alpha = parameters[0]
    beta = parameters[1]
    gamma = parameters[2]

    PPtoDE = 1 - ((np.exp(-alpha*(age+(1/12))-(beta/gamma)*
        (np.exp(gamma*(age+(1/12)))-1)))/
        (np.exp(-alpha*age-(beta/gamma)*(np.exp(gamma*age)-1))))

    if age < 65:
        PPtoOUT = (1 - PPtoDE)*moveProb
    else:
        PPtoOUT = 0

    PPtoPP = 1 - PPtoDE - PPtoOUT

    matrix = np.array(([PPtoPP,0,0],[PPtoDE,1,0],[PPtoOUT,0,1]))

    return matrix


yearsAhead = 30
age0 = 60
t0 = 0
steps = np.arange(0,30+(1/12),1/12)
```

```
populations = np.zeros((len(steps),4))
populations[0] = [1,0,0,0]

for i in range(len(steps)-1):
    matrix = transitionMatrix(age0+steps[i], parametersWomen, moveProb)
    populations[i+1,:3] = np.matmul(matrix,populations[i,:3])
    if age0+steps[i] > 65:
        populations[i,3] = populations[i,0]
        populations[i,0] = 0

popDF = pd.DataFrame(populations,columns=['PP','DE','TR','PO'])
popDF['Year'] = steps+2020
area = popDF.plot.area(x = 'Year', y = ['PP','PO','DE','TR'])
plt.show()
```

5.3 Task 4

```
def payout(age0,z,s,interest,G0,parameters):

    alpha = parameters[0]
    beta = parameters[1]
    gamma = parameters[2]

    ## PP:

    ages = np.arange(age0,z,1/12)
    t = 1
    scaling = 0

    for age in ages:
        scaling += (
            ((np.exp(-alpha*(age+(1/12))-(beta/gamma)*
            (np.exp(gamma*(age+(1/12)))-1)))/
             (np.exp(-alpha* age0         -(beta/gamma)*
             (np.exp(gamma* age0)         -1))))
             * np.exp(-interest*t)
        )
        t += 1
    G = G0 + 10000 * scaling

    ## PO:
    ages = np.arange(z,s,1/12)
    scaling = 0

    for age in ages:
        scaling += (
```

```
            ((np.exp(-alpha*(age+(1/12))-(beta/gamma)*
            (np.exp(gamma*(age+(1/12)))-1)))/
             (np.exp(-alpha* age0          -(beta/gamma)*
             (np.exp(gamma* age0 )         -1))))
             * np.exp(-interest*t)
        )
        t += 1

    b = G / (scaling/12)

    return b

payout = payout(60,65,120,rm,G0,parameters)
print('Yearly payout: ',payout)


#b)
monthlyPayout = payout/12

def reserve(age0,z,s,interest,G0,parameters,payout):

    alpha = parameters[0]
    beta = parameters[1]
    gamma = parameters[2]

    reserves = np.zeros((s-age0)*12+1)
    reserves[0] = G0
    ages = np.arange(age0,s,(1/12))
    i = 1

    for age in ages:
        #PP
        if age<65:
            q = (1-((np.exp(-alpha*(age+(1/12))-(beta/gamma)*
            (np.exp(gamma*(age+(1/12)))-1)))/
                    (np.exp(-alpha* age          -(beta/gamma)*
                    (np.exp(gamma* age          )-1)))))
            riskPremium = reserves[i-1]*np.exp(interest)*(q/(1-q))
            reserves[i] = reserves[i-1]*np.exp(interest) + 10000 + riskPremi

        #PO
        else:
            q = (1-((np.exp(-alpha*(age+(1/12))-(beta/gamma)*
            (np.exp(gamma*(age+(1/12)))-1)))/
                (np.exp(-alpha* age          -(beta/gamma)*
                (np.exp(gamma* age          )-1)))))
            riskPremium = reserves[i-1]*np.exp(interest)*
```

```
            (q/(1-q))
            reserves[i] = reserves[i-1]*np.exp(interest)
            - payout + riskPremium
        i += 1

    return reserves


reserves = reserve(60,65,120,rm,G0,parameters,monthlyPayout)

plt.rcParams['figure.figsize'] = [12, 6]

plt.plot(steps+2020,reserves[:len(steps)])

plt.ylabel('Reserve')
plt.xlabel('Year')
plt.show()

reserve = np.zeros(4)
for i in range(0,4):
    reserve[i] = reserves[i*120]

print(reserve)
```