Федеральное государственное автономное образовательное учреждение высшего профессионального образования

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ МАТЕМАТИКИ И МЕХАНИКИ

КАФЕДРА ТЕОРИИ ФУНКЦИЙ И ПРИБЛИЖЕНИЙ

Специальность: 010100 – Математика

Специализация: Действительный анализ

ДИПЛОМНАЯ РАБОТА

Аппроксимация случайных величин

Работа завершена:

«___»_____2015 г. _____С.Р. Рамазанов

Работа проверена:

Научный руководитель
кандидат физико-математических наук,
доцент кафедры теории функций и приближений

«___»_____2015 г. _____А.Ф. Галимянов

Заведующий кафедрой теории функций и приближений,
доктор физико-математических наук, профессор

«___»_____2015 г. _____Ф.Г. Авхадиев

Казань – 2015 г.

Оглавление

Ведение **2**

1 Предварительные сведения 3

1.1 Обозначения 3

1.2 Метод наименьших квадратов 3

1.3 Полином Лагранжа 6

1.4 Кубический сплайн 6

1.5 Случайная величина 8

1.6 Случайный процесс 10

2 Аппроксимация случайного процесса 11

3 Аппроксимация случайной величины 12

3.1 Метод регрессии 12

3.2 Надежность 14

4 Метод аппроксимации случайной величины 15

5 Ошибка аппроксимации методом 16

Вывод **17**

Список литературы **18**

Приложение **19**

Код программ 19

Метод Лагранжа 19

Интерполяционный кубический сплайн 29

Интерполяционный полином Лагранжа 35

Введение

В настоящие время очень актуальна тема как прогнозирование. Но никогда нельзя сказать наверняка, что прогноз точен с вероятностью в 100 процентов. Можно приблизить предыдущие значения и на основе приближения сказать чему равно следующие значение. Но в данном случае получается большая ошибка предсказания. По этому в настоящие время есть предложения прогнозировать с некоторым промежутком, то есть верхнее значение и нижнее значения прогнозируемой величины. А.В. Сульдин в своей статье предложил метод регрессии для представления следующей случайной величины в виде линейной комбинации предыдущих.

Для написания данной работы была поставлена цель: получить метод прогнозирования случайной величины. Предыдущим результатом был метод аппроксимирующий случайную величину на основании двух методов, что стало основоположной идеи для нового метода. В новом методе используется результат А.В. Сульдина, метод регрессии и обычная аппроксимация функции.

Набор случайных величин разделяется на две составляющие. Первой составляющей является линейная функция построенная по случайным величинам с помощью метода наименьших квадратов. После чего строится новый набор случайных величин, который есть абсолютная разница между линейной функцией и случайными величинами в точке. Новый набор случайных величин используется в методе регрессии А.В. Сульдина. И на выходе требуется сложить следующее значение линейной функции и результат метода регрессии, что бы получить приблизительное значение следующей случайной величины.

1 Предварительные сведения

В этой главе показываются некоторые результаты, на которые опираемся в данной работе. Во первых, это приближения полиномами и оценки данных приближений. Во вторых, в работе применяется аппроксимация кубическими сплайнами. А так же представлен метод регрессии А.В. Сульдина, который будет рассмотрен очень подробно.

1.1 Обозначения

Для начала предлагаю ввести некоторые обозначения, что бы ввести некую ясность в работе.

Пусть имеются случайные величины: $\xi_0, \xi_1, ..., \xi_n$. Это величины, которые отвечают за так называемую "историю"процесса. Каждая из них ставится в соответствии с временной переменной t. Имеем следственно разбиение временного промежутка $a = t_0 < t_1 < ... < t_n = b$. И получается привязка случайных величин ко времени $\xi(t_0), \xi(t_1), ..., \xi(t_n)$.

С помощью метода наименьших квадратов находится общий тренд движения случайной величины, обозначим его через функцию f.

Величины, которые используются в методе регрессии представляются как абсолютная разница случайных величин и функции тренда и обозначаются через $\eta(t_i)$:

$$\eta(t_i) = |f(t_i) - \xi(t_i)|, \quad i = \overline{0, n}.$$

Ниже представлены методы аппроксимации, которые используются в данной работе.

1.2 Метод наименьших квадратов

Пусть измерения физической величины a дают значения $\xi_0, \xi_1, ..., \xi_n$. Будем считать, что эти значения представляют собой сумму a и случайных погрешностей

$$\triangle_i = x_i - a, \quad (i = 0, ..., n),$$

причем погрешность $\triangle_i$ независимы в совокупности и нормальны. Далее, будем считать, что

$$E(\triangle_i) = 0; \quad E(\triangle_i^2) = \sigma^2$$

Равенство $E(\triangle_i) = 0$ означает **несмещенность измерений** (отсутствие систематической ошибки). То, что число σ, неизвестное нам, одно и то же для всех значений i, означает **равноточность измерений**.

Требуется найти приближенное значение физической величины a на основание наблюдений $\xi_0, \xi_1, ..., \xi_n$. Как мы видели, это задача оценивания параметра a при повторной выборке. Применим ее для ее решения способ максимального правдоподобия. Функция правдоподобия имеет вид

$$L(\xi_0, \xi_1, ..., \xi_n, a) = \prod_{i=0}^{n} f(\xi_i, a),$$

где

$$f(\xi_i, a) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(\xi_i - a)^2}{2\sigma^2}}.$$

Таким образом,

$$L(\xi_0, \xi_1, ..., \xi_n, a) = (2\pi)^{-\frac{n}{2}} \sigma^{-n} e^{-\frac{1}{2\sigma^2} \sum_{i=0}^{n} (\xi_i - a)^2}. \qquad (1)$$

Предписание максимального правдоподобия требует при заданном σ выбирать $a = \hat{a}$ так, что бы было

$$L(\xi_0, \xi_1, ..., \xi_n, a) = \max.$$

Из (1) очевидно, что это равносильно условию

$$Q = \sum_{i=0}^{n} (\xi_i - a)^2 = \min.$$

которое и есть предписание наименьших квадратов.

Введя среднее выборочное

$$\bar{\xi} = \frac{1}{n} \sum_{i=0}^{n} \xi_i,$$

находим

$$Q = \sum_{i=0}^{n}(\xi_i - a)^2 = \sum_{i=0}^{n}(\xi_i - \bar{\xi})^2 + n(a - \bar{\xi})^2,$$

откуда видно, что $minQ$ получается тогда и только тогда, если заменим a на $\bar{\xi}$, т.е.

$$\hat{a} = \bar{\xi}.$$

Итак, оценка по методу наименьших квадратов в данном случае дает $\hat{a} = \bar{\xi}$ в качестве приближенного значения a. Выясним реальный смысл такого приближения.

Имеем

$$E(\bar{\xi}) = a; \quad D(\bar{\xi}) = \frac{\sigma^2}{n}.$$

Поскольку $\bar{\xi}-$ нормальная случайная величина, то

$$\bar{\xi} \in N(a, \frac{\sigma}{\sqrt{n}}).$$

В частности, будем иметь

$$P\{|\bar{\xi} - a\} > \frac{1,96\sigma}{\sqrt{n}} = 0,05.$$

При большом числе наблюдений n мы будем иметь, при работе с оценкой $\bar{\xi}$, почти наверное приближение порядка $\frac{2\sigma}{\sqrt{n}}$ к измеряемой величине a. Однако изредка(с вероятностью $< 0,05$) возможны и большие отклонения. Отклонение более чем на $\frac{3\sigma}{\sqrt{n}}$ имеем вероятность около $0,003$. Далее, $\bar{\xi}$ является эффективной оценкой для a, ибо имеем для весьма широкого класса других несмещенных оценок t параметра a неравенство

$$D(t) \geq \frac{\sigma^2}{n} = D(\bar{\xi}).$$

Таким образом, на основании сказанного, реальный смысл оценки $\bar{\xi}$ по методу наименьших квадратов таков: в весьма широком классе асимптотически несмещенных и асимптотически нормальных оценок t имеем при заданном сколь угодно малом $\epsilon > 0$

$$P\{|\bar{\xi} - a| > \epsilon\} \leq P\{|t - a| > \epsilon\},$$

1.3 Полином Лагранжа

Требуется найти полином $L_n(t)$ степени не выше чем n, который совпадал бы со случайными величинами $\eta(t_0), ..., \eta(t_n)$. Таким образом, что бы выполнялось условие:

$$L_n(t_k) = \eta(t_k), \quad k = 0, ..., n.$$

Тогда полином находится по формуле:

$$L(t) = \sum_{i=0}^{n} \eta(t_i) l_i(t),$$

где базисные полиномы находятся по формуле:

$$l_i(t) = \prod_{j=0, j\neq i}^{n} \frac{t - \eta(t_j)}{\eta(t_i) - \eta(t_j)}.$$

Таким образом имеем полином степени не выше чем n, который аппроксимирует поведение случайной величины.[6]

1.4 Кубический сплайн

Пусть $\mathbb{P}_m$ множество многочленов степени не выше m, $m = 3$, и $C^{(k)} = C^{(k)}[a, b]$ множество непрерывных на $[a, b]$ функций, имеющих непрерывную k-ю производную, $k \in \mathbb{Z}_+$.

Определение: Функцию

$$S_m(t) = S_{m,k}(t, \triangle_n),$$

где $\triangle_n$ - есть разбиение отрезка времени, называют полиномиальным сплайном степени m дефекта k, $(1 \leq k \leq m)$ с узлами $t_0, ..., t_n$, если

1) $S_m(t) \in \mathbb{P}_m$ на $[t_i, t_{i+1}]$, $(i = 0, 1, 2, ..., n)$;

2) $S_m(t) \in C^{m-k}[a, b]$

Точки $\{t_i\}$ называют узлами сплайна.

Для однозначного определения коэффициентов кубического сплайна требуются:

1) Условие интерполяции: введем сетку равноотстоящих экспериментальных значений $\{\eta(t_i)\}_{i=0}^n$. Тогда условие интерполяции имеет вид:

$$S_3(t_i) = \eta(t_i).$$

2) Краевые условие: как известно для однозначного определения коэффициентов кубического сплайна требуется, кроме уравнений условий интерполяции, еще два уравнения. Обычно на интерполяционный сплайн налагаются дополнительные условия:

$$S_3^{(i)}(a) = S_3^{(i)}(b) \quad (i = 1, 2).$$

Положим

$$h_i = t_{i+1} - t_i \text{ и } M_i = S_3^{(2)}(t_i),$$
$$m_i = S_3'(t_i) \qquad (i = 0, 1, 2, ..., n). \qquad (1)$$

Тогда

$$S_3(t) = M_{i-1}\frac{(t_i - t)^3}{6h_{i-1}} + M_i\frac{(t - t_{i-1})^3}{6h_{i-1}} +$$
$$+(\eta(t_{i-1}) - \frac{M_{i-1}h_{i-1}^2}{6})\frac{t_i - t}{h_{i-1}} + (\eta(t_i) - \frac{M_i h_{i-1}^2}{6})\frac{t - t_{i-1}}{h_{i-1}},$$
$$S_3(t) = m_i\frac{(t_i - t)^2(t - t_{i-1})}{h_{i-1}^2} - m_i\frac{(t - t_{i-1})^2(t_i - t)}{h_{i-1}^2} +$$
$$+\eta(t_{i-1})\frac{(t_i - t)^2[2(t - t_{i-1}) + h_{i-1}]}{h_{i-1}^3} +$$
$$+\xi_i\frac{(t - t_{i-1})^2[2(t_i - t) + h_{i-1}]}{h_{i-1}^3}.$$

Положим

$$\lambda'_{i+1} = \frac{h_{i+1(h_i - \overline{h}_i)}}{\overline{h}_i(h_i + h_{i+1})}, \quad \beta'_{i+1} = \frac{\overline{h}_{i+1} h_i}{(h_{i+1} - \overline{h}_{i+1})},$$

$$\lambda_{i+1} = \frac{h_{i+1}}{(h_i + h_{i+1})}, \quad \beta_{i+1} = \frac{h_i}{h_i + h_{i+1}},$$

где $0 < \overline{h}_i < h_i$.

Тогда

$$d_i = 6\delta(\eta(t_{i-1}), \eta(t_i), \eta(t_{i+1})),$$

$$c_i = 3\lambda_i \delta(\eta(t_{i-1}), \eta(t_i)) + 3\beta_i \delta(\eta(t_i), \eta(t_{i+1})) \quad (i = 1, 2, ..., n-1),$$

где $\delta(\eta(t_i), \eta(t_{i+1}))$ есть первая разделенная разность, а $\delta(\eta(t_{i-1}), \eta(t_i), \eta(t_{i+1}))$- вторая разделенная разность.

Тогда, учитывая условия интерполяции, гладкости и краевые условия, определяем параметры M_i, соответственно m_i, из систем:

$$\begin{cases} \beta_i M_{i+1} + 2M_i + \lambda_i M_{i+1} = d_i, \quad (i = 1, 2, ..., n), \\ M_0 = M_n, \quad M_{n+1} = M_1, \quad h_n = h_0, \\ \lambda_i m_i + 2m_i + \beta_i m_{i+1} = c_i \quad (i = 1, 2, ..., n) \\ m_0 = m_n, \quad m_{n+1} = m_1. \end{cases}$$

Найдя все требуемые коэффициенты получаем непрерывный полином третьей степени на каждом из отрезков $[t_i, t_{i+1}]$. Что собой представляет на отрезке [a,b] интерполяционный кубический сплайн. [5]

1.5 Случайная величина

Рассмотрим $n+1$ случайных величин $\eta(t) = \eta(t_0), \eta(t_1), ..., \eta(t_n)$ с нулевыми средними значениями и конечными моментами второго порядка, где $\eta(t)$ - будем рассматривать в виде абсолютной разницы $f(t_k)$ и изначальной случайной величины $\xi(t_i), \quad i = 0, 1, ..., n$:

$$\eta(t_i) = |\xi(t_i) - f(t_i)|, \quad i = 1...n.$$

$$E\xi(t_i) = 0,\ E\xi(t_i)\xi(t_k) = \lambda_{ik},\ |\lambda_{ik}| < +\infty, ik = 0, 1, ..., n.$$

Тогда среднее будет иметь вид:

$$E\eta(t_i) = E(\xi(t_i) - f(t_i)) = E(\xi(t_i)) - E(f(t_i)) = E(f(t_i))$$

А их конечный момент:

$$\begin{aligned} E\eta(t_i)\eta(t_k) = E((\xi(t_i) - f(t_i))(\xi(t_k) - f(t_k))) = \\ = E(f(t_i)f(t_k) - f(t_i)\xi(t_k) - f(t_k)\xi(t_i) + \xi(t_i)\xi(t_k)) = \\ = E(f(t_i)f(t_k)) - E(f(t_i)\xi(t_k)) - E(f(t_k)\xi(t_i)) + \\ + E(\xi(t_i)\xi(t_k)) = E(f(t_i)f(t_k)) + \lambda_{ik}. \end{aligned}$$

Будем аппроксимировать случайную величину $\eta(t_k) = \xi(t_k) - f(t_k)$ линейными комбинациями случайных величин $\eta(t_0), \eta(t_1), ..., \eta(t_n)$. Наилучшей линейной оценкой для $\eta(t_k)$ будем называть то приближение, для которого среднее

$$E[\eta(t_k) - \sum_{i=1}^{n} C_i\eta(t_i)]^2$$

имеет наименьшее значение.

В нашем случае мы имеем такое определение:

$$E[\eta(t_k) - \sum_{i=1}^{n} C_i\eta(t_i)]^2 =$$

$$= E[\xi(t_k) - f(t_k) - \sum_{i=1}^{n} C_i(\xi(t_i) - f(t_i))]^2 =$$

$$= E[\eta(t_k) - f(t_k) - \sum_{i=1}^{n} C_i\eta(t_i)]^2 + \sum_{i=1}^{n} C_i f(t_i) =$$

$$= E[(\xi(t_k) - \sum_{i=1}^{n} C_i\xi(t_i)) - (f(t_k) - \sum_{i=1}^{n} C_i f(t_i))]^2$$

Плоскость в $n+1$-мерном пространстве

$$\eta(t_k) = \sum_{i=1}^{n} C_i \eta(t_i) = \sum_{i=1}^{n} C_i \xi(t_i) - \sum_{i=1}^{n} C_i f(t_i)$$

соответствующая наилучшей оценке, обычно называется плоскостью средней квадратичной регрессии величины $\eta(t_k)$ относительно $\eta(t_0), \eta(t_1), ..., \eta(t_n)$.

Если Λ есть матрица вторых моментов случайных величин $\eta(t_0), \eta(t_1), ..., \eta(t_n)$: $\Lambda = (\lambda_{ik}), \quad i, k = 0, 1...n, \quad \Lambda-$ алгебраическое дополнение элемента λ_{ik}, то коэффициенты C_i наилучшей оценки находится по формуле:

$$C_i = -\frac{\Lambda_{0i}}{\Lambda_{00}}.$$

Но так же надо иметь ввиду, что у величины $\eta(t_i), i = 0, ..., n$ есть добавок от функции тренда. Он так же находится через алгебраическое дополнение и по свойству линейности матрицы разделяются на две через суму.

1.6 Случайный процесс

Имеется случайная величина $\eta(t_k) = \xi(t_k) - f(t_k)$ и случайный дискретный процесс $x(t_i) = \xi(t_i) - f(t_i)$, $a = t_0 \leq t_1 \leq ... \leq t_n = b$. Причем, так же существует случайный процесс $y(t)$ от величин $\xi(t_0), ..., \xi(t_n)$ и его среднее

$$Ey(t) = 0.$$

Тогда найдем среднее случайного процесса $x(t)$:

$$Ex(t) = E(\xi(t) - f(t)) = E(\xi(t)) - E(f(t)) = E(f(t)).$$

Так же найдем произведение процессов:

$$\begin{aligned} Ex(t)x(s) &= E((\xi(t) - f(t))(\xi(s) - f(s))) = \\ &= E(f(t)f(s) - \xi(t)f(s) - f(t)\xi(s) + \xi(t)\xi(s)) = \\ &= E(f(t)f(s)) - E(\xi(t)f(s)) - E(\xi(s)f(t)) + \\ &+ E(\xi(t)\xi(s)) = E(f(t)f(s)) + r(t, s), \end{aligned}$$

где $r(t, s)$ - есть $E(y(s)y(t))$.

Аналогичным способом найдем произведение случайной величины и процесса:

$$E\eta(t_k)x(t) = E((\xi(t_k) - f(t_k))(\xi(t) - f(t))) =$$
$$= E(f(t_k)f(t) - \xi(t_k)f(t) - f(t_k)\xi(t) + \xi(t_k)\xi(t)) =$$
$$= E(f(t_k)f(t)) - E(\xi(t_k)f(t)) - f(t_k)E(\xi(t))+$$
$$+\xi(t_k)E(\xi(t)) = f(t_k)E(f(t)) - \xi(t_k)E(f(t)).$$

Таким образом получаем необходимые обозначения и выкладки, для дальнейшей работы.

2 Аппроксимация случайного процесса

По предложению метода регрессии для аппроксимации случайной величины, требуется непрерывный случайный процесс. Для этого дискретный процесс предлагается приблизить двумя способами.

Первый способ, заключается в приближении дискретного случайного процесса с помощью полинома степени не выше, чем n. В нашем случае предлагается аппроксимировать его с помощью полином Лагранжа. Тогда ошибка данной аппроксимации будет иметь вид:

$$r(t) = \frac{f^{(n+1)}(\zeta)}{(n+1)!}\prod_{i=0}^{n+1}(t - t_i), \quad \zeta \in [a, b].$$

Прошу заметить, что многочлен Лагранжа точен при малых значениях, допустим $n < 20$.

Второй способ, предполагает приближение проводить с помощью кубического сплайна, так как его точность будет выше. Это происходит из-за того, что не требуется ограничится в выборе количества узлов для аппроксимации. Приведем оценку данного метода аппроксимации:

$$\|f^{(i)}(t) - S^{(i)}_{m,\triangle_n}(t)\|_{L_p[a,b]} \leq c\|\triangle_n\|^{1-\frac{1}{p}+\frac{1}{q}}\omega_{m-i}(f^{(i+1)}, \|\triangle_n\|)_q),$$

где $1 \leq p \leq q \leq \infty$.

Имеется наилучшая константа для оценки, приведем ее:

$$\|f^{(i)}(t) - S^{(i)}_{m,\triangle_n}(t)\|_{L_p[a,b]} \leq \frac{5}{348}h^4\|\frac{d^4f}{dx^4}\|.$$

Теперь имея непрерывный процесс, который может быть получен двумя способами, можно перейти к самому методу.

3 Аппроксимация случайной величины

Аппроксимация случайного процесса происходит так, что бы функции $r(t,s)$, $t \neq s$, $r(t,t)$, $q(t)$ имели непрерывные производные по t.

Возьмем n значений t

$$a \leq t_1 < t_2 < ... < t_n \leq b$$

и рассмотрим n случайных величин $x(t_i)$, $i = 1, 2, ..., n$.

Здесь появляется вопрос о наилучшей оценке $\eta(t_k)$ с помощью линейных комбинаций

$$\sum_{i=1}^{n} C_i x(t_i).$$

Однако в этом случае у нас больше возможностей. Мы остановимся на ряде вариантов, различных по своему значению. Прежде всего отметим, что наилучшая линейная оценка всегда ниже будет пониматься, как оценка, минимизирующая(в некотором смысле) среднее значение:

$$E(\eta(t_k) - \sum_{i=1}^{n} C_i x(t_i))^2. \qquad (1)$$

3.1 Метод регрессии

Проводить приближение случайной величины можно следующими способами:

1. Узлы t_i фиксированы, минимизация (1) происходит по C_i. Это обычный метод средней квадратической регрессии.

2. Константы C_i фиксированы, минимизация производится по t_i. Для определения t_i легко получаем систему уравнений

$$C_k \frac{d}{dt_k} r(t_k, t_k) + 2 \sum_{j=1, j\neq k, k=1,2,...,n}^{n} C_j \frac{d}{dt_k} r(t_k, t_j) = 2 \frac{dq(t_k)}{dt_k}. \quad (2)$$

3. Узлы t_i фиксированы, минимизация происходит по C_i при некоторых дополнительных условиях на C_i

$$\varphi(C_1, ..., C_n) = 0, \quad k = 1, ..., m; \quad m < n. \quad (3)$$

Система уравнений для определения C_i состоит из (3) и

$$\sum_{j=1}^{n} C_j r(t_i, t_j) - q(t_i) + \sum_{j=1}^{m} \lambda_j \frac{d\varphi_j(C_1, ..., C_n)}{dC_i} = 0, \quad (4)$$

$$i = 1, 2, ..., n.$$

4. Коэффициенты C_i фиксированы, минимизация ведется по t_i при некоторых дополнительных условиях на t_i:

$$\varphi(t_1, ..., t_n) = 0, \quad k = 1, ..., m; \quad m < n. \quad (5)$$

Система уравнений для определения t_i состоит из (5) и

$$C_i \frac{d}{dt_i} r(t_i, t_i) + 2 \sum_{j=1, j\neq i}^{n} C_j \frac{d}{dt_i} r(t_i, t_j) +$$

$$+ \sum_{k=1}^{m} \lambda_k \frac{d\varphi_j(t_1, ..., t_n)}{dt_i} = 2 \frac{dq(t_i)}{dt_i}.$$

$$i = 1, 2, ..., n.$$

5. Минимизация происходит как по C_i, так и по t_i. Система уравнений для определения C_i и t_i состоит из (2) и

$$\sum_{i=1}^{n} C_i r(t_i, t_k) = q(t_k), \quad k = 1, 2, ..., n.$$

6. Минимизация происходит как по C_i, так и по t_i, на C_i и t_i наложены дополнительные условия.

7. Зачастую представляет интерес производить усреднение по условной мере. В этом случае так же возможны все шесть вариантов.

8. Если случайная величина $\eta(t_k)$ сама зависит от t_i, то в этом случае оценку

$$\eta(t_k) \approx \sum_{i=1}^{n} C_i(t)x(t_i)$$

естественно назвать наилучшей, если она минимизирует выражение

$$E\{\int_a^b [\eta(t) - \sum_{i=1}^{n} C_i(t)x(t_i)]^2 dt\},$$

при условии, естественно, что оно конечно. [4]

3.2 Надежность

Пусть $\eta_0, \eta_1, ..., \eta_n$ - случайный величины. Приближенная формула

$$\eta_k \approx \sum_{i=1}^{n} C_i\eta_i$$

будет иметь погрешность

$$\rho = \eta - \sum_{i=1}^{n} C_i\eta(t_i).$$

Погрешность ρ так же является случайной величиной.

Пусть

$$P(|\rho| > \varepsilon) = 1 - \frac{p}{100}.$$

Тогда будем говорить,что надежность формулы, то есть вероятность получить погрешность меньшую ε, равно p в процентах.

В частном случае, когда все случайные величины нормальны и имеют нулевые средние, ρ так же будет нормальна (O, V), где

$$V^2 = D^2\rho = E[\eta - \sum_{i=1}^{n} C_i\eta(t_i)]^2.$$

Если обозначить

$$u = \frac{1}{V}, \Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{t^2}{2}} dx,$$

то в последнем случае

$$P(|\rho| > \varepsilon) = 2[1 - \Phi(\varepsilon u)]$$

и, следовательно, надежность равна [4]

$$p = [2\Phi(\varepsilon u) - 1].$$

4 Метод аппроксимации случайной величины

Метод аппроксимации заключается в суммировании результатов двух методов. Выше показаны вспомогательный материал для аппроксимации случайной величины.

Приближается случайная величина ξ_k, которая есть сумма двух величин $f(t_k)$ и $\eta(t_k)$.

$$\xi_k = f(t_k) + \eta(t_k).$$

Для аппроксимации величины $f(t_k)$ берется результат курсовой работы: общий тренд аппроксимируется с помощью метода наименьших квадратов. А для величины $\eta(t_k)$ берется результат Сульдина: представление случайной величины с помощью линейной комбинации случайных величин $\eta(t_0), ..., \eta(t_n)$.

Но А.В. Сульдин замечает, что для аппроксимации случайной величины, требуется непрерывность случайного процесса. А мы имеем дискретный случайный процесс. Для того, что бы можно было использовать метод регрессии, предлагается аппроксимировать случайные величины $\eta(t_0), ..., \eta(t_n)$, интерполяционным полиномом Лагранжа либо интерполяционным кубическим сплайном.

5 Ошибка аппроксимации методом

Остается вопрос о точности показанного метода. Нельзя показать общую оценку метода, но как видно, что ошибка приближения состоит из двух определенных оценок. Первая погрешность возникает конечно же при использовании метода наименьших квадратов. Она равна:

$$r_1(t) = max_{i=\overline{1,n}}(\xi_i - (at_i + b))^2.$$

Далее ошибка возникает при интерполяции случайных величин для того, что бы получить непрерывный случайный процесс. Там используется два метода: полиномиальная аппроксимация Лагранжа и кубический сплайн.

$$1)\ r_{21}(t) = \frac{f^{(n+1)}(\zeta)}{(n+1)!}\prod_{i=0}^{n+1}(t - t_i), \quad \zeta \in [a, b].$$

И

$$2)\ r_{22}(t) = \|f^{(i)}(t) - S^{(i)}_{m,\triangle_n}(t)\|_{L_p[a,b]} \leq \frac{5}{348}h^4\|\frac{d^4 f}{dx^4}\|.$$

Так же возникает вопрос о надежности метода регрессии. Сульдин предлагает формулу:

$$p(t) = [2\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\epsilon u} e^{-\frac{(\epsilon u)^2}{2}}dx] - 1$$

Данная величина дает надежность метода в процентах.

Так как нельзя соединить численную ошибку и надежность во едино обычной суммой по-этому будем говорить надежность погрешности.

Таким образом, погрешность метода есть:

Оценка метода наименьших квадратов $r_1(t)$ плюс погрешность аппроксимации случайного процесса $r_2(t)$. И надежность погрешности есть величина $p(t)$.

$$R(t) = r_1(t) + r_2(t)$$

надежность погрешности есть $p(t)$.

Вывод

В работе построен метод аппроксимации функций, возмущенных случайными величинами. В ходе работы было реализовано несколько программ для достижения поставленной цели. В первую очередь была написана программа для вычисления коэффициентов интерполяционного полинома Лагранжа. Для приближения случайного процесса. Так же использована программа, написанная мной в прошлом году, для нахождения коэффициентов интерполяционного кубического сплайна. А для метода регрессии реализован алгоритм метода Лагранжа. Данные программы приведены в приложении.

В настоящий момент, представлено множество алгоритмов и методов для прогнозирования случайных величин. Аппроксимации случайных величин с помощью комбинации метода регрессии и метода наименьших квадратов является удачным примером прогнозирования функций, возмущенных случайными величинами.

Литература

[1] Львовский, С. М. *Набор и вёрстка в системе LaTex* / С. М. Львовский - 3-е изд. - 2003. - 428с.

[2] Сульдин, А.В. *Интегрирование в функциональных пространствах и его применение. Диссертация на соискание ученной степени доктора физ.мат. наук* / А.В Сульдин. - Казань, 1967. - 283с.

[3] Демидович, Б.П. *Численные методы анализа. Приближение функций, дифференциальные и интегральные уравнения*/Б.П. Демидович - Рипол Классик, 2013. - 374 с.

[4] Сульдин, А.В. *Том 123. Книга 6. Метод регрессии в теории приближений*/ А.В. Сульдин - Казань, 1963. - 35 с.

[5] Квасов, Б.И. *Монотонная выпуксая интерполяция весовыми кубическими сплайнами*/ Б.И Квасов - Журнал вычислительной метематики и метаматической физики, 2013. - 11 с.

[6] Ибрагимов, И.И. *Методы интерполяции функций и некоторые их применения*/И.И. Ибрагимов - Рипол Классик, 2013. - 524 с.

Приложение

Код программ

Метод Лагранжа

Здесь приводится программа написанная на языке программирования Pascal для нахождения коэффициентов C_i в методе регрессии по Методу Лагранжа.

```
unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

type

TForm1 = class(TForm)

TabbedNotebook1: TTabbedNotebook;

GroupBox1: TGroupBox;

GroupBox2: TGroupBox;

StringGrid1: TStringGrid;
```

```
StringGrid2: TStringGrid;

GroupBox3: TGroupBox;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Edit1: TEdit;

Edit2: TEdit;

Edit3: TEdit;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;

StringGrid3: TStringGrid;

StringGrid4: TStringGrid;

StringGrid5: TStringGrid;

StringGrid6: TStringGrid;

GroupBox4: TGroupBox;

GroupBox5: TGroupBox;

GroupBox6: TGroupBox;
```

```
    GroupBox7: TGroupBox;
    GroupBox8: TGroupBox;
    GroupBox9: TGroupBox;
    GroupBox10: TGroupBox;
    StringGrid7: TStringGrid;
    StringGrid8: TStringGrid;
    StringGrid9: TStringGrid;
    GroupBox11: TGroupBox;
    GroupBox12: TGroupBox;
    GroupBox13: TGroupBox;
    StringGrid10: TStringGrid;
    StringGrid11: TStringGrid;
    StringGrid12: TStringGrid;
    GroupBox14: TGroupBox;
    StringGrid13: TStringGrid;
    Panel1: TPanel;
    Label4: TLabel;
```

```
Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

Label10: TLabel;

Label11: TLabel;

Label12: TLabel;

Label13: TLabel;

procedure FormActivate(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure M(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;
```

```
var

Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormActivate(Sender: TObject);

const

begin

ShowMessage(M) ;

ShowMessage(M1) ;

ShowMessage(M2) ;

StringGrid1.Cells[0,0]:='80000' ;

StringGrid1.Cells[0,1]:='90000' ;

StringGrid1.Cells[0,2]:='100000' ;

StringGrid1.Cells[0,3]:='110000' ;

StringGrid1.Cells[0,4]:='120000' ;

StringGrid2.Cells[0,0]:='12,56' ;

StringGrid2.Cells[0,1]:='11,56' ;
```

```
StringGrid2.Cells[0,2]:='10,46' ;

StringGrid2.Cells[0,3]:='9,70' ;

StringGrid2.Cells[0,4]:='9,20' ;

StringGrid2.Cells[1,0]:='13,90' ;

StringGrid2.Cells[1,1]:='12,78' ;

StringGrid2.Cells[1,2]:='11,66' ;

StringGrid2.Cells[1,3]:='10,00' ;

StringGrid2.Cells[1,4]:='9,75' ;

StringGrid2.Cells[2,0]:='13,34' ;

StringGrid2.Cells[2,1]:='12,78' ;

StringGrid2.Cells[2,2]:='12,22' ;

StringGrid2.Cells[2,3]:='11,54' ;

StringGrid2.Cells[2,4]:='10,61' ;

StringGrid2.Cells[3,0]:='12,90' ;

StringGrid2.Cells[3,1]:='12,16' ;

StringGrid2.Cells[3,2]:='11,41' ;

StringGrid2.Cells[3,3]:='10,53' ;
```

```
StringGrid2.Cells[3,4]:='9,50' ;

StringGrid2.Cells[4,0]:='14,59' ;

StringGrid2.Cells[4,1]:='13,30' ;

StringGrid2.Cells[4,2]:='12,01' ;

StringGrid2.Cells[4,3]:='10,78' ;

StringGrid2.Cells[4,4]:='10,05' ;

StringGrid2.Cells[5,0]:='13,46' ;

StringGrid2.Cells[5,1]:='12,71' ;

StringGrid2.Cells[5,2]:='11,95' ;

StringGrid2.Cells[5,3]:='11,62' ;

StringGrid2.Cells[5,4]:='10,10' ;

end;

procedure TForm1.BitBtn1Click(Sender: TObject);

var

i,j: Integer ;

ly1,T,C1,C2,A1,B1,sum1,sum2,sum3,sum4,sum5,sum6,znam,kvadrat,sumG,sumQ : R

a,b,r,s,e,f,ly2,ly3,Q,G : array [1..6] of real;
```

```
const

M='Вы уверены?';

begin

ShowMessage(M) ;

C1:= StrToFloat(Edit2.Text);

C2:= StrToFloat(Edit3.Text);

For i:=1 to 5 do

begin

sum1:=sum1+StrToFloat(StringGrid1.Cells[0,i-1])*

StrToFloat(StringGrid1.Cells[0,i-1]);

sum2:=sum2+StrToFloat(StringGrid1.Cells[0,i-1]);

end;

kvadrat:=sum2*sum2;

znam:=sum1-0.2*kvadrat;

For i:=1 to 6 do

For j:=1 to 5 do

begin
```

```
sum3:= sum3 + StrToFloat(StringGrid2.Cells[i-1,j-1]);

sum4:= sum4 + StrToFloat(StringGrid2.Cells[i-1,j-1])*

StrToFloat(StringGrid1.Cells[0,j-1]);

a[i]:= (0.2*sum3*sum2-sum4)/(sum1-0.2*kvadrat);

b[i]:= 0.2*(sum3+a[i]*sum2);

A1:=C2/(C2-C1);

B1:=16.02*C1/(C2-C1);

r[i]:=-0.085*A1*a[i];

s[i]:=0.085*A1*b[i]-0.085*B1;

If j=5 Then begin

sum3:=0;

sum4:=0;

e[i]:=r[i];

f[i]:=s[i];

StringGrid3.Cells[i-1,0]:=FloatToStr(RoundTo(a[i],-9));

StringGrid4.Cells[i-1,0]:=FloatToStr(b[i]);

StringGrid5.Cells[i-1,0]:=FloatToStr(RoundTo(r[i],-9));
```

```
StringGrid6.Cells[i-1,0]:=FloatToStr(RoundTo(s[i],-4));

sum5:=sum5+(f[i]/(2*e[i]));

sum6:=sum6+(1/(2*e[i]));

end

end;

ly1:=(-100000-(sum5/3))/(sum6/3);

StringGrid9.Cells[0,0]:=FloatToStr(RoundTo(ly1,-4));

For i:=1 to 6 do

begin

ly2[i]:= -(f[i]/3)-(ly1/3)-(( 320000*e[i])/3);

ly3[i]:= (f[i]/3)+(ly1/3)+((80000*e[i])/3);

Q[i]:= (-f[i]-ly1-ly2[i]+ly3[i])/(2*e[i]);

sumQ:= sumQ+ Q[i];

G[i]:= e[i]*sqr(Q[i])+f[i]*Q[i];

sumG:=sumG +G[i];

StringGrid7.Cells[i-1,0]:=FloatToStr(RoundTo(ly2[i],-4));

StringGrid8.Cells[i-1,0]:=FloatToStr(RoundTo(ly3[i],-4));
```

```
StringGrid10.Cells[i-1,0]:=FloatToStr(RoundTo(Q[i],-4));

StringGrid11.Cells[i-1,0]:=FloatToStr(RoundTo(G[i],-4));

end;

StringGrid13.Cells[0,0]:=FloatToStr(sumQ);

StringGrid12.Cells[0,0]:=FloatToStr(sumG);

end;

procedure TForm1.M(Sender: TObject; var Action: TCloseAction);

const

begin

end;

end.
```

Интерполяционный кубический сплайн

В данной пункте представлен код программы для нахождения коэффициентов интерполяционного кубического сплайна взятый с моей курсовой работы 2014 года.

```
#include "stdafx.h"

using namespace System;

#include "stdafx.h"
#include "stdio.h"
```

```
#include "string.h"
#include "math.h"

class knot{
public:
double x,f;
void Add(double arg,double func)
{
x=arg;
f=func;
}
knot(){}
};

class vector{
public:
double* x;
void Add(int m)
{
x = new double[m];
}
vector()
{
}
};

knot* KnotArray;
int n;
double** Coef;
double* b;

void SolveTriDiag(double** TDM, double* F)
{
double* alph = new double[n-1];
```

```
double* beta = new double[n-1];

int i;

alph[0] = - TDM[2][0]/TDM[1][0];
beta[0] = F[0]/TDM[1][0];

for (i=1; i<n-1; i++)
{
alph[i] = -TDM[2][i]/(TDM[1][i] + TDM[0][i]*alph[i-1]);
beta[i] = (F[i]-TDM[0][i]*beta[i-1])/(TDM[1][i] + TDM[0][i]*alph[i-1]);
}
b[n-1] = (F[n-1]-TDM[0][n-1]*beta[n-2])/(TDM[1][n-1] + TDM[0][n-1]*alph[n-

for (i=n-2; i>-1; i--)
{
 b[i] = b[i+1] * alph[i] + beta[i];
}
}

int BuildSpline()
{
    double* a = new double[n-1];
    double* c = new double[n-1];
    double* d = new double[n-1];
double* delta = new double[n-1];
double* h = new double[n-1];
double** TriDiagMatrix = new double*[3];

b = new double[n];

TriDiagMatrix[0] = new double[n];
TriDiagMatrix[1] = new double[n];
TriDiagMatrix[2] = new double[n];
```

```
double* f = new double[n];
double x3,xn;
    int i;

if (n<3)
return -1;

x3 = KnotArray[2].x - KnotArray[0].x;
xn = KnotArray[n-1].x - KnotArray[n-3].x;

for (i=0; i<n-1; i++)
{
a[i] = KnotArray[i].f;
h[i] = KnotArray[i+1].x - KnotArray[i].x;
delta[i] = (KnotArray[i+1].f - KnotArray[i].f)/h[i];
TriDiagMatrix[0][i] = i>0?h[i]:x3;
f[i] = i>0?3*(h[i]*delta[i-1] + h[i-1]*delta[i]):0;
}
TriDiagMatrix[1][0] = h[0];
TriDiagMatrix[2][0] = h[0];
for (int i=1; i<n-1;i++)
{
TriDiagMatrix[1][i] = 2*(h[i] + h[i-1]);
TriDiagMatrix[2][i] = h[i];
}
TriDiagMatrix[1][n-1] = h[n-2];
TriDiagMatrix[2][n-1] = xn;
TriDiagMatrix[0][n-1] = h[n-2];



i = n-1;
f[0] = ((h[0]+2*x3)*h[1]*delta[0] + powf(h[0],2)*delta[1])/x3;
f[n-1]=(powf(h[i-1],2)*delta[i-2]+(2*xn+h[i-1])*h[i-2]*delta[i-1])/xn;
```

```
SolveTriDiag(TriDiagMatrix,f);

Coef = new double*[4];
Coef[0] = new double[n-1];
Coef[1] = new double[n-1];
Coef[2] = new double[n-1];
Coef[3] = new double[n-1];
int j;
for (j=0; j<n-1; j++)
{
d[j] = (b[j+1]+b[j]-2*delta[j])/(h[j]*h[j]);
c[j] = 2*(delta[j]-b[j])/h[j]-(b[j+1]-delta[j])/h[j];

Coef[j][0] = a[j];
Coef[j][1] = b[j];
Coef[j][2] = c[j];
Coef[j][3] = d[j];
}
}

double Interpolate(double x)
{

int i=0;

while (KnotArray[i].x < x)
i++;
i--;
return Coef[i][0] + Coef[i][1]*(x-KnotArray[i].x) + Coef[i][2]*powf((x-Kno

}
```

```
int Load_Data()
{
printf("Input filename with data\n");
char FileName[20];
int i=0;
double ii;
FILE *File;
scanf("%s",&FileName);
if (!fopen_s(&File,FileName,"r"))
{
printf("file is opened\n");
}
else
{
printf("%s: file doesn't exist\n",FileName);
return -1;
}
double x,f;
fscanf(File,"%d",&n);
KnotArray = new knot[n];
while (!feof(File))
{
fscanf(File,"%lf%lf",&x,&f);
KnotArray[i].Add(x,f);
i++;
if (i==n+1)
return -1;
}
fclose(File);
return 1;
}

int main(array<System::String ^> ^args)
{
```

```
double x=0;
if(Load_Data()!=-1)
{
BuildSpline();
scanf("%lf",&x);
printf("x:  %lf\nans:   %lf\n",x,Interpolate(x));
scanf("%lf",&x);

}
return 0;
}
```

Интерполяционный полином Лагранжа

В данной пункте представлен код программы для нахождения интерполяционного полинома Лагранжа для построения функции случайного процесса.

```
double lagrange(double _x);
double func(double _x);
int main( void )
{
  setlocale(LC_ALL, "Russian");
  int a=-2, b=3, m=4;
  double X[9], Nevyazka[9];


    for (short j=0; j<=9; j++)
    {
        X[j]=a+j*(b-a)/9;
        Nevyazka[j]=fabs(lagrange(X[j])-func(X[j]));
        cout<<"f(X"<<j<<")="<<func(X[j])<<";\t"<<"L(X"<<j<<")="<<lagrange(
    }

        getch();
    return (0);
```

```
}



double lagrange(double _x)
{
        double L = 0, P = 1, x[5], y[5];
        int a=-2, b=4;
        short n=4;
        //double x[5] = {-2, -0.75, 0.5, 1.75, 3};
        //double y[5] = {-1.64, 1.77, -1.36, 0.11, 11.01};
        for (short i = 0; i < n; i++)
        {
            x[i]=a+(i*(b-a))/n;
            y[i]=4*x[i]-7*sin(x[i]);
        }

        for (short i = 0; i < n; i++)
        {
            for (short j = 0; j < n; j++)
                {
                 if (j - i)
                 P *= (_x - x[j])/ (x[i] - x[j]);
                 }
             L += P * y[i];
        }

        return L;
}

double func(double _x)
{
    return 4*_x-7*sin(_x);
}
```