# Введение в Численные Методы
# Аналитический отчёт по практическому заданию

Выполнила студентка 208 группы ВМК МГУ
Мазур Анастасия Вадимовна

# Математическая постановка задачи

Функция $f(x)$ задана таблично на отрезке [0, a] в точках $x_i$, $x_i = ih$,
$i = 0, 1, ..., n, h = a/n$

- Построить интерполяционный многочлен по точкам $x$.
- Приблизить функцию по методу наименьших квадратов полиномом заданной степени $n$, $n < 9$. Оценить погрешность.
- Результаты сравнить.

Отрезок $[0, 2]$, $n = 4$
Таблица значений функции в точках:

| $i$ | $x$ | $f(x)$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0.2 | 0.006732 |
| 2 | 0.4 | 0.058195 |
| 3 | 0.6 | 0.030482 |
| 4 | 0.8 | 0.387483 |
| 5 | 1 | 0.958924 |
| 6 | 1.2 | 0.48283 |
| 7 | 1.4 | 1.802771 |
| 8 | 1.6 | 4.052411 |
| 9 | 1.8 | 2.403475 |
| 10 | 2 | 4.352169 |

# Используемые алгоритмы и формулы

## Построение интерполяционного многочлена в форме Лагранжа

Чтобы интерполировать функцию построим полином в форме Лагранжа.
Искомый полином $P_n(x)$ будет иметь следующий вид:

$$P_n(x) = \sum_{i=0}^{n} f(x_i) Q_{n,i}(x),$$

где $Q_{n,i}(x)$ - полиномы степени $n$, "ориентированные"на точки $x_i$
в том смысле, что

$$Q_{n,i}(x) = \begin{cases} 0, x = x_j, \ \forall j \neq i \\ 1, x = x_i \end{cases}$$

Полиномы имееют вид:

$$Q_{n,i}(x) = \prod_{\substack{j=0 \\ j \neq i}}^{j=n} \frac{(x - x_j)}{(x_i - x_j)}$$

или в нашем случае, когда $x_i = ih$, то есть известны значения в точках, расстояние между которыми фиксировано, то выражение можно упростить до следующей записи:

$$Q_{n,i}(x) = h^{-n} \cdot \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - jh)}{(i - j)}$$

Учитывая, что полином в форме Лагранжа $P_n(x)$ представляет собой линейную комбинацию алгебраических уравнений $f(x_i)Q_{n,i}(x), Q_{n,i}(x)$ - полиномы степени n, можно утверждать, что $P_n(x)$ будет иметь степень не более $n$.

Данные формулы будут далее использоваться в программной реализации.

## Приближение функции методом наименьших квадратов

В методе наименьших квадратов аппроксимирующая функция $y(x)$ ищется в виде следующей суммы:

$$F(x) = \sum_{k=0}^{m} a_k \varphi_k(x), \quad m < n$$

В каждой точке сетки $x_i$ можно подсчитать погрешность:

$$\delta_i = y_i - F(x_i) = y_i - \sum_{k=0}^{m} a_k \varphi_k(x_i), \quad i = 0, 1, 2, \ldots, n$$

Сумма квадратов этих величин называется суммарной квадратичной погрешностью

$$J = \sum_{i=0}^{n} \delta_i^n = \sum_{i=0}^{n} \left( y_i - \sum_{k=0}^{m} a_k \varphi_k(x_i) \right)^2$$

Главной задачей является подобрать такие коэффициенты $a_k$, чтобы суммарная квадратичная погрешность была минимальной.

Таким образом, построение наилучшего приближения сводится к классической задаче математического анализа об экстремуме функции нескольких переменных. Необходимым условием экстремума является равенство нулю в экстремальной точке всех первых частных производных функции.

$$\frac{\partial J}{\partial a_e} = -2 \sum_{i=0}^{n} \left( y_i \sum_{k=0}^{m} a_k \varphi_k(x_i) \right) \varphi_L(y_i) = 0, \quad l = 0, 1, \ldots, m.$$

Оставим члены, содержащие $a_k$, слева и поменяем в них порядок суммирования по индексам $i$ и $k$. Члены, содержащие $y_i$, перенесем направо. В результате уравнения примут вид:

$$\sum_{k=0}^{m} \gamma_{lk} a_k = b_l, \quad l = 0, 1, \ldots, m,$$

где

$$\gamma_{lk} = \sum_{i=0}^{n} \varphi_l(x_i) \varphi_k(x_i)$$

$$b_l = \sum_{i=0}^{n} \varphi_l(x_i) y_i$$

Мы получили систему линейных алгебраических уравнений, в которой роль неизвестных играют искомые коэффициенты разложения $a_0, a_1, \ldots, a_m$. Используя найденные коэффициенты разложения, мы сможем построить наилучшее приближении сеточной функции по методу наименьших квадратов.

Данные формулы будут далее использоваться в программной реализации.

# Цифровое представление результатов

Для того, чтобы произвести необходимые вычисления и проанализировать результаты, мною была написана программа.

В данном отчёте представлены лишь основные функции и результат работы программы на входных данных из условия, однако при желании с полным кодом можно ознакомиться отдельно и произвести тестирование на других входных данных (см. архив).

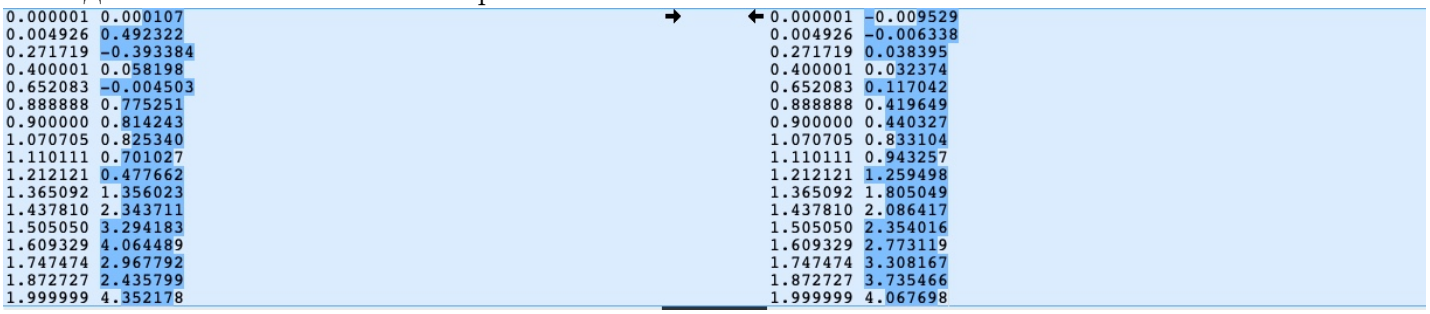
**Принцип работы программы**

Известные точки записываются записываются в файл *dots.txt*, каждая точка на отдельной строке, абсцисса и ордината отделены пробельным символом. Точки, значения которых хотим узнать, записываются в файл *input.txt*.

После запуска программы в файлах *output_Lagrange.txt* и *output_LeastSquares.txt* будут записаны прогнозируемые значения, посчитанные по этим методам соотвественно.

Такой подход позволяет изменять входные данные, не изменяя программного кода.

Затем файлы можно сравнить с помощью следующей команды терминала: *meld*, она наиболее наглядно показывает отличия в файлах.



| | |
|---|---|
| 0.000001 0.000107 | 0.000001 -0.009529 |
| 0.004926 0.492322 | 0.004926 -0.006338 |
| 0.271719 -0.393384 | 0.271719 0.038395 |
| 0.400001 0.058198 | 0.400001 0.032374 |
| 0.652083 -0.004503 | 0.652083 0.117042 |
| 0.888888 0.775251 | 0.888888 0.419649 |
| 0.900000 0.814243 | 0.900000 0.440327 |
| 1.070705 0.825340 | 1.070705 0.833104 |
| 1.110111 0.701027 | 1.110111 0.943257 |
| 1.212121 0.477662 | 1.212121 1.259498 |
| 1.365092 1.356023 | 1.365092 1.805049 |
| 1.437810 2.343711 | 1.437810 2.086417 |
| 1.505050 3.294183 | 1.505050 2.354016 |
| 1.609329 4.064489 | 1.609329 2.773119 |
| 1.747474 2.967792 | 1.747474 3.308167 |
| 1.872727 2.435799 | 1.872727 3.735466 |
| 1.999999 4.352178 | 1.999999 4.067698 |

Сравнение предсказанных значений в произвольных точках отрезка на основе вычислений по разным методам с помощью *meld*
Слева - интерполяция полиномом Лагранжа
Справа - аппроксимация методом наименьших квадратов

**Функции**

```
long double
y_Lagrange(long double *x, long double *y, int num, long double x0) {
    long double L;
    int i, j;
    long double P;
    L = 0;
    for (i = 0; i < num; i++) {
        P = 1;
        for (j = 0; j < num; j++) {
            if (i != j) {
                P *= (x0 - x[j]) / (x[i] - x[j]);
            }
        }
        L += y[i] * P;
    }
    return L;
}

long double
y_LeastSquares(long double *x, long double* y, int num, long double x0) {
    int i, j, k;
    for (i = 0; i < n + 1; i++){
        a[i]=0;
        b[i]=0;
            for ( j = 0; j < n + 1; j++){
                sum[i][j] = 0;
```

```
            }
        }
    }
    for (i = 0; i < n + 1; i++) {
        for (j = 0; j < n + 1; j++) {
            sum[i][j] = 0;
            for (k = 0; k < num; k++) {
                sum[i][j] += pow(x[k], i + j);
            }
        }
    }
    for (i = 0; i < n + 1; i++) {
        for (k = 0; k < num; k++) {
            b[i] += pow(x[k], i) * y[k];
        }
    }
    long double temp = 0;
    for (i = 0; i < n + 1; i++) {
        if (sum[i][i] == 0) {
            for (j = 0; j < n + 1; j++) {
                if (j == i) {
                    continue;
                }
                if (sum[j][i] != 0 && sum[i][j] != 0) {
                    for (k = 0; k < n + 1; k++) {
                        temp = sum[j][k];
                        sum[j][k] = sum[i][k];
                        sum[i][k] = temp;
                    }
                    temp = b[j];
                    b[j] = b[i];
                    b[i] = temp;
                    break;
                }
            }
        }
    }
    for (k = 0; k < n + 1; k++) {
        for (i = k + 1; i < n + 1; i++) {
            if (sum[k][k] == 0) {
                printf("\n Solution doesn't exist! \n");
                return 0;
            }
            long double M = sum[i][k] / sum[k][k];
            for (j = k; j < n + 1; j++) {
                sum[i][j] -= M * sum[k][j];
        }
        b[i] -= M * b[k];
    }
}
    for (i = n; i >= 0; i--) {
        long double s = 0;
        for (j = i; j < n + 1; j++){
            s += sum[i][j] * a[j];
        }
        a[i] = (b[i] - s) / sum[i][i];
    }
    long double result = 0;
    for (i = 0; i < n + 1; i++){
        result += a[i] * pow(x0, i);
    }
    return result;
}
```

# Графическое представление результатов


Синие точки - точки, известные из условия
Оранжевая кривая - кривая интерполирующего многочлена в
форме Лагранжа
Зелёная кривая - кривая, построенная по методу наименьших квадратов

# Анализ результатов

Ключевым отличием двух рассматриваемых методов является то, что полином Лагранжа интерполирует функцию $f(x)$, точки которой нам известны, а метод наименьших квадратов эту функцию аппроксимирует.

То есть для полинома Лагранжа важно, чтобы полученная интерполяционная функция строго проходила через известные узлы, однако вне известных точек функция сильно "скачет". Такой разброс значений приводит к тому, что полученная интерполяционная кривая плохо характеризует поведение исходной функции $f(x)$ в целом. Это усложняет прогнозирование значений и дальнейшую работу с функцией, затрудняет визуальное восприятие графика.

Метод наименьших квадратов, напротив, на выходе даёт нам такую функцию, которая лишь приближает $f(x)$, то есть может совпадать с рассматриваемой функцией на очень маленьком наборе точек, но зато полученная кривая довольно удобна и наглядна.

# Дополнительно: применимость методов

В данный момент метод наименьших квадратов активно применяется в анализе данных. Этот метод помогает аппроксимировать различные экспериментальные данные в разных областях, например, в экономике. Это очень сильный инструмент, потому что позволяет сделать довольное

точное прогнозирование новых значений.

Что касается интерполяции с помощью полинома Лагранжа, то такой метод тоже пользуется популярностью в современном мире. Например, метод был использован для вычислений при разработке двигателей. Построенный полином помог сократить количество реальных тестирований.

# Источники и ресурсы

Вводные лекции по численным методам (Д.П. Костомаров, А.П. Фаворский)
Для построения графика использовался ресурс *www.geogebra.com*