УДК 004.415

ПОГРАММНАЯ РЕАЛИЗАЦИЯ МОДЕЛЕЙ ДОЛГОСРОЧНОГО СТРАХОВАНИЯ ЖИЗНИ НА ЯЗЫКЕ PYTHON

Сафина Г.Ф.

к. ф.-м. н, доцент,

ФГБОУ ВО «Уфимский университет науки и технологий», Нефтекамский филиал,

Нефтекамск, Россия

Гатиятова А.А.

студент,

ФГБОУ ВО «Уфимский университет науки и технологий», Нефтекамский филиал,

Нефтекамск, Россия

Аннотация

В работе представлена программная реализация актуарных моделей долгосрочного страхования жизни на языке Python с использованием фреймворка Flask. Описаны математические модели расчета нетто-премий, нетто-резервов и дисперсии приведенной стоимости выплат для пяти видов договоров страхования жизни. Реализована веб-система, обеспечивающая полный операционный цикл работы со страховым портфелем: ведением базы клиентов и договоров, актуарный расчет, сценарный анализ визуализацию результатов. Проведена верификация алгоритмов на аналитических примерах.

Ключевые слова: Python, Flask, страхование жизни, актуарная математика, нетто-премия, страховой резерв, таблица смертности, Гомпертц.

SOFTWARE IMPLEMENTATION OF LONG-TERM LIFE INSURANCE MODELS IN PYTHON

Safina G.F.

PhD, Associate Professor,
Neftekamsk branch of the Ufa University of Science and Technology,
Neftekamsk, Russia
Gatiyatova A.A.
student,
Neftekamsk branch of the Ufa University of Science and Technology,
Neftekamsk, Russia

Abstract

The paper presents a software implementation of actuarial models for long-term life insurance in Python using the Flask framework. Mathematical models for calculating net premiums, net reserves and variance of the present value of payments for five types of life insurance contracts are described. A web-based system has been implemented that provides a complete operational cycle for working with an insurance portfolio: client and contract database management, actuarial calculations, scenario analysis and results visualization. The algorithms have been verified on analytical examples.

Keywords: Python, Flask, life insurance, actuarial mathematics, net premium, insurance reserve, mortality table, Gompertz.

Долгосрочное страхование жизни является одним из важнейших инструментов финансовой защиты населения. Договоры страхования жизни, заключаемые на срок от пяти до тридцати лет и более, требуют применения специализированного математического аппарата – актуарной математики – для корректного расчёта страховых тарифов и формирования резервов. Без учёта вероятностного характера момента страхового события и временной стоимости денег страховая компания не способна обоснованно оценить свои будущие обязательства.

Актуарным расчетам в страховании жизни посвящены труды многих других авторов [1-4]. Вопросы практического применения таблиц смертности в российских условиях и в моделях краткосрочного страхования рассматриваются в работах [5-7]. В данном же исследовании практическое применение рассматривается для долгосрочного страхования и разрабатывается программное приложение для расчетов основных параметров моделей страхования.

В части программных решений известны библиотека «lifecontingencies» для языка R, пакет «lifelib» для Python, а также коммерческие актуарные системы Prophet (FIS) и Moses (Moody's Analytics) [4, 7].

Вместе с тем открытых систем, сочетающих прозрачный актуарный код, полноценный веб-интерфейс и немедленную готовность к работе, практически нет. Программное решение данного исследования призвано восполнить такой пробел.

Известно, что центральным объектом актуарной модели является остаточное время жизни T_x – случайная величина, равная числу лет, которое проживёт лицо в возрасте *x* сверх текущего момента.

Основные показатели смертности в долгосрочном страховании жизни для первоначально наблюдаемой группы из l_0 человек представлены в таблице 1 [5, 7]:

Таблица 1– Показатели таблицы смертности

Обозначение	Название	Формула / Описание
l_x	Число живущих в возрасте x	$l_x = l_0 \cdot \prod_{k=0}^{x-1}(1-q_t)$
d_x	Число умерших в возрасте x	$d_x = l_x - l_{x+1}$
q_x	Вероятность смерти в возрасте x	$q_x = d_x / l_x$
p_x	Вероятность дожития на 1 год	$p_x = 1 - q_x = l_{x+1} / l_x$

e_x	Ожидаемая продолжительность жизни	$e_x = (1/l_x)\cdot\sum_{t\geq 1} l_{x+t}$

Основные характеристики дожития застрахованного возраста в течение следующих *t* лет представлены в следующей таблице 2.

Таблица 2– Показатели дожития в долгосрочном страховании

Вероятность дожития ${}_t p_x$	${}_t p_x = P(T_x \geq t) = \prod_{k=0}^{t-1}(1-q_{x+k}).$
EPV срочного страхования	$A_{x:n}\Big\vert^1 = \sum_{t=0}^{n-1} {}_t p_x \cdot {}_t q_x \cdot \upsilon^{t+1}.$
EPV смешанного страхования	$A_{x:n}\Big\vert = A_{x:n}\Big\vert^1 + {}_n p_x \cdot \upsilon^n.$
Аннуитет взносов	$a_{n\vert} = (1-\upsilon^n)/i.$
Нетто-премия (принцип экв.)	$P = A_{x:n}\vert / a_{x:n}\vert$
Проспективный резерв t=1	${}_tV = S\cdot A_{x+t:n-t}\Big\vert - P\cdot a_{x+t:n-t}\Big\vert$,
Дисперсия (Бернулли)	$D(z) = S^2\cdot q_x\cdot(1-q_x)$

В моделях таблицы 2:

– υ^t – коэффициент дисконтирования (показывает, сколько стоит сегодня денежная единица, которую получим через t лет: $\upsilon^t = 1/(1+i)^t$);

– S – страховая сумма резерва;

– q_x – вероятность смерти в возрасте *x* лет, взятая из таблицы смертности или рассчитанная по закону Гомпертца: $q_x = a\cdot e^{bx}$. Для мужчин используются параметры *a* = 0,00022, *b* = 0,097, для женщин – *a* = 0,00015, *b* = 0,090.

В представленной работе реализованы модели для пяти видов договоров: срочного страхования на случай смерти (term life), пожизненного страхования (whole life), смешанного страхования (endowment), накопительного страхования и аннуитета. Для каждого вида применяются соответствующие формулы EPV и аннуитета взносов.

Система разработана на языке Python 3.11 с использованием микрофреймворка Flask 2.x и ORM Flask-SQLAlchemy. Архитектура приложения трёхзвенная: уровень представления (Jinja2-шаблоны), уровень бизнес-логики (маршруты и актуарный модуль) и уровень данных (SQLite/PostgreSQL). Выбор Flask вместо более тяжёлых фреймворков (Django, FastAPI) обусловлен минимализмом и прозрачностью: каждый маршрут, каждая модель данных и каждый актуарный алгоритм явно видны в коде без «магии» фреймворка [8].

Актуарные расчёты сосредоточены в модуле actuarial.py (505 строк, 16 функций), написанном на чистом Python без использования numpy или scipy. Это решение принято намеренно: каждая формула реализована явно и может быть непосредственно сопоставлена с её математическим определением. Ключевые функции модуля:

– survival_probability(qx, age, t) – вероятность дожития ${}_t p_x$;

– annuity_immediate(i, n, age, gender) – актуарный аннуитет взносов $A_{x:n}|$;

– net_premium_term_life – нетто-премия по принципу эквивалентности;

– net_reserve(S, i, n, qx, age, t, P) – проспективный резерв □V;

– accumulation_schedule(premium, rate, term) – график накопления.

Приведем далее листинг кодирования ключевых функций модуля actuarial.py:

```
def survival_probability(qx_list: list, age: int, t: int) ->
float:
```

```
    prob = 1.0
    for k in range(t):
        current_age = age + k
        if current_age < len(qx_list):
            prob *= (1 - qx_list[current_age])
        else:
            prob *= (1 - 0.5)
    return prob
def annuity_immediate(i: float, n: int) -> float:
    if i == 0:
        return float(n)
    v = 1 / (1 + i)
    return (1 - v ** n) / i
def net_premium_term_life(
    S: float,
    i: float,
    n: int,
    qx_list: list,
    age: int,
) -> dict:
    v = 1 / (1 + i)
    epv_death = sum(
        death_probability(qx_list, age, t) * (v ** (t + 1))
        for t in range(n)
    ) * S
    ann = annuity_immediate(i, n)
    net_p = epv_death / ann if ann > 0 else 0.0
    return {
        "epv":        round(epv_death, 4),
        "annuity":    round(ann,       4),
```

```
            "net_premium": round(net_p,    4),
        }
    def net_reserve(
        S: float,
        i: float,
        n: int,
        qx_list: list,
        age: int,
        t: int,
        net_p: float,
        insurance_type: str = "term",
    ) -> float:
        if t >= n:
            return 0.0
        remaining = n - t
        current_age = age + t
        if insurance_type == "endowment":
            result   =   net_premium_endowment(S,   i,   remaining,
qx_list, current_age)
            epv_future = result["epv_total"]
        else:
            result   =   net_premium_term_life(S,   i,   remaining,
qx_list, current_age)
            epv_future = result["epv"]
        ann_future = annuity_immediate(i, remaining)
        reserve = epv_future - net_p * ann_future
        return round(max(reserve, 0.0), 4)
    def accumulation_schedule(
        premium: float,
        rate: float,
```

```
        term: int,
        frequency: str = "yearly",
    ) -> list:
        periods_per_year = 12 if frequency == "monthly" else 1
        total_periods    = term * periods_per_year
        r_period         = rate / periods_per_year
        rows        = []
        accumulated = 0.0
        for t in range(1, total_periods + 1):
        accumulated  = (accumulated + premium) * (1 + r_period)
interest       = accumulated - (accumulated / (1 + r_period)) -
premium + \
(accumulated / (1 + r_period) * r_period)
          # упрощённо: проценты за период
            interest     = accumulated / (1 + r_period) * r_period
            year = math.ceil(t / periods_per_year)
            rows.append({
                "period":      t,
                "year":        year,
                "premium":     round(premium,      2),
                "accumulated": round(accumulated,  2),
                "interest":    round(interest,     2),
            })
        return rows
```

Схема базы данных включает девять таблиц. Центральные сущности – Client и Policy. С договором связаны: Payment (платёж), Claim (страховая выплата), PolicyHistory (история изменений), Reserve (резерв). Отдельно хранятся MortalityTable и MortalityRate – для загрузки реальных таблиц смертности. При первом запуске система автоматически заполняет таблицу

смертности значениями Гомпертца через функцию _seed_mortality_table(), что обеспечивает работоспособность расчётов без ручного ввода данных.

Веб-интерфейс включает 16 страниц, сгруппированных по разделам: клиенты, договоры, платежи, выплаты, актуарный расчёт, сценарный анализ, резервы, таблицы смертности, накопление. Визуализация реализована через Chart.js (8 графиков: платёжный поток, динамика резерва, возрастное распределение, кривая смертности qx и др.). Математические модели (формулы) отображаются в браузере через MathJax.

Для проверки корректности реализованных алгоритмов результаты программы сравнивались с аналитическими решениями, полученными по формулам актуарной математики. В таблице 3 приведены результаты восьми тестов.

Таблица 3 – Сравнение результатов программы с аналитическими решениями

Тест	Параметры	Аналитическое значение	Значение программы	Отклонение
Аннуитет a_n	i=5%, n=10	7,7217	7,7217	0%
Дисконтный множитель v	i=5%, t=1	0,9524	0,9524	0%
q_x (Гомпертц, М, 40 лет)	a=0,00022, b=0,097	0,010653	0,010653	0%
Нетто-премия term life	S=100 000, i=5%, n=20, x=35	1 420,85	1 420,85	0%
Нетто-премия endowment	S=100 000, i=5%, n=20, x=35	3 471,38	3 471,38	0%
Накопление (5 лет)	P=1 000, r=5%, n=5, ежегодно	5 801,91	5 801,91	0%
Резерв t=1 (term)	S=100 000, i=5%, n=20, x=35	883,36	883,36	0%
Нетто-премия (задача 3)	x=25, n=3, S=100 000, i=25%	51 300,00	51266,10	0,07%

По семи из восьми тестов отклонение равно нулю – результаты программы совпадают с аналитическими формулами при машинной точности. Небольшое отклонение 0,07% в задаче 3 объясняется тем, что аналитическое значение приведено с точностью до 100 руб., тогда как программа даёт более точный результат 51 266,10 руб.

Таким образом, работе показана реализация системы моделирования долгосрочного страхования жизни на языке Python. Система включает 1 740 строк Python-кода в четырёх модулях, 22 HTML-шаблонов и реализует полный операционный цикл работы со страховым портфелем. Все актуарные алгоритмы верифицированы на аналитических примерах с нулевым или пренебрежимо малым отклонением от теоретических значений. Система может применяться страховыми компаниями в учебных и демонстрационных целях, а также служить основой для разработки полноценного коммерческого актуарного инструмента.

Библиографический список:

1. Flask Documentation. Version 2.x [Электронный ресурс]. – URL: https://flask.palletsprojects.com (дата обращения: 01.05.2026).
2. SQLAlchemy Documentation. Version 2.0 [Электронный ресурс]. – URL: https://docs.sqlalchemy.org (дата обращения: 01.05.2026).
3. Фалин Г. И., Фалин А. И. Актуарная математика в задачах. – М.: МГУ, 2003. – 192 с.
4. Бауэрс Н. Л., Гербер Х. У., Хикман Дж. К. и др. Актуарная математика / пер. с англ. – М.: Янус-К, 2001. – 656 с.
5. Королёв В. Ю., Бенинг В. Е. Введение в математическую теорию надёжности. – М.: URSS, 2003. – 176 с.
6. Сафина Г.Ф. Вероятностные задачи актуарной математики: учебный практикум / Г.Ф. Сафина. – Уфа: РИЦ БашГУ, 2017. – 144 с.

7. Сафина Г.Ф., Садрисламова А.Р. Применение математического пакета к приближенным моделям страхования жизни / Заметки ученого, 2020. №9. – С. 78-82.

8. Основы программирования на языке Python: учеб. пособие / под ред. А. Ю. Богачева. – Краснодар: РЭА, 2023. – 135 с.