

Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Локтионова Оксана Геннадьевна
Должность: проректор по учебной работе
Дата подписания: 31.12.2020 13:36:44
Уникальный программный ключ:
0b817ca911e6668abb13a5d426d39e5f1c11eabbf73e943df4a48314aa36d0b9


АОБЧ

МИНОБРАЗОВАНИЯ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра биомедицинской инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
О.Г. Локтионова
«Юго-Западный государственный университет»
(ЮЗГУ)
2017 г.



ОСНОВЫ АВТОМАТИЗИРОВАННОЙ ОБРАБОТКИ РЕЗУЛЬТАТОВ МЕДИКО-БИОЛОГИЧЕСКИХ ИССЛЕДОВАНИЙ

Методические указания к лабораторным работам по дисциплинам
«Автоматизация обработки биомедицинской информации»,
«Стандартные программные средства в имитационном
моделировании биотехнических систем» и «Автоматизация
обработки экспериментальных данных»

Курск 2017

УДК 004.93:61

Составители: О.В. Шаталова, К.Д.А. Кассим, С.А. Филист.

Рецензент

Доктор технических наук, профессор Р.А. Томакова

Основы автоматизированной обработки результатов медико-биологических исследований: методические указания к лабораторным работам / Юго-Зап. гос. ун-т; сост.: О.В. Шаталова, К.Д.А. Кассим, С.А. Филист. Курск, 2017. 130 с.

Предназначено для студентов направления подготовки 12.03.04 «Биотехнические системы и технологии» и по специальности 30.05.03 «Медицинская кибернетика» по дисциплинам «Автоматизация обработки биомедицинской информации», «Стандартные программные средства в имитационном моделировании биотехнических систем» и «Автоматизация обработки экспериментальных данных». Может быть использована аспирантами, обучающимися по направлениям 05.11.13 – Системный анализ, управление и обработка информации и 05.11.17 – Приборы, системы и изделия медицинского назначения.

Текст печатается в авторской редакции

Подписано в печать 5.05.17. Формат 60×84 1/16. Бумага офсетная.
Усл. печ. л. 7,56. Уч.-изд. л. 6,84. Тираж 100 экз. Заказ 889.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Лабораторная работа №1 «Интерполяция функций»

Цель работы: изучение методов решения задачи интерполяции; приобретение навыков программирования методов интерполяции; приобретение навыков использования стандартных средств системы MATLAB для проведения интерполирования.

Теоретические сведения

Постановка задачи интерполяции

Пусть на отрезке $a \leq x \leq b$ задана сетка $\bar{\omega} = \{x_0 = a < x_1 < x_2 < \dots < x_n = b\}$ и в ее узлах x_i заданы значения функции $y(x)$, равные

$$y(x_0) = y_0, \dots, y(x_i) = y_i, \dots, y(x_n) = y_n.$$

Требуется построить интерполянту – функцию $f(x, c)$, совпадающую с функцией $y(x)$ в узлах сетки

$$f(x_i, c) = y_i, \quad i = 0, 1, \dots, n, \quad (1.1)$$

где $c = \{c_0, \dots, c_n\}$ – некоторые неизвестные параметры. Основная цель решения этой задачи состоит в том, чтобы иметь возможность вычисления значений функции $y(x)$ для значений x , не содержащихся в таблице данных. Основным вопросом интерполяции является выбор интерполянты $f(x, c)$ и оценка погрешности интерполяции, т.е. величины $y(x) - f(x, c)$. Математически он заключается в определении неизвестных параметров c при выборе определенного вида функциональной зависимости.

Рассмотрим линейную зависимость функции $f(x, c)$ от параметров c , т.е. будем считать, что она представима в виде обобщенного многочлена

$$f(x, c) = \sum_{k=0}^n c_k \phi_k(x).$$

Интерполяционный многочлен Лагранжа

Будем строить многочлен n -й степени, который исторически обозначается $L_n(x)$ и называется многочленом Лагранжа, в виде линейной комбинации специальных (базисных) многочленов n -й степени $l_i(x)$ при $i=0,1,\dots,n$:

$$L_n(x) = \sum_{i=0}^n c_i l_i(x). \quad (1.3)$$

Для того чтобы такой многочлен был интерполяционным для функции $y(x)$, потребуем выполнения условий интерполяции $L_n(x) = y(x_i) = y_i$. Эти условия будут выполнены, если $c_i = y_i$, а базисные многочлены $l_i(x)$ удовлетворяют условиям

$$l_i(x_j) = \begin{cases} 0, & \text{если } j \neq i \\ 1, & \text{если } j = i \end{cases} \quad \forall i, j \in \{0,1,\dots,n\}. \quad (1.4)$$

Положим

$$l_i(x) = \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}, \quad i = 0,1,2,\dots,n. \quad (1.5)$$

Нетрудно проверить, что многочлены $l_i(x)$ удовлетворяют условиям (1.4). подставляя (1.5) в (1.3), получаем

$$L_n(x) = \sum_{i=0}^n \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)} y_i \quad (1.6)$$

или в более компактной форме

$$L_n(x) = \sum_{i=0}^n y_i \prod_{j \neq i} \frac{(x - x_j)}{(x_i - x_j)}.$$

Погрешность интерполяции степенными функциями

Рассмотрим в качестве погрешности величину остаточного члена интерполяции в произвольной точке $x \in [a, b]$:

$$R_n(x) = y(x) - L_n(x).$$

Оценку величины $R_n(x)$ будем проводить в предположении, что функция $f(x)$ имеет $(n+1)$ непрерывную производную на отрезке $[a, b]$.

Введем многочлен $(n+1)$ -й степени $\Pi_{n+1}(x)$, определенный через узлы x_0, x_1, \dots, x_n :

$$\Pi_{n+1}(x) = \prod_{i=0}^n (x - x_i) = (x - x_0)(x - x_1) \dots (x - x_n).$$

Тогда абсолютную погрешность интерполяционной формулы Лагранжа (1.6) в произвольной точке $x' \in [a, b]$ можно оценить с помощью неравенства

$$|R_n(x)| \leq \frac{M}{(n+1)!} |\Pi_{n+1}(x)|, \quad (1.7)$$

где $M = \max_{x \in [a, b]} |f^{(n+1)}(x)|$.

Рассмотрим как ведет себя погрешность интерполяции при увеличении числа узлов. Выражение для погрешности (1.7) состоит из трех различных частей; факториал и произведение разностей с увеличением n уменьшают ошибку, а порядок производной при этом растет. Для многих функций величина M_{n+1} увеличивается быстрее, чем $(n+1)!$. В результате полиномиальные интерполянты редко сходятся к обычной непрерывной функции при $n \rightarrow \infty$. Наблюдаемый на практике эффект проявляется в том, что интерполяционный полином высокой степени может вести себя «плохо» в точках, отличных от узлов интерполяции (x_i, y_i) , $i = \overline{0, n}$. Поэтому на практике часто используют интерполянты степени не выше 5-6.

Примером может служить функция Рунге вида $r(x) = 1/(1 + 25x^2)$. С увеличением порядка интерполирующего полинома при равномерном распределении узлов интерполяции на интервале $[-1, 1]$ происходит ухудшение качества приближения на краях интервала. Это объясняется тем, что производные $r(x)$, которые фигурируют в выражении для погрешности интерполяции, быстро растут с увеличением числа n . Таким образом, точность приближения зависит не только от числа узлов интерполяции (т.е. порядка интерполирующего полинома), но и от их расположения на интервале $[a, b]$.

Задача о наилучшем выборе узлов интерполирования была решена Чебышевым. Наилучшие узлы интерполирования выбираются равными корнями «полинома, наименее отклоняющегося от нуля» на отрезке интерполирования. Полином, наименее отклоняющийся от нуля на отрезке $[-1, 1]$, был найден Чебышевым и назван его именем. Полином Чебышева определяется следующим выражением:

$$T_n(t) = \cos(n \arccost), \quad (1.8)$$

где $n=0, 1, 2, \dots$. Эта тригонометрическая функция является многочленом при любом n . При $n=0$ и $n=1$ непосредственно из (1.8) получаем $T_0(t) = 1, T_1(t) = t$. Далее, обозначая $\alpha = \cos t$, имеем $T_1(t) = \cos \alpha, T_n(t) = \cos n\alpha$. Так как по правилу сложения косинусов $\cos(n+1)\alpha + \cos(n-1)\alpha = 2\cos \alpha \cos n\alpha$, справедливо

$$T_{n+1}(t) + T_{n-1}(t) = 2T_1(t)T_n(t). \quad (1.9)$$

Из (1.9) следует, что последовательность функций $T_n(t)$, определяемая рекуррентно, представляет собой многочлен степени n : $T_2(t) = 2t^2 - 1, T_3(t) = 4t^3 - 3t, T_4(t) = 8t^4 - 8t^2 + 1$ и т.д.

Из всех многочленов степени n со старшим коэффициентом 1 нормированный многочлен Чебышева $T'_n(t) = 2^{-n+1}T_n(t)$ наименее уклоняется от нуля на отрезке $[-1, 1]$. Последнее фактически означает, что среди всех многочленов степени n вида

$$P_n(t) = t^n + a_1 t^{n-1} + a_2 t^{n-2} + \dots + a_n$$

именно нормированный многочлен $T'_n(t)$ минимизирует максимальное расстояние от графика многочлена до оси абсцисс при $t \in [-1, 1]$. Таким образом, максимальная погрешность интерполирования будет минимальной, если в качестве многочлена $P_{n+1}(t)$ взять нормированный многочлен Чебышева $T_n(t)$, так как именно он наименее уклоняется от нуля на отрезке $[-1, 1]$. В этом случае точки t_0, t_1, \dots, t_n будут корнями многочлена Чебышева $T_{n+1}(t)$ или $T'_{n+1}(t)$, которые имеют вид

$$t_k = \cos \frac{2k+1}{2n} \pi, \text{ где } k = 0, 1, \dots, n-1.$$

Обычно эти узлы называются чебышевскими узлами интерполяции.

На интервале $[a, b]$ эти корни можно представить по формуле Чебышева

$$x_{i+1} = \frac{a+b}{2} + \frac{b-a}{2} \cos \left(\frac{(2i+1)\pi}{2n+2} \right), i = \overline{0, n}.$$

Интерполяционная функция с использованием именно этих точек в качестве узлов дает наилучшее приближение к интерполируемой функции.

Средства MATLAB для проведения интерполяции

Для интерполирования функций в MATLAB можно использовать операторы `polyfit` и `polyval`.

Оператор `polyfit(x,y,n)` находит массив коэффициентов a длиной $(n+1)$ полинома степени n по массивам длиной m узлов x и значений функции y в узлах $x, m \geq n+1$. Этот полином аппроксимирует функцию $y(x)$ по методу наименьших квадратов. Предполагается, что полином задается в виде

$$y(x) = a_1 x^n + a_2 x^{n-1} + \dots + a_n x + 1.$$

Если $m=n+1$, то программа возвращает коэффициенты интерполяционного полинома.

Оператор `polyval(a,x)` возвращает значение полинома в точке x , коэффициенты которого определены в векторе a .

Кроме этого, существуют операторы, позволяющие проводить интерполяцию одно- и многомерных данных:

- `interpft(y,n,dim)` - аппроксимация периодической функции на основе быстрого преобразования Фурье (y - одномерный массив значений функции; n - число узлов в массиве значений);

- `spline(x,y,z)` - интерполяция $y=y(x)$ кубическим сплайном и вывод

соответствующих значений в точках z . Для получения большей информации используется конструкция `pp=spline(x,y)`, здесь командой `v=ppval(pp,z)` можно найти значения в точках z , а командой `[xs,Coef,m,L]=unmkpp(pp)` получить данные о векторе разбиений аргумента x_s , коэффициентах $Coef$, $m=length(x_s)$, $L=length(Coef)/m$;

- `interp1(x,y,z)`, `interp1(x,y,z,'method')` - одномерная табличная интерполяция (если y - двумерный массив, интерполяция ведется по каждому столбцу; значения z должны входить в диапазон значений x). Можно указать метод интерполяции - кусочно-линейной (`linear` по умолчанию), ступенчатой (`nearest`), кубической (`cubic`), кубическими сплайнами (`spline`). Функция `interp1q(x,y,z)` реализует быструю линейную интерполяцию на неравномерной сетке;

- `interp2(x1,x2,y,z1,z2)`, `interp2(x1,x2,y,z1,z2,'method')` - двумерная табличная интерполяция $y=y(x_1,x_2)$, аргументы должны меняться монотонно и быть заданы в формате функции `meshgrid`;

- `interp3(x1,x2,x3,y,z1,z2,z3)`, `interp3(...,'method')` - трехмерная табличная интерполяция $y=y(x_1,x_2,x_3)$;

- `interpn(x1,x2,...,y,z1,z2,...)`, `interp3(...,'method')` - многомерная табличная интерполяция $y=y(x_1,x_2,...)$;

- `griddata(x1,x2,y,z1,z2)`, `griddata(x1,x2,y,z1,z2,'method')` - двумерная табличная интерполяция на неравномерной сетке.

Для реализации программы полезными являются следующие операторы:

- `sum(x)` и `prod(x)` - суммирование и произведение элементов вектора (для двумерного массива выполняется поиск сумм и произведений по столбцам);

- $\text{diff}(x)$, $\text{diff}(x,n)$, $\text{diff}(x,n,\text{dim})$ - вычисление конечных разностей (первых, n -го порядка или по указанному измерению); если x - массив, берутся разности между столбцами.

Порядок выполнения работы

Написать соответствующие m -файлы сценарии и m -файлы функции для реализации следующих задач.

1. Провести интерполяцию функции Рунге на отрезке $[-1, 1]$ по формуле Лагранжа для $n=11$ при равномерном распределении узлов интерполяции.
2. Провести интерполяцию функции Рунге на отрезке $[-1, 1]$ по формуле Лагранжа для $n=11$ для чебышевских узлов.
3. Построить графики функции Рунге и ее интерполянт не менее чем в 100 узлах. Сравнить результаты.
4. Выбрать функцию согласно номеру варианта и провести ее интерполяцию по формуле Лагранжа при равномерном распределении узлов на заданном интервале для $n=11$ и $n=6$.
5. Провести интерполяцию по тем же узлам, используя стандартные функции MATLAB.
6. Построить графики исходной функции и интерполянт не менее чем в 100 узлах. Сравнить результаты.
7. Вычислить и построить графики функций ошибок интерполяции.

Содержание отчета

1. Цель работы, задание;
2. Описание метода решения, краткие сведения из теории (формулы, алгоритм и т.п.);
3. Программу (распечатку), ее описание;
4. Сравнение результатов расчета;
5. Краткие выводы.

Задание

Номер варианта	$y(x)$	a	b
1	$\sqrt{(x-2)^3(5-x)}$	2	5
2	$(1+x)(x^2+3)^{-1}$	-9	9
3	$10e^{-x}(x^3-2x+1)$	-2	2
4	$(1+x^2)\sin 2x$	-1	5
5	$\exp(-0,5x \cos x)$	-5	3
6	$\exp(2,5 \sin x) \cos x$	1	3
7	$x^2 \sin(2x-3)$	0	4
8	$x(x^2+3)^{-1}$	-5	5
9	$(1+x^2)(1+x^3)^{-1}$	0	4
10	$(x-3)\cos^2 x$	-3	3
11	$(1-x)\cos x$	0	4
12	$\sqrt{x} - \cos(1,5x)$	0	8
13	$\sin 2x - 0,2x^2$	0	6
14	$(x^2+1)^{-1} \sin(x+1)$	-3	2
15	$0,5x^2 + 8x^{-1} + 8$	-4	-1

Контрольные вопросы

1. Сформулируйте постановку задачи интерполяции.
2. Сформулируйте постановку задачи полиномиальной интерполяции со степенным базисом.
3. Выведите интерполяционную формулу Лагранжа. Приведите примеры для интерполяции по 2 и 3 точкам.
4. Приведите оценку погрешности интерполяционной формулы Лагранжа.
5. Дайте определение полиномов Чебышева. Выведите рекуррентное соотношение для их вычисления.
6. Как осуществляется интерполяция по чебышевским узлам?

7. Дайте определение интерполяционного полинома Ньютона. Укажите условия практического применения многочленов Ньютона и Лагранжа.

8. Дайте определение разделенных и конечных разностей. Укажите их свойства.

9. Постройте таблицу разделенных разностей для пяти точек.

10. Поставьте задачу кусочно-полиномиальной интерполяции.

Лабораторная работа №2 «Аппроксимация данных методом наименьших квадратов»

Цель работы: изучение метода наименьших квадратов для аппроксимации данных; приобретение навыков программирования метода наименьших квадратов для аппроксимации экспериментальных данных; приобретение навыков использования стандартных средств системы MATLAB для задачи аппроксимации.

Теоретические сведения

Пусть имеются результаты измерений двух величин x и y и предполагается, что они связаны линейной зависимостью

$$y = b_0 + b_1x + \varepsilon. \quad (2.1)$$

При этом ошибка измерения является некоторой случайной величиной, а ее среднее значение равно нулю. Так как результаты обычно не ложатся на прямую, то необходимо построить наилучшую линейную функцию, проходящую наиболее близко к каждому результату измерения, но возможно не совпадающую с ними. Это типичная задача обработки результатов эксперимента.

Для нахождения неизвестных коэффициентов модели (2.1) используют метод наименьших квадратов, а именно минимизируют функцию двух переменных вида

$$Q(b_0, b_1) = \sum_{i=1}^n [y_i - (b_0 + b_1x_i)]^2.$$

Запишем необходимое условие минимума дифференцируемой функции:

$$\frac{\partial Q}{\partial b_0} = 0, \quad \frac{\partial Q}{\partial b_1} = 0. \quad (2.2)$$

В результате решения системы (2.2) из двух линейных уравнений получим

$$b_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}, \quad b_0 = \bar{y} - b_1 \bar{x}, \quad (2.3)$$

где средние значения \bar{x} и \bar{y} определяются по формулам

$$\bar{x} = \frac{1}{n} \sum x_i, \quad \bar{y} = \frac{1}{n} \sum y_i.$$

Формулы (2.3) позволяют построить график прямой, называемой прямой регрессии y на x , которая дает приближенное представление о зависимости между переменными y и x . В дополнение к ней обычно строят и регрессию x на y , модель которой аналогична (2.1), только в качестве независимой переменной используют y . Таким образом, получают две пересекающиеся в точке (\bar{x}, \bar{y}) прямые, при этом большинство результатов измерений лежит именно между ними. Во многих случаях удается линеаризовать исходную зависимость и рассматривать ее относительно некоторых новых переменных, которые образуют линейную связь. Например, для функции $y = ax^b + c$ берут новые переменные $Y = \lg(y-c)$ $X = \lg x$ и получают зависимость вида $Y = bX + \lg a$.

В общем случае, если регрессия y на x отличается от линейной, рассматривают линейную по параметрам регрессионную модель вида

$$y = b_0 + b_1 a_1(x) + \dots + b_{k-1} a_{k-1}(x), \quad (2.4)$$

где $a_1(x), \dots, a_{k-1}(x)$ - известные функции; b_0, b_1, \dots, b_{k-1} - неизвестные параметры.

Пусть имеется n наблюдений (x_i, y_i) , которые являются результатом реализации случайного вектора (X, Y) . Подставляя в (2.4) выборочные точки x_i , получаем систему:

$$y_i = b_0 + b_1 a_1(x_i) + \dots + b_{k-1} a_{k-1}(x_i) + \varepsilon_i, \quad i = 1, 2, \dots, n. \quad (2.5)$$

Здесь ε_i - случайные независимые друг относительно друга и распределенные по нормальному закону ошибки наблюдений. Необходимо отметить, что система уравнений (2.5) относительно неизвестных b_j является типичной пере- или недоопределенной задачей (в зависимости от соотношения n и k) и должна решаться методом наименьших квадратов.

По методу наименьших квадратов в качестве оценок b_j принимают значения \tilde{b}_j , дающие минимум функции

$$Q(b_0, b_1, \dots, b_{k-1}) = \sum_{i=1}^n (y_i - (b_0 + b_1 a_1(x_i) + \dots + b_{k-1} a_{k-1}(x_i)))^2.$$

Из необходимых условий минимума следует, что оценки \tilde{b}_j являются решениями алгебраической системы k уравнений с k неизвестными. Данную систему уравнений часто называют нормальной системой. В матричных обозначениях эту систему уравнений можно записать в виде

$$(A^T A) \tilde{b} = A^T y, \quad (2.6)$$

где $y^T = (y_1, y_2, \dots, y_n)$ - n -вектор наблюдений; $\tilde{b}^T = (b_0, b_1, \dots, b_{k-1})$ - k -вектор оценок параметров:

$$A = \begin{bmatrix} 1 & a_1(x_1) & \dots & a_{k-1}(x_1) \\ 1 & a_1(x_2) & \dots & a_{k-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & a_1(x_n) & \dots & a_{k-1}(x_n) \end{bmatrix} - \text{регрессионная } n \times k - \text{ матрица}$$

При условии, что матрица $(A^T A)$ невырожденная, решение (2.6) можно записать в виде

$$\tilde{b} = (A^T A)^{-1} A^T y.$$

Часто в качестве функций $a_k(x)$ принимают степенные функции, т.е. $a_k(x) = x^k$. В этом случае регрессионная матрица имеет вид

$$A = \begin{bmatrix} 1 & x_1 & \cdot & x_1^{k-1} \\ 1 & x_2 & \cdot & x_2^{k-1} \\ \cdot & \cdot & \cdot & \cdot \\ 1 & x_n & \cdot & x_n^{k-1} \end{bmatrix}$$

Уравнение для нахождения неизвестного вектора параметров b будет иметь вид

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 & \cdot & \sum x_i^{k-1} \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \cdot & \sum x_i^k \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \cdot & \sum x_i^{k+1} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \sum x_i^{k-1} & \sum x_i^k & \sum x_i^{k+1} & \cdot & \sum x_i^{2k-2} \end{bmatrix} \begin{bmatrix} \tilde{b}_0 \\ \tilde{b}_1 \\ \tilde{b}_2 \\ \cdot \\ \tilde{b}_{k-1} \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \\ \sum y_i x_i^2 \\ \cdot \\ \sum y_i x_i^{k-1} \end{bmatrix} \quad (2.7)$$

При больших значениях k система (2.7) плохо обусловлена и для ее решения используются специальные методы. В обычной практике статистической обработке данных k невелико ($k \leq 5$) и тогда эту систему можно решить либо методом Гаусса, либо подходящим итерационным методом.

Средства MATLAB для аппроксимации функций

Для предварительной статистической обработки данных используются операторы:

- $\max(A)$, $\min(A)$ - поиск экстремальных элементов по столбцам массива A ;
- $\max(A,B)$, $\min(A,B)$ - формирование массива с элементами, равными экстремальным из соответствующих элементов массивов;
- $\max(A,[],dim)$, $\min(A,[],dim)$ - вектор экстремальных элементов по измерению dim ;
- $[C,I]=\max(...)$, $[C,I]=\min(...)$ - дополнительно выводится строка индексов экстремальных элементов;
- $\text{median}(x)$, $\text{median}(x,dim)$ - медианы массива;
- $\text{mean}(x)$, $\text{mean}(x,dim)$ - средние значения;

- $\text{std}(x)$, $\text{std}(x, \text{flag})$, $\text{std}(x, \text{flag}, \text{dim})$ - стандартное отклонение ($\text{flag}=0$ - несмещенная оценка для s ; $\text{flag}=1$ - смещенная оценка);
- $\text{cov}(x, y)$, $\text{cov}(x, y, \text{flag})$ - ковариация для массивов x и y (каждый столбец - переменная, строка - наблюдение);
- $\text{cov}(x)$, $\text{cov}(x, \text{flag})$ - ковариация для столбцов массива x ;
- $\text{corrcoef}(x)$, $\text{corrcoef}(x, y)$ - коэффициенты корреляции.

Для построения полиномиальной регрессионной кривой используется оператор $\text{polyfit}(x, y, n)$, возвращающий вектор коэффициентов полинома $p(x)$ степени n , который с наименьшей среднеквадратичной погрешностью аппроксимирует функцию $y(x)$. Результатом является вектор-строка длиной $(n+1)$, содержащая коэффициенты полинома в порядке уменьшения степеней x . Если $\text{length}(x)=n+1$, то реализуется обычная полиномиальная аппроксимация, при которой график полинома точно проходит через узловые точки с координатами (x, y) , хранящиеся в векторах x и y . В противном случае точного совпадения графика с узловыми точками не наблюдается. Полином строится в виде

$$p(1) * x^n + p(2) * x^{(n-1)} + \dots + p(n) * x + p(n+1).$$

Порядок выполнения работы

1. Написать m -файлы для реализации метода наименьших квадратов для построения линейной регрессии по x и y . Взять 10 первых точек из своего варианта. Вычислить медиану, средние значения, смещенную и несмещенную оценку среднеквадратичного отклонения, коэффициент корреляции.
2. Написать m -файлы функции для реализации метода наименьших квадратов для построения нелинейной полиномиальной регрессии со степенным базисом ($n=3, 5, 7$) и построить аппроксимирующие кривые.
3. Построить график и отобразить на нем следующие данные: исходные данные в виде точек и графики аппроксимирующих функций.
4. Использовать стандартные средства MATLAB для построения соответствующих кривых. Сравнить результаты.
5. Построить графики ошибок аппроксимации для всех n .

Содержание отчета

1. Цель работы, задание;
2. Описание метода решения, краткие сведения из теории (формулы, алгоритм и т.п.);
3. Программу (распечатку), ее описание;
4. Сравнение результатов расчета;
5. Краткие выводы.

Задание

Вариант 1. $x=[0:0.1:5]$

2,2097	-6,8439	-1,1201	-5,6987	-4,8524	1,7439	1,0118	-0,5526	6,1079	2,3993
1,6262	8,4938	-4,8348	10,1190	7,7189	11,5615	-3,3621	9,6958	17,9124	1,1666
-2,0931	4,5551	9,6773	18,1837	19,0898	7,0177	9,5207	23,9392	16,3089	22,8691
29,6142	35,6152	30,7599	48,6692	50,3154	28,7633	47,1599	32,2506	56,5752	3,4583
57,5376	70,1709	77,7399	69,4833	91,2748	85,4722	100,9104	98,6184	104,6567	1258,1061
121,2690									

Вариант 2. $x=[1:0.1:6]$

0,9770	0,7950	0,6755	0,6424	0,5276	0,4223	0,4701	0,3811	0,36	0,2269
0,1635	0,1913	0,2440	0,1776	0,1848	0,2027	0,1306	0,1317	0,1842	0,1531
0,1250	0,0713	0,1601	0,1217	0,1106	0,0325	-0,0109	0,0017	0,0237	0,0494
0,0590	0,1345	0,0776	0,0551	0,0402	0,0998	0,0805	0,0174	0,1028	0,0804
0,0264	-0,0383	0,0896	0,0043	0,0742	0,0487	0,0620	-0,0322	-0,0132	0,1340
0,0458									

Вариант 3. $x=[0:0.1:5]$

-0,0627	0,2727	0,4647	0,5818	0,7122	0,8727	0,9211	0,9423	0,9318	0,9329
0,9794	1,1605	1,0324	1,1018	1,0523	1,2707	1,2655	1,3511	1,2770	1,2805
1,3692	1,1997	1,4977	1,4779	1,7262	1,6303	1,7824	1,6529	1,7918	1,7635
1,8133	1,7746	1,7036	1,6771	1,8053	1,9213	1,8812	1,9930	2,0343	2,1015
2,1716	2,0307	1,9889	2,0551	2,0983	2,1195	2,1950	1,9982	2,2107	2,3409
2,2633									

Вариант 4. $x=[0.1:0.1:5.1]$

-2,3660	-1,7701	-1,3201	-1,2720	-0,5055	-0,3275	-0,4319	-0,4051	-0,1362	0,0405
-0,2024	0,3066	0,1005	-0,0493	0,3263	0,6423	0,0442	0,7559	0,5858	0,5291
0,4964	0,8012	0,7039	1,2297	0,9044	1,1075	1,3314	0,8089	0,7722	1,0514
1,3509	0,6801	1,2743	1,0409	1,2800	1,0181	1,2439	1,4303	1,3457	1,4073
1,1276	1,2935	1,3850	1,6022	1,6745	1,3950	1,2535	1,7307	1,8445	1,2650
1,6089									

Вариант 5. $x=[-2.5:0.1:2.5]$

0,4032	0,1809	-0,2587	-0,0399	-0,6520	0,1575	0,1644	0,3564	0,4596	-0,2643
0,8811	0,3974	1,5723	-0,0888	0,0314	-0,1035	0,9185	0,4832	0,4557	1,4323
1,4972	1,0005	0,0651	-0,2495	0,8214	1,8526	0,9669	1,0241	1,3991	1,4036

Вариант 12. $x=[-2:0.1:3]$

-1,6072	-1,4757	-0,7322	-0,7821	-0,3665	0,3615	0,5739	1,5260	0,8965	2,0787
1,9542	0,7020	0,9112	1,433	1,6451	1,7587	2,1061	1,4802	1,4615	1,6351
1,6585	1,7164	1,9547	1,6961	0,6543	1,5326	1,4709	1,2881	1,4408	0,9872
0,4062	0,4934	-0,5482	0,5004	0,2399	-0,1578	-0,2936	-0,5693	-0,2558	-0,1781
-0,5483	-0,6949	-0,5803	-0,6985	-1,0657	-0,2113	-0,5887	-0,2807	-1,0114	-1,2703
-0,5019									

Вариант 13. $x=[0:0.8:4]$

0,0059	-0,0274	0,1511	-0,0370	0,4939	0,2898	0,4232	0,8032	0,5692	0,7222
0,7837	1,1532	0,9884	0,8656	0,8900	1,3163	1,3501	1,3000	0,8693	0,9910
0,9580	0,9890	0,8174	0,7106	0,7361	0,9040	1,0387	0,6846	0,5683	0,6640
0,4547	0,7545	0,6140	0,2515	0,3056	0,5706	0,4154	0,4001	0,0869	0,3531
0,4111	0,2174	0,1631	0,2805	0,1600	0,3068	0,3666	0,5361	0,3825	0,3006
0,6375									

Вариант 14. $x=[0:0.8:4]$

1,8205	0,7979	0,9409	0,8066	2,0887	20372	2,1345	1,8844	2,336	2,1583
1,7639	2,1481	2,2576	2,1162	2,1942	2,1289	1,6940	1,3912	1,9982	1,6866
1,5450	1,3948	1,1302	1,3512	1,0199	1,0516	1,1084	1,2309	1,1400	0,8365
1,3023	1,0661	1,3884	0,9011	1,0076	1,0544	1,2169	1,1322	1,2020	1,0408
1,3792	1,6504	1,6726	1,7785	1,8982	2,0655	2,0827	2,1873	2,5384	2,5431
2,8220									

Вариант 15. $x=[2:0.06:5]$

2,0000	2,0600	2,1200	2,1800	2,2400	2,3000	2,3600	2,4200	2,4800	2,5400
2,6000	2,6600	2,7200	2,7800	2,8400	2,9000	2,9600	3,0200	3,0800	3,1400
3,2000	3,2600	3,3200	3,3800	3,4400	3,5000	3,5600	3,6200	3,6800	3,7400
3,8000	3,8600	3,9200	3,9800	4,0400	4,1000	4,1600	4,2200	4,2800	4,3400
4,4000	4,4600	4,5200	4,5800	4,6400	4,7000	4,7600	4,8200	4,8800	4,9400
5,5000									

Контрольные вопросы

1. Какие бывают виды аппроксимации экспериментальных данных?
2. В чем разница между аппроксимацией по методу наименьших квадратов и интерполяцией?
3. Что такое линейная регрессия? Как можно линеаризовать данные? Приведите примеры.
4. Что такое линейная по параметрам регрессия? Какие требования предъявляются к базисным функциям?
5. Построить алгоритм вычисления линейной по параметрам регрессионной модели со степенным базисом.
6. Что такое регрессионная матрица?

Лабораторная работа №3 «Регрессионный анализ»

Цель работы: овладеть способами реализации регрессии различного вида, наиболее часто применяемый в задачах обработки данных.

Теоретические сведения

Под регрессионным анализом (или просто регрессией) обычно подразумевают нахождение некоторой формальной аналитической зависимости, которая приближенно (по критерию минимума среднеквадратической ошибки) аппроксимирует исходную зависимость. Последняя чаще всего бывает представлена некоторым набором точек (например, полученных в результате эксперимента).

Реализация линейной регрессии общего вида

При этом виде регрессии заданная совокупность точек приближается к функции вида

$$F(x, K_1, K_2, \dots, K_n) = K_1 * F_1(x) + K_2 F_2(x) + \dots + K_n F_n(x).$$

Таким образом, функция регрессии является линейной комбинацией функций $F_1(x), F_2(x), \dots, F_n(x)$, причем сами эти функции могут быть нелинейными, что резко расширяет возможности такой аппроксимации и распространяет ее на многие нелинейные функции. Для реализации линейной регрессии общего вида (ЛРОВО) используется функция Linfit (VX, VY, F), возвращающая вектор K коэффициентов ЛРОВО, при котором среднеквадратичная погрешность приближения «облака» исходных точек, координаты которых хранятся в векторах VX и VY, оказывается минимальной. Вектор F должен содержать функции $F_1(x), F_2(x), \dots, F_n(x)$, записанные в символическом виде (рисунок 3.1).

$$VX := \begin{pmatrix} 1.1 \\ 2 \\ 3 \\ 4 \\ 4.8 \end{pmatrix} \quad VY := \begin{pmatrix} 25 \\ 12 \\ 9.4 \\ 16.2 \\ 26 \end{pmatrix} \quad F(x) := \begin{pmatrix} 1 \\ x \\ x^2 \\ \exp(x) \end{pmatrix}$$

$$K := \text{linfit}(VX, VY, F) \quad g(t) := F(t) \cdot K$$

$$i := 0..4 \quad r := 1, 1.25..5$$

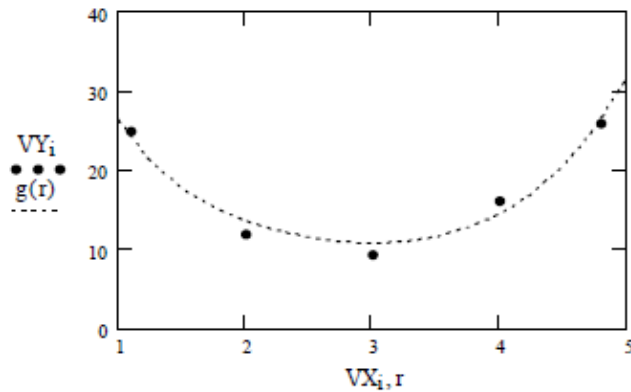


Рисунок 3.1 – Пример проведения линейной регрессии общего вида

Коэффициент функции регрессии

$$K = \begin{pmatrix} 26.064 \\ -0.284 \\ 0.228 \end{pmatrix}$$

Реализация одномерной полиномиальной регрессии

Для обеспечения полиномиальной регрессии при произвольной степени полинома вводится функция `regress (VX, VY, n)`, которая возвращает вектор `VS`, запрашиваемый функцией `interp (VS, VX, VY, x)` и содержащий коэффициенты многочлена n -й степени, который наилучшим образом приближает «облако» точек с координатами, хранящимися в векторах `VX` и `VY`. Для вычисления коэффициентов полинома регрессии в данном случае используется функция `submatrix` (рисунок 3.2).

$$\text{data} := \begin{pmatrix} 1 & 0.8 \\ 2 & 3.5 \\ 3 & 8 \\ 4 & 15 \\ 5 & 19 \\ 6 & 15 \end{pmatrix} \quad \text{Степень полинома } k := 3$$

Создание векторов X и Y исходных данных

$$X := \text{data}^{(0)} \quad Y := \text{data}^{(1)} \quad n := \text{rows}(\text{data})$$

$$z := \text{regress}(X, Y, k) \quad \text{fit}(x) := \text{interp}(z, X, Y, x)$$

$$i := 0..n - 1 \quad \text{coeffs} := \text{submatrix}(z, 3, \text{length}(z) - 1, 0, 0)$$

$$j := 0..49 \quad (\text{coeffs})^T = (7.133 \quad -11.231 \quad 5.734 \quad -0.606)$$

$$tx_j := \min(X) + j \cdot \frac{\max(X) - \min(X)}{50}$$

$$R_2 := \frac{\sum (\text{fit}(X) - \text{mean}(Y))^2}{\sum (Y - \text{mean}(Y))^2} = 0.995$$

График полинома и исходные точки

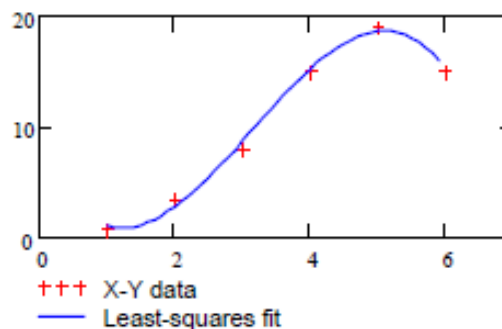


Рисунок 3.2 – Полиномиальная регрессия

На практике не рекомендуется делать степень аппроксимирующего полинома выше 4-6, поскольку погрешности реализации регрессии сильно возрастают. Функция regress создает единственный приближающий полином, коэффициенты которого вычисляются по всей совокупности заданных точек.

Регрессия отрезками полинома второй степени

Иногда, например, для фильтрации сигналов, полезна другая функция полиномиальной регрессии, дающая локальные приближения отрезками полиномов второй степени `loes` (`VX`, `VY`, `span`), которая возвращает вектор `VS`, используемый функцией `interp` (`VS`, `VX`, `VY`, `x`) для наилучшего приближения данных векторов `VX` и `VY` отрезками полиномов второй степени. Аргумент `span > 0` указывает размер локальной области приближаемых данных (рекомендуемое начальное значение `-0,75`). Чем больше `span`, тем сильнее сказывается сглаживание данных. При больших значениях `span` эта функция приближается к функции `regress` (`VX`, `VY`, `2`). Пример приближения сложной функции со случайным разбросом ее значений с помощью совокупности отрезков полиномов второй степени для двух значений параметра `span` показан на рисунке 3.3.

$$i = 1 \dots 199 \quad VX_i = i$$

$$VY_i = a \tan\left(\frac{i}{5}\right) \cdot \left(1 - \exp\left(\frac{-i}{10}\right) + 0.5 \cdot \text{md}(i)\right)$$

$$\text{span}_1 = 0.05 \quad \text{span}_2 = 0.5$$

$$VS_1 = \text{loess}(VX, VY, \text{span}_1) \quad VS_2 = \text{loess}(VX, VY, \text{span}_2)$$

$$F_1(x) = \text{interp}(VS_1, VX, VY, x)$$

$$F_2(x) = \text{interp}(VS_2, VX, VY, x)$$

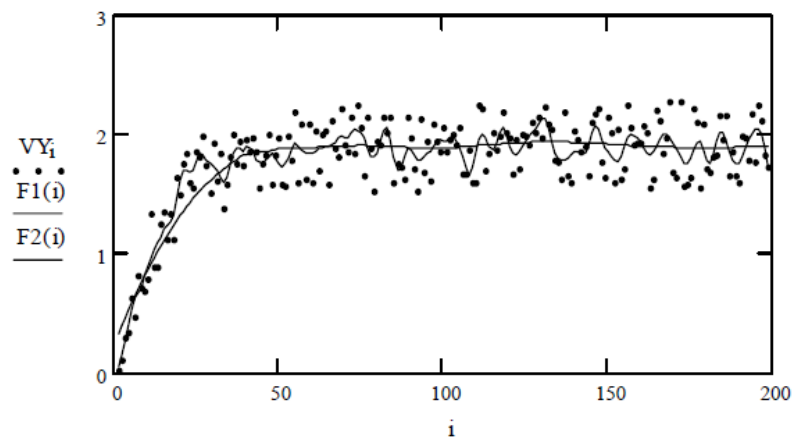


Рисунок 3.3 – Пример регрессии отрезками полиномов второй степени

Из рисунка 3.3 видно, что при значении $\text{span}=0.05$ отслеживаются характерные случайные колебания значений функции, тогда как уже при $\text{span}=0.5$ кривая регрессии становится практически гладкой. Недостатком данного вида регрессии является отсутствие простого описания аппроксимирующей функции в виде отрезков полиномов.

Проведение нелинейной регрессии общего вида

Под нелинейной регрессией общего вида (НРОВ) подразумевается нахождение вектора K параметров произвольной функции $F(x, K_1, K_2, \dots, K_n)$, при котором обеспечивается минимальная среднеквадратичная погрешность приближения «облака» исходных точек.

Для проведения НРОВ используются функции genfit (VX, VY, VS, F), которая возвращает вектор K параметров функции F , дающий минимальную квадратичную погрешность приближения функцией $F(x, K_1, K_2, \dots, K_n)$ исходных данных. Вектор F должен быть вектором с символьными элементами, содержащими аналитические выражения для исходной функции и ее производных, по всем параметрам. Вектор VS должен содержать начальные значения элементов вектора K , необходимые для решения системы нелинейных уравнений регрессии итерационным методом (рисунок 3.4).

$$\text{ORIGIN} := 1 \quad F(x, a, b) := a \cdot \exp(-b \cdot x) + a \cdot b$$

$$\frac{d}{da} F(x, a, b) \rightarrow \exp(-b \cdot x) + b \quad \frac{d}{db} F(x, a, b) \rightarrow a \cdot \exp(-b \cdot x) + a$$

$$F_1(x, k) := \begin{pmatrix} k_1 \cdot \exp(-k_2 \cdot x) + k_1 \cdot k_2 \\ \exp(-k_2 \cdot x) + k_2 \\ -k_1 \cdot x \cdot \exp(-k_2 \cdot x) + k_1 \end{pmatrix} \quad VX := \begin{pmatrix} 0.1 \\ 0.5 \\ 1 \\ 1.5 \\ 2 \\ 2.9 \end{pmatrix} \quad VY := \begin{pmatrix} 1.9 \\ 1.6065 \\ 1.34 \\ 1.22 \\ 1.1353 \\ 1.05 \end{pmatrix}$$

$$VS := \begin{pmatrix} 2 \\ 2 \end{pmatrix} \quad P := \text{genfit}(VX, VY, VS, F_1) \quad G(x) := F_1(x, P)_1$$

$$i := 1..6 \quad x := 0, 0.1..3$$

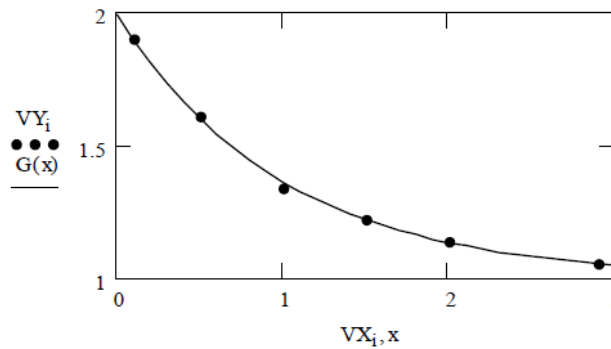


Рисунок 3.4 – Пример выполнения линейной регрессии общего вида с помощью функции $F(x,a,b)$

Вектор P возвращает значения $a = k_1$ и $b = k_2$ для наилучшего среднеквадратического приближения $F(x,a,b)$. вычисление значений произвольных по переменным a и b проведены средствами символьных операций (первая строка документа рисунок 3.4). замены параметра a на k_1 , а параметра b на k_2 связаны с необходимостью применения функции `genfit` в ее стандартном виде.

Порядок выполнения работы

1. Ознакомиться с особенностями проведения линейной регрессии, ЛРОВ, полиномиальной регрессии, регрессии отрезками полинома второй степени, НРОВ и регрессиями частного вида (`sinfit`, `earfit`, `logsfite`, `lgfit`, `medfit`, `pwfit`).
2. Для данных рисунка 3.2 провести линейную регрессию.
3. Выполнить синусоидальную регрессию для вектора данных $Y(x) = \sin(x) - N/10 + md(n/5)$, где N -номер варианта, а $x=0 \dots 50$.
4. Ознакомиться с возможностями многомерной регрессии (привести пример).

Содержание отчета

1. Цель работы, задание;
2. Описание метода решения, краткие сведения из теории (формулы, алгоритм и т.п.);
3. Программу (распечатку), ее описание;
4. Сравнение результатов расчета;
5. Краткие выводы.

Задание

1. Для данных примера 3 провести регрессии следующих видов:
 - полиномиальную (2-й и 3-й степени);
 - регрессию отрезками полиномов;
 - регрессию логарифмической функцией $\text{logfit}(x,y,q)$;
 - регрессию логистической функцией $\text{lgsfit}(x,y,q)$;
2. Дополнить пример 4 регрессией специального вида с помощью функции $\text{exrfit}(x,y,q)$;
3. Построить графики исходных и аппроксимирующих данных и провести сравнительный анализ с помощью коэффициента корреляции Пирсона.
4. Сделать выводы об эффективности различных видов регрессии.

Контрольные вопросы

1. В чем суть метода наименьших квадратов?
2. По каким критериям можно оценить точность аппроксимации?
3. Укажите последовательность проведения и область применения линейной регрессии для аппроксимации экспериментальных данных.
4. Укажите последовательность проведения и область применения линейной регрессии общего вида.
5. Укажите последовательность проведения и область применения аппроксимации экспериментальных данных полиномами.
6. Укажите последовательность проведения и область применения регрессии отрезками полинома второй степени.
7. Каково значение функции $\text{intercept}(VX,VY)$?
8. Каково значение функции $\text{slope}(VX,VY)$?
9. Каково значение функции $\text{linfit}(VX,VY,F)$?
10. Каково значение функции $\text{genfit}(VX,VY,VS,F)$?

Лабораторная работа №4 «Статистическая обработка результатов эксперимента в системе MATLAB»

Цель работы: получение навыков статистической обработки данных различными методами в системе MATLAB.

Теоретические сведения

Дискриминантный анализ, как раздел многомерного статистического анализа, включает в себя статистические методы классификации многомерных наблюдений в ситуации, когда исследователь обладает так называемыми обучающими выборками ("классификация с учителем"). Например, для оценки финансового состояния своих клиентов при выдаче им кредита банк классифицирует их по надежности на несколько категорий по ряду признаков. В случае, когда следует отнести клиента к той или иной категории используют процедуры дискриминантного анализа. Очень удобно использовать дискриминантный анализ при обработке результатов тестирования. Так при выборе кандидатов на определенную должность можно всех опрошенных претендентов разделить на две группы - удовлетворяющих и не удовлетворяющих предъявляемым требованиям.

Все процедуры дискриминантного анализа можно разбить на две группы и рассматривать их как совершенно самостоятельные методы. Первая группа процедур позволяет интерпретировать различия между существующими классами, вторая - производить классификацию новых объектов в тех случаях, когда неизвестно заранее, к какому из существующих классов они принадлежат.

Пусть имеется множество единиц наблюдения - генеральная совокупность. Каждая единица наблюдения характеризуется несколькими признаками: x_{ij} - значение j -й переменной i -го объекта ($i=1, \dots, n$; $j=1, \dots, p$). Предположим, что все множество объектов разбито на несколько подмножеств (два и более). Из каждого подмножества взята выборка объемом n_k , где k - номер подмножества (класса), $k = 1, \dots, q$.

Признаки, которые используются для того, чтобы отличать один класс (подмножество) от другого, называются дискриминантными переменными. Число объектов наблюдения

должно превышать число дискриминантных переменных: $p < n$. Дискриминантные переменные должны быть линейно независимыми. Основной предпосылкой дискриминантного анализа является нормальность закона распределения многомерной величины. Это означает, что каждая из дискриминантных переменных внутри каждого из рассматриваемых классов должна быть подчинена нормальному закону распределения.

Основная идея дискриминантного анализа заключается в том, чтобы определить, отличаются ли совокупности по среднему какой-либо переменной (или линейной комбинации переменных), и затем использовать эту переменную, чтобы предсказать для новых членов их принадлежность к той или иной группе. Канонической дискриминантной функцией называется линейная функция:

$$d_{km} = \beta_0 + \beta_1 \cdot x_{1km} + \dots + \beta_p \cdot x_{pkm},$$

где: d_{km} - значение канонической дискриминантной функции для m -го объекта в группе k ($m=1, \dots, n, k=1, \dots, g$); x_{pkm} - значение дискриминантной переменной X_i для m -го объекта в группе k ; $\beta_0, \beta_1, \dots, \beta_p$ - коэффициенты дискриминантной функции.

С геометрической точки зрения дискриминантные функции определяют гиперповерхности в p -мерном пространстве. В частном случае при $p=2$ она является прямой, а при $p=3$ — плоскостью.

Коэффициенты β_i первой канонической дискриминантной функции выбираются таким образом, чтобы центроиды (средние значения) различных групп как можно больше отличались друг от друга. Коэффициенты второй группы выбираются также, но при этом налагается дополнительное условие, чтобы значения второй функции были некоррелированы со значениями первой. Аналогично определяются и другие функции. Отсюда следует, что любая каноническая дискриминантная функция d имеет нулевую внутригрупповую корреляцию с d_1, d_2, \dots, d_{g-1} . Если число групп равно g , то число канонических дискриминантных функций будет на единицу меньше числа групп. Однако по многим причинам практического характера полезно иметь одну, две или же три дискриминантных функций. Тогда графическое изображение объектов будет представлено в одно-, двух- и трехмерных

пространствах. Такое представление особенно полезно в случае, когда число дискриминантных переменных p велико по сравнению с числом групп g .

Рассмотрим простой пример. Пусть имеются следующие наблюдения:

x_1	x_2
1	2
2	6
3	1
4	3
5	5

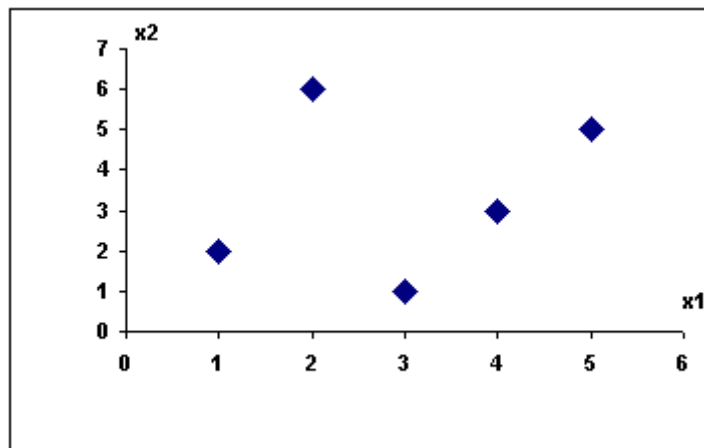


Рисунок 4.1 – Пример получения двух множеств

Из рисунка 4.1 видно, что наблюдения можно разделить на два множества (наблюдения 1,2 и наблюдения 3,4,5). Что бы наилучшим образом разделить множества, нужно построить соответствующую линейную комбинацию переменных x_1 и x_2 - дискриминантную функцию:

$$d = \beta_1 \cdot x_1 + \beta_2 \cdot x_2.$$

Обозначим \bar{x}_{ij} среднее значение j -го признака у объектов i -го множества. Тогда для первого множества среднее значение функции будет равно:

$$\bar{d}_1 = \beta_1 \cdot x_{11} + \beta_2 \cdot x_{12}$$

для второго множества:

$$\bar{d}_2 = \beta_1 \cdot x_{21} + \beta_2 \cdot x_{22}$$

Коэффициенты дискриминантной функции определяются таким образом, чтобы \bar{d}_1 и \bar{d}_2 как можно больше различались между собой. Требуется максимизировать разность $\bar{d}_1 - \bar{d}_2$. Вектор коэффициентов $V = (\beta_1, \beta_2)^T$ определяется из следующего выражения:

$$V = C^{-1}(\bar{X}_1 - \bar{X}_2),$$

где: C - объединенная ковариационная матрица наблюдений; \bar{X}_1, \bar{X}_2 - центры первого и второго множеств.

Объединенная ковариационная матрица в общем виде определяется из выражения:

$$C = (X_1^T X_1 + X_2^T X_2) / (n_1 + n_2 - 2),$$

где: X_1, X_2 - матрицы центрированных значений наблюдений двух групп; n_1, n_2 - количество наблюдений в каждой группе.

Для нашего примера матрица C имеет вид:

$$\begin{array}{cc} 0,83 & 2 \\ 2 & 5,3 \end{array}$$

обратная матрица C^{-1} имеет вид:

$$\begin{array}{cc} 13,3 & -5 \\ -5 & 2,1 \end{array}$$

Вектор V имеет вид:

$$\begin{array}{c} -3,46 \\ 1,3 \end{array}$$

Таким образом:

$$\bar{d}_1 = \beta_1 \cdot \bar{x}_{11} + \beta_2 \cdot \bar{x}_{12} = 3,46 \cdot 1,5 + 1,3 \cdot 4 = 0,01;$$

$$\bar{d}_2 = \beta_1 \cdot \bar{x}_{21} + \beta_2 \cdot \bar{x}_{22} = 3,46 \cdot 4 + 1,3 \cdot 3 = -9,94,$$

и разность $\bar{d}_1 - \bar{d}_2$ составляет:

$$\bar{d}_1 - \bar{d}_2 = 0,01 + 9,94 = 9,95.$$

Для классификации новых данных нужно определить границу, разделяющую в частном случае две рассматриваемые группы. Такой величиной может быть значение функции, равноудаленное от \bar{d}_1 и \bar{d}_2 :

$$A = (\bar{d}_1 + \bar{d}_2) / 2 = (0,01 - 9,94) / 2 = -4,965.$$

Величина A называется константой дискриминации. Объекты расположенные выше прямой:

$$-3,46 \cdot x_1 + 1,3 \cdot x_2 = -4,965$$

расположены ближе к центру первой группы и могут быть отнесены к этой группе. Представленная функция называется функцией классификации.

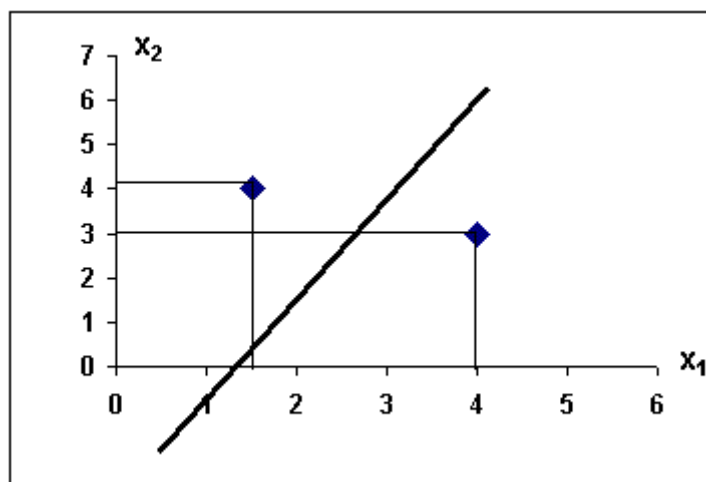


Рисунок 4.2 – Функция классификации

Пусть новое наблюдение имеет значения: $x_1 = 1$, $x_2 = 3$. Тогда:

$$-3,46 \cdot x_1 + 1,3 \cdot x_2 = 3,46 \cdot 1 + 1,3 \cdot 3 = 0,44 > -4,965$$

и следовательно наблюдение принадлежит первой группе.

Пусть следующее наблюдение имеет значения: $x_1 = 3, x_2 = 1$.

Тогда:

$$-3,46 \cdot x_1 + 1,3 \cdot x_2 = 3,46 \cdot 3 + 1,3 \cdot 1 = -6,48 < -4,965$$

и, следовательно, наблюдение принадлежит второй группе.

Дискриминантный анализ. Классификация при наличии k обучающих выборок

При необходимости возможно разбиение множества объектов (наблюдений) на k классов более двух. В этом случае необходимо построение k дискриминантных функций аналогично тому как это представлено в предыдущем параграфе.

Рассмотрим пример. Пусть имеются следующие наблюдения:

x_1	x_2
1	8
2	6
3	10
4	4
5	1
6	4
7	10
8	6
9	8

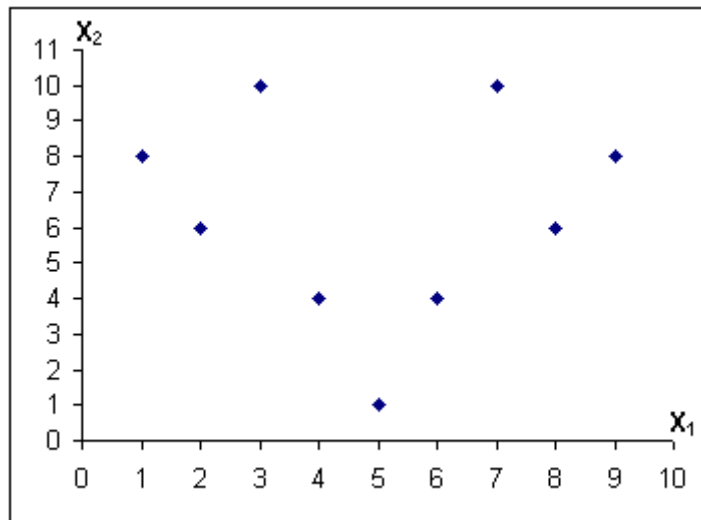


Рисунок 4.3 – Пример получения трех множеств

Наблюдения разделены на три множества (наблюдения 1,2,3, наблюдения 4,5,6 и наблюдения 7,8,9). Что бы наилучшим образом разделить множества, нужно построить соответствующую линейную комбинацию переменных x_1 и x_2 - дискриминантную функцию:

$$d = \beta_1 \cdot x_1 + \beta_2 \cdot x_2.$$

Обозначим x_{ij} среднее значение j -го признака у объектов i -го множества. Тогда для первого множества среднее значение функции будет равно:

$$\bar{d}_1 = \beta_1 \cdot x_{11} + \beta_2 \cdot x_{12}.$$

для второго множества:

$$\bar{d}_2 = \beta_1 \cdot x_{21} + \beta_2 \cdot x_{22}.$$

для третьего множества:

$$\bar{d}_3 = \beta_1 \cdot x_{31} + \beta_2 \cdot x_{32}.$$

Произведя вычисления по рассмотренной ранее схеме получим следующие функции классификации:

для классификации первого и второго множеств:

$$-4 \cdot x_1 + 2 \cdot x_2 = 3;$$

для классификации второго третьего множеств:

$$-6,2 \cdot x_1 - 3,2 \cdot x_2 = -57,8;$$

для классификации первого и третьего множеств:

$$-6,4 \cdot x_1 - 0,8 \cdot x_2 = -38,4.$$

Графики функций представлены на рисунке 4.4.

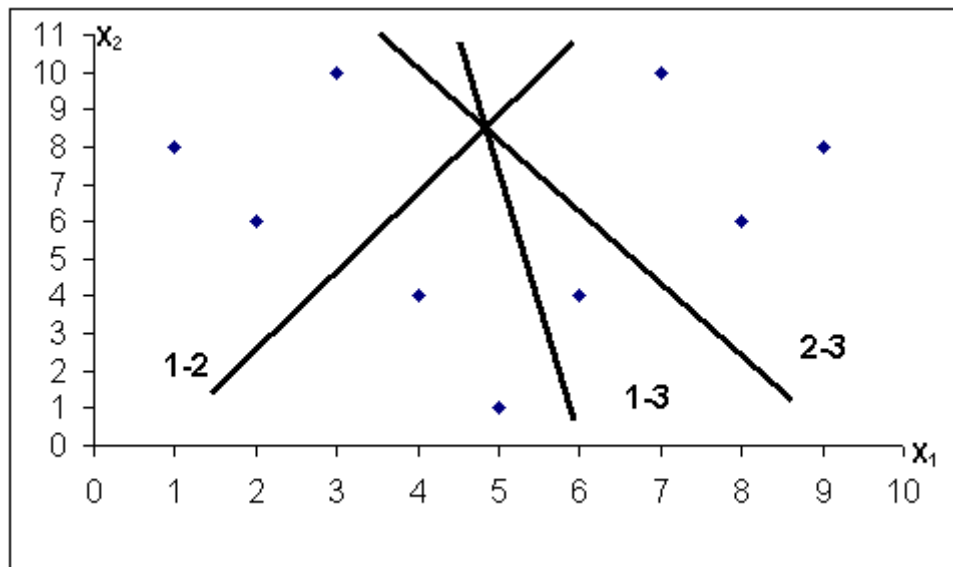


Рисунок 4.4 – График функций

Пусть новое наблюдение имеет значения: $x_1 = 3, x_2 = 7$. Тогда значения дискриминантных функций будут следующими:

$$f_{12} = -4 \cdot x_1 + 2 \cdot x_2 = -4 \cdot 3 + 2 \cdot 7 = -2 > -3;$$

$$f_{23} = -6,2 \cdot x_1 - 3,2 \cdot x_2 = -6,2 \cdot 3 - 3,2 \cdot 7 = -41 > -57,8;$$

$$f_{13} = -6,4 \cdot x_1 - 0,8 \cdot x_2 = -6,4 \cdot 3 - 0,8 \cdot 7 = -24,8 > -38,4.$$

Если $x_1 = 7, x_2 = 3$, то:

$$f_{12} = -4 \cdot x_1 + 2 \cdot x_2 = -4 \cdot 7 + 2 \cdot 3 = -22 < -3;$$

$$f_{23} = -6,2 \cdot x_1 - 3,2 \cdot x_2 = -6,2 \cdot 7 - 3,2 \cdot 3 = -53 > -57,8;$$

$$f_{13} = -6,4 \cdot x_1 - 0,8 \cdot x_2 = -6,4 \cdot 7 - 0,8 \cdot 3 = -47,2 < -38,4.$$

Необходимо отметить, что изменение числа переменных существенно влияет на результат дискриминантного анализа. Поэтому для определения необходимости включения (исключения) дискриминантной переменной обычно используют специальные критерии.

Для оценки вклада отдельной переменной в значение дискриминантной функции целесообразно пользоваться стандартизованными коэффициентами дискриминантной функции. Вычислить стандартизованные коэффициенты можно по следующей формуле:

$$a_j = \beta_j \cdot (c_{ij} / (p - m))^{1/2},$$

где: p - общее число исходных переменных; m - число групп; c_{ij} - элементы ковариационной матрицы.

Стандартизованные коэффициенты не зависят от масштабов измерений переменных. Именно поэтому по абсолютным величинам этих коэффициентов можно определить какая дискриминантная переменная вносит наибольший или наименьший вклад в значение дискриминантной функции.

Порядок выполнения работы

Порядок выполнения лабораторной работы будет рассмотрен на примере двух кластеров A01 и A02, которые состоят из ста объектов, характеризующихся двумя признаками x_1 и x_2 .

Объекты каждого кластера можно представить в виде множества точек на плоскости. По оси абсцисс откладывается значение первого признака, по оси ординат – второго.

```
figure
hold on
plot(A01(:,1),A01(:,2),'b','.')
plot(A02(:,1),A02(:,2),'r','.')

```

```
hold off
set(gca,'XLim',[0 1],'YLim',[0 1])
```

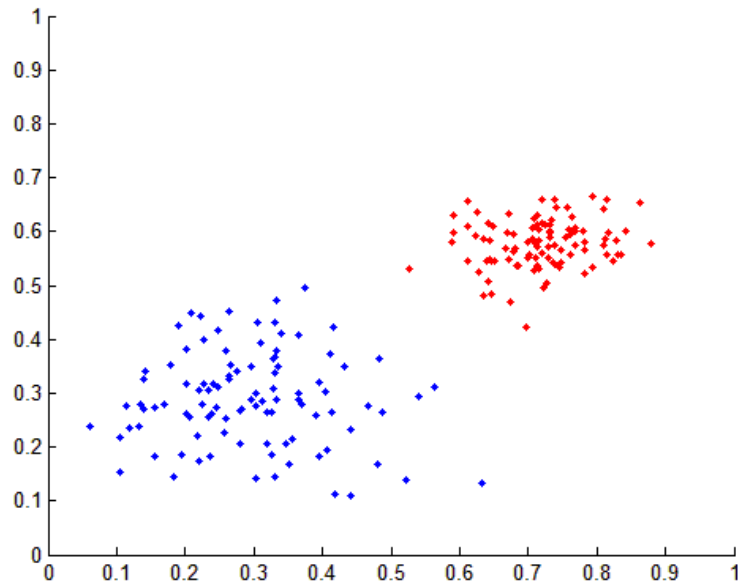


Рисунок 4.5 - Два кластера A01 и A02, которые состоят из ста объектов, характеризующихся двумя признаками x_1 и x_2

1 Линейные дискриминантные функции

1.1 Вектор проекций.

Основной целью дискриминантного анализа является нахождение такой линейной комбинации переменных (координат вектора проекций), при которой оптимально разделяются исследуемые кластеры, т.е. при которой расстояние между проекциями кластеров на вектор максимально. В каноническом дискриминантном анализе расстояние между проекциями оценивается через разность математических ожиданий, в дискриминантном анализе Фишера – через дискриминант Фишера (критерий J).

Программная реализация описанных методов в рамках решаемой задачи одинакова. При этом следует помнить, что поиск координат вектора проекций и последующие расчеты осуществляются на обучающей выборке или на части контрольной выборки (в рассматриваемом примере A1 и A2).

```
N1=length(A1(:,1));%Количество объектов в кластере A1
N2=length(A2(:,1));%Количество объектов в кластере A2
M1=mean(A1);%Мат. ожидание кластера A1
```

```

M2=mean(A2);%Мат. ожидание кластера A2
    X1=zeros(N1,2);
for i=1:N1
X1(i,:)=A1(i,:)-M1;
end
S1=X1'*X1;%Ковариационная матрица кластера A1
X2=zeros(N2,2);
for i=1:N2
    X2(i,:)=A2(i,:)-M2;
end
S2=X2'*X2;%Ковариационная матрица кластера A2
S=S1+S2;%Объединенная ковариационная матрица
W=abs(S*(M1-M2)');%Вектор проекций
figure
plot(A1(:,1),A1(:,2),'b')
hold on
plot(A2(:,1),A2(:,2),'r')
plot([0 W(1)],[0,W(2)],'r')

```

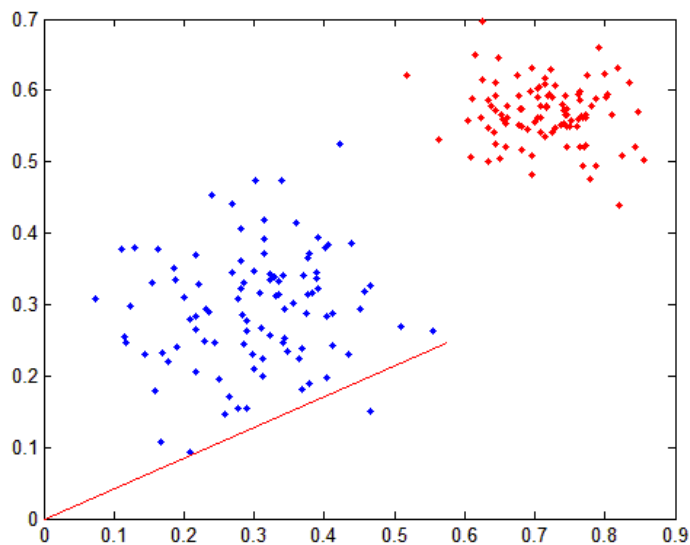


Рисунок 4.6 – Вектор проекций

1.2 Гистограмма проекций.

Чтобы определить оптимальное положение разделяющей прямой, необходимо спроецировать объекты кластеров на полученный вектор проекций.

```

Xp1=zeros(1,N1);
for i=1:N1

```

```

    Xp1(i)=A1(i,:)*W;%Массив проекций кластера A1
end
Xp2=zeros(1,N2);
for i=1:N2
    Xp2(i)=A2(i,:)*W;%Массив проекций кластера A2
end
[y1,x1]=hist(Xp1,10);
[y2,x2]=hist(Xp2,10);
figure
hold on
bar(x1,y1,'b')
bar(x2,y2,'r')
hold off

```

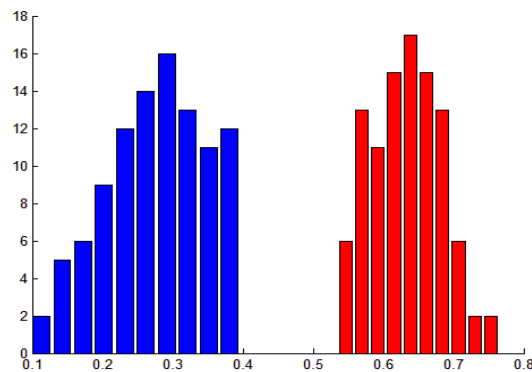


Рисунок 4.7 – Гистограмма проекций

1.3 Положение разделяющей прямой.

Положение разделяющей прямой можно найти двумя способами: методом минимизации ошибки (1) и методом минимизации стоимости ошибки (2).

При решении задачи методом 2 (для случая нормального распределения данных, равновероятного появления объектов каждого кластера и одинаковой стоимости ошибочной верификации объектов каждого кластера) положение разделяющей прямой определяется как среднее арифметическое математических ожиданий проекций.

```

m1=mean(Xp1); %мат. ожидание проекций кластера A1
m2=mean(Xp2); %мат. ожидание проекций кластера A2
a=(m1+m2)/2; %разделяющая прямая

```

При решении задачи методом 1 последовательно перебираются возможные значения разделяющей прямой, для них вычисляются ошибки верификации. Оптимальным считается такое положение разделяющей прямой, при котором ошибка минимальна.

```

a0=min(min(Xp1),min(Xp2)):0.01:max(max(Xp1),max(Xp2));
Na=length(a0);
e=zeros(1,Na);
e1=zeros(1,Na);
e2=zeros(1,Na);
for i=1:Na
    for j=1:N1
        if Xp1(j)>=a0(i)
            e1(i)=e1(i)+1;%ошибка первого рода
        end
    end
    for k=1:N2
        if Xp2(k)<a0(i)
            e2(i)=e2(i)+1;%ошибка второго рода
        end
    end
    e(i)=e1(i)+e2(i);%суммарная ошибка
end
[em,im]=min(e);%минимум ошибки
plot(e)
a=a0(im);%разделяющая прямая
figure
hold on
bar(x1,y1,'b')
bar(x2,y2,'r')
stem(a,20,['g','!'])
hold off

```

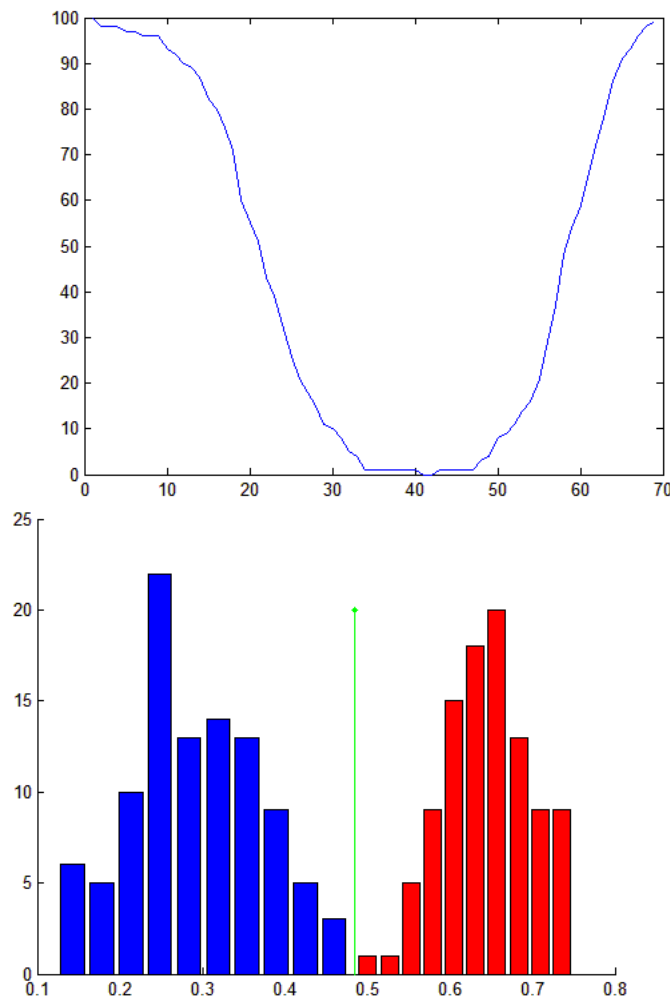



Рисунок 4.8 – Положения разделяющей прямой

1.4 Разделяющая прямая в пространстве признаков.

В пространстве признаков разделяющая прямая перпендикулярна вектору проекций и пересекает его на расстоянии, найденном в предыдущем пункте, от начала координат.

%Поиск координат ортогонального вектора осуществляется по уравнению:

*%ax*W(1)+ay*W(2)=0*

ax=1/10;%абсцисса вектора, ортогонального вектору проекций

*ay=-ax*W(1)/W(2);%ордината вектора, ортогонального вектору проекций*

*aa=a*pinv(W);%смещение ортогонального вектора вдоль вектора проекций на расст-е a*

figure

hold on

plot(A1(:,1),A1(:,2),'b')

plot(A2(:,1),A2(:,2),['r','r'])

```

plot([0 W(1)], [0, W(2)], 'k')
plot([aa(1) ax+aa(1)], [aa(2), ay+aa(2)], 'g')
hold off

```

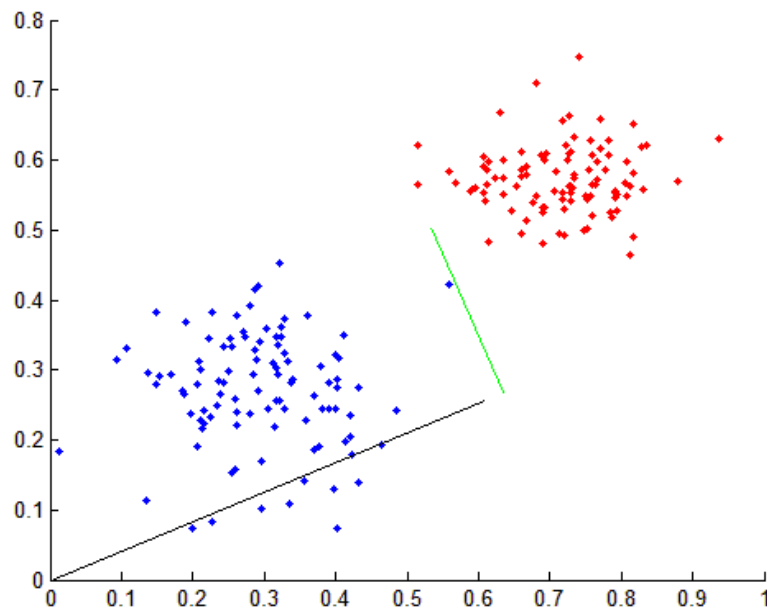


Рисунок 4.9 – Разделяющая прямая в пространстве признаков

1.5 Ошибки кластеризации.

Вычисление ошибок производится для контрольной выборки или, в случае если в качестве обучающей выборки была взята часть исследуемых данных, для всей выборки.

```

E1=0;
E2=0;
for i=1:N1
    if A01(i,:)*W1>=a
        E1=E1+1;%ошибка первого рода
    end
end
for i=1:N2
    if A02(i,:)*W1<a
        E2=E2+1;%ошибка второго рода
    end
end
E=E1+E2;%суммарная ошибка

```

2 Непараметрические методы

2.1 Вектор проекций.

Целью непараметрических методов является нахождение таких координат вектора проекций, при которых проекции одного кластера будут положительными, а другого – отрицательными. Таким образом, кластеризация проводится при сравнении значений проекций с нулем. Поиск координат вектора осуществляется путем последовательного пересчета их значений относительно координат текущего объекта и заданного коэффициента. Коэффициент может быть постоянным (метод непараметрического обучения с постоянным коэффициентом) или переменным (методы непараметрического обучения с абсолютной или частичной коррекцией).

При программной реализации методов можно столкнуться с ситуацией, когда невозможно разделить кластеры с абсолютной точностью, что влечет за собой бесконечную цикличность процесса. Чтобы избежать такой ситуации можно задать допустимый уровень ошибки (п. 2.1.2) или максимальное число итераций (п. 2.1.3).

```
clear X1 X2 W1
X1(:,1:2)=A1;
X1(:,3)=1;%Добавление столбца, заполненного единицами в
кластер A1
X2(:,1:2)=A2;
X2(:,3)=1;%Добавление столбца, заполненного единицами в
кластер A2
X2=X2*(-1);%Умножение второго кластера на -1
X=[X1; X2];%Объединение матриц
N=length(X(:,1));
```

2.1.1 С постоянным коэффициентом.

```
C1=1
W1=[1,1,1];%Начальное значение вектора проекций
i=0;
e1=1;
while e1>0
    i=i+1;
```

```

if i>N
i=1;
end
    e1=0;
    if X(i,:)*W1'<0
        W1=W1+C1*X(i,:); %Пересчет координат вектора
        проекций
    end
    for j=1:N
        if X(j,:)*W1'<0
            e1=e1+1; % Уровень ошибки
        end
    end
end
end

```

2.1.2 С абсолютной коррекцией.

```

W2=[1,1,1];%Начальное значение вектора проекций
i=0;
e2=1;
while e2>=10%Допустимый уровень ошибки
    i=i+1;
    if i>N
        i=1;
    end
    e2=0;
    if X(i,:)*W2'<0
        C2=abs(W2*X(i,:)')/(X(i,:)*X(i,:)');
        W2=W2+C2*X(i,:); %Пересчет координат вектора
        проекций
    end
    for j=1:N
        if X(j,:)*W2'<0
            e2=e2+1; % Уровень ошибки
        end
    end
end
end

```

2.1.3 С частичной коррекцией.

```

W3=[1,1,1];%Начальное значение вектора проекций

```

```

i=0;
e3=1;
M=0;
while e3>0&&M<1000%Максимальноечисло итераций
    i=i+1;
    if i>N
        i=1;
    end
    e3=0;
    if X(i,:)*W3'<0
        C3=2*abs(W3*X(i,:)')/(X(i,:)*X(i,:)');
        W3=W3+C3*X(i,:); %Пересчет координат вектора
        проекций
    end
    for j=1:N
        if X(j,:)*W3'<0
            e3=e3+1; %Уровень ошибки
        end
    end
    M=M+1; %Счетчик итераций
end

```

2.2 Гистограмма проекций.

Аналогично п. 1.2.

2.3 Положение разделяющей прямой.

В случае абсолютной делимости кластеров проекция разделяющей прямой проходит через ноль. Если же кластеры невозможно разделить с нулевой ошибкой (если приходится задавать допустимый уровень ошибки или максимальное число итераций), положение разделяющей прямой приходится уточнять с помощью метода минимизации ошибки или метода минимизации стоимости ошибки (п.1.3).

2.4 Разделяющая прямая в пространстве признаков.

Аналогично п. 1.4.

2.5 Ошибки кластеризации.

Аналогично п. 1.5.

Содержание отчета

1. Цель работы, задание;
2. Описание метода решения, краткие сведения из теории (формулы, алгоритм и т.п.);
3. Программу (распечатку), ее описание;
4. Сравнение результатов расчета;
5. Краткие выводы.

Задание

1. Используя генератор случайных чисел пакета MathCad создайте как обучающие, так и контрольные выборки с объемом и размерности пространства информативных признаков, заданные преподавателем. Пример листа MathCad, в котором создается выборка из 28 объектов с нормально распределенным законом распределения признаков (признаковое пространство четырехмерное) показан на рисунке 4.10.

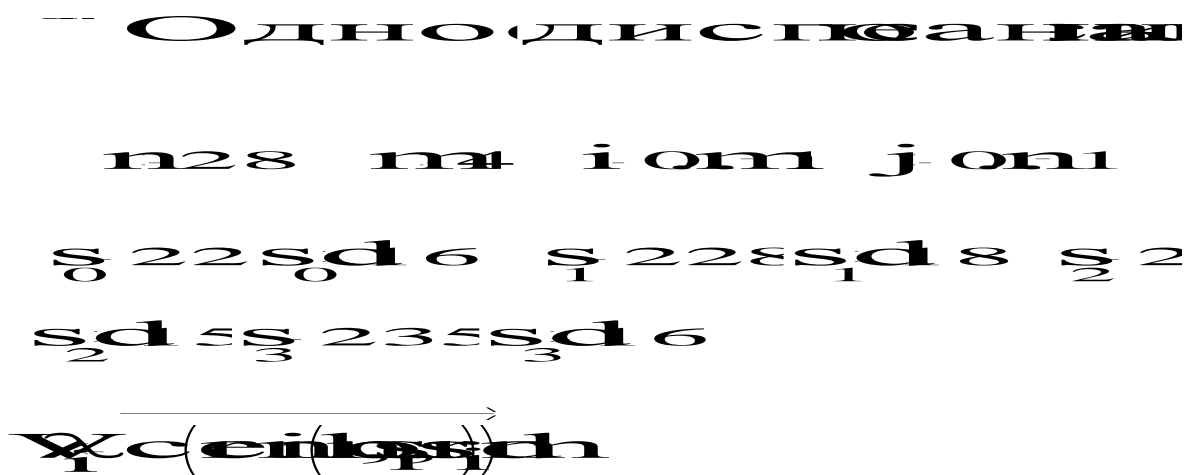


Рисунок 4.10. - Лист MathCad с генератором случайных чисел

Каждый информативный признак в выборке представлен вектором-столбцом V_{λ} из 28 элементов. Случайные числа, распределенные по нормальному закону распределения, генерируются функцией

$$\text{rnorm}(n, s, \text{sd}),$$

где n -число элементов в выборке, s – математическое ожидание информативного признака, sd -дисперсия информативного признака.

Варьируя эти параметры (s и sd) вы можете менять структуру распределения классов в признаковом пространстве и исследовать эффективность классификации при различных классовых структурах.

Функция `ceil` округляет случайное число до ближайшего целого.

Значок \longrightarrow обозначает операцию векторизации, то есть одновременное выполнение скалярной операции над всеми элементами вектора.

2. В соответствии с порядком выполнения работы проведите статистическую обработку результатов эксперимента.

3. Объясните полученные результаты.

4. Сделайте выводы.

Контрольные вопросы

1. Что такое дискриминантный анализ?

2. Какова основная идея дискриминантного анализа?

3. Укажите последовательность проведения дискриминантного анализа.

4. Укажите последовательность проведения классификации при наличии k обучающих выборок.

5. Как в системе MATLAB программно реализуется вектор проекций?

6. Как в системе MATLAB программно реализуется гистограмма проекций?

7. Как в системе MATLAB программно реализуется разделяющая прямая в пространстве признаков?

8. Как создать обучающие и контрольные выборки, используя генератор случайных чисел пакета MathCad?

Лабораторная работа №5 «Основы программирования в MATLAB»

Цель работы: изучение методики проектирования программ цифровой обработки сигналов (ЦОС): моделирования сигналов, нормально распределенного и белого шума, программирования операций математической обработки, операций графического вывода данных, проектирования средств графического пользовательского интерфейса.

Теоретические сведения

Программные средства систем ЦОС обычно создаются на языке C/C++, так как в этом языке предусмотрены средства, обычно необходимые при аппаратной реализации на базе программируемых логических интегральных схем (ПЛИС) и сигнальных процессоров: удобные средства работы с битами, логические операции, средства работы с аппаратными прерываниями и др. Кроме того, компилятор C обеспечивает формирование исполняемой программы, близкой по скорости исполнения к программе на ассемблере.

При разработке систем жесткого реального времени на базе ПЛИС и быстродействующих сигнальных процессоров необходимо производить отработку алгоритмов ЦОС на универсальных средствах (с помощью компьютерных программ) используя низкоуровневое программирование без библиотечных функций (на C/C++) с тем, чтобы на следующем этапе можно было реализовать разработанные алгоритмы ЦОС на ПЛИС и сигнальных процессорах. Кроме того, как известно, ЦОС должна быть выполнена в течение критического срока обслуживания и это так же важно, как и корректность алгоритма ЦОС. Невыполнение задачи реального времени в течение критического срока обслуживания равносильно невыполнению задачи в целом. Поэтому очень важно, чтобы на втором этапе разработки отработка алгоритмов ЦОС производилась с использованием соответствующей аппаратуры и скорость выполнения операций ЦОС контролировалась, например, с использованием системного таймера компьютера.

Конечной целью разработки и исследования различных алгоритмов ЦОС является их практическая реализация в устройствах на базе ПЛИС и сигнальных процессоров, таких,

например, как модуль ЦОС SAMC-401, содержащий ПЛИС серии Virtex-4, в которой интегрированы два процессора Power PC с тактовой частотой до 450 МГц, и сигнальный процессор Texas Instruments TMS320C6455, работающий на частоте 1.2ГГц с submodule АЦП SAMC-ADC, реализующий аналого-цифровое преобразование 12/14 бит с предельной частотой тактирования до 210 МГц.

Основные стадии разработки алгоритмов ЦОС:

1. Высокоуровневая программная (MATLAB).
2. Низкоуровневая программная (LabWindows/CVI, C/C++).
3. Аппаратно-программная (LabWindows/CVI, C/C++).
4. Аппаратная (VHDL, C/C++).

На первом этапе разработки наиболее совершенным средством является MATLAB, т.к. он содержит библиотеки функций для сложных видов математической обработки, таких как быстрое преобразование Фурье, цифровой фильтрации, корреляционной обработки и др., библиотеки программ для создания объектов графического пользовательского интерфейса (панелей, кнопок управления, окон цифрового ввода/вывода, окон графического вывода и др.).

Программные средства, создаваемые для работы в среде MATLAB, содержат две составляющие:

1. Собственно программу (mat-файл) на языке C, в которой содержатся необходимые функции математической обработки, отображения таблиц и графиков. При ее создании следует пользоваться описаниями функций библиотек математической обработки и графического пользовательского интерфейса.

2. Файлы ресурсов графического пользовательского интерфейса <имя программы>.fig.

Создаваемая пользователем прикладная программа представляет проект, содержащий два файла:

- файл основной программы <имя>.mat;
- файл макета <имя>.fig;

Программа на языке C может быть написана и редактироваться пользователем. Файлы макета <имя>.fig создаются автоматически при создании и редактировании пользователем графических панелей. Эти файлы нельзя редактировать!

Создание средств графического пользовательского интерфейса в традиционных системах программирования, таких как Visual C++

возможно, но достаточно сложно, так как для этого необходимо создание большого количества нестандартных графических объектов.

Создание прикладной программы ЦОС в среде MATLAB

1. Создать панель интерфейса пользователя: File/New/GUI/Blank GUI. В результате появится всплывающее диалоговое окно `untitled.fig`

2. Разместить элементы управления (кнопки управления `PushButton`, окна ввода/вывода текста `EditText`, окна графического вывода `Axes` и др.), например, так, как показано на рисунке 5.1.

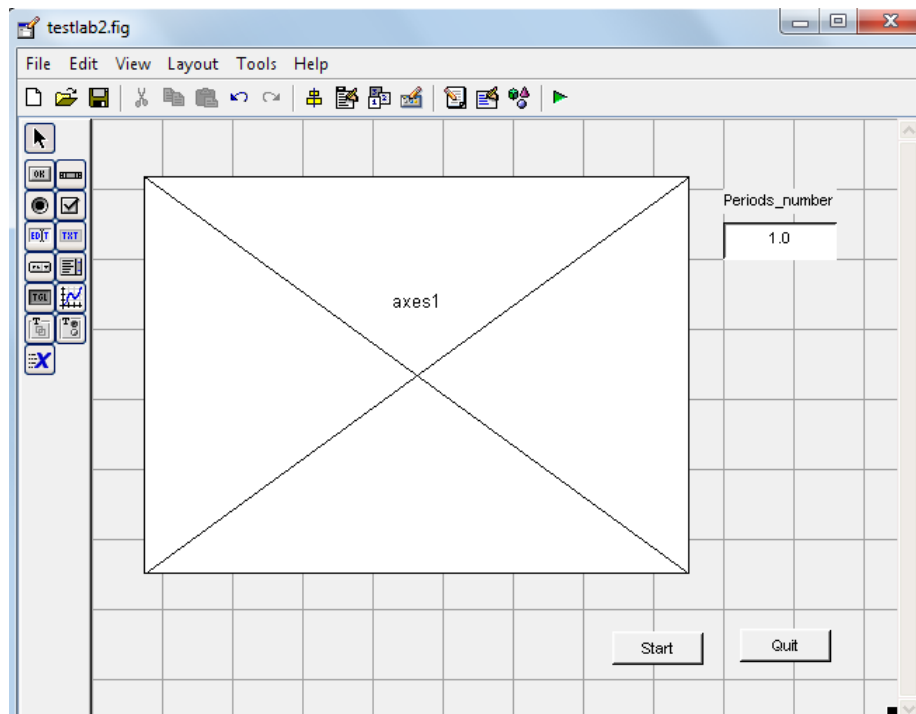


Рисунок 5.1 – Панель интерфейса пользователя

и задать их атрибуты. Панель атрибутов элемента управления вызывается двойным щелчком левой кнопки мыши на изображении элемента. Основные атрибуты кнопки управления и окна ввода/вывода текста – `Callback`, `CreateFcn` и `tag`. Например, при задании атрибуту `tag` окна ввода/вывода текста значения `freq` имя окна будет `freq`. Такое же имя может быть дано и переменной, значение которой читается из этого окна.

Атрибуты `Callback`, `CreateFcn` создаются автоматически после задания `tag`.

3. Сохранить `untitled.fig` в файле с именем `<имя>.fig`. В результате будет не только сохранен файл интерфейса пользователя, но и создан и сохранен в файле `<имя>.m` «скелет» программы.

4. Создать содержательную часть программы в блоке функции, вызываемой, например, нажатием кнопки управления (Приложение 2).

В итоге будут созданы два файла: файл программы с расширением `.m` и файл GUI с расширением `.fig`. Для внесения изменений в интерфейс пользователя нужно вызвать файл командой `>>guide ('<полное имя файла, включая путь>.fig')`.

Загрузка и запуск MATLAB

Для загрузки и запуска LabWindows CVI 8.0 в среде ОС Windows XP/7 нужно вызвать из меню «Пуск»:

Programs/MATLAB

Создание программы

Создать файл макета.

а) File/New/GUI/Blank GUI

б) сохранить созданный файл макета File/Save Untitled.fig As.../{имя файла}

В результате на экране дисплея появляется изображение панели GUI. Размеры и место расположения панели на экране можно установить с помощью мыши. Далее нужно установить атрибуты панели. Для этого перейти в режим редактирования панели двойным щелчком мыши и во всплывающей панели задать:

- Name - идентификатор панели, который определяет имя `m`-файла и `fig`-файла.
- и другие.

4. Создать элементы GUI (графические окна вывода, текстовые окна ввода/вывода, командные кнопки и т.д.) из библиотеки элементов и разместить их на созданной панели.

Объекты GUI выбираются из меню и размещаются на экране методом "drag and drop" с помощью "мыши".

MATLAB предоставляет широкий выбор объектов:

Edit text - окно ввода/вывода текста;
 Static Text – окно вывода текстового сообщения;
 Push Button – кнопка с двумя состояниями;
 Toggle Button – кнопка с двумя состояниями
 (включено/выключено);
 Check Box – флажок;
 Radio Button – аналог Option Button;
 Pop-up-menu – меню;
 Slider – линейка прокрутки;
 List box – поле со списком
 Axis1 - окно вывода массивов данных в графической форме;
 При выполнении лабораторной работы рекомендуется
 использовать следующие элементы управления и ввода/вывода: Edit
 text, Axis1, Pushbutton.

4.1 Задание атрибутов элементов GUI.

Сделать двойной щелчок левой клавишей мыши на объекте GUI, далее во всплывающей панели задать его атрибуты, в первую очередь tag.

Далее нужно сохранить сделанные установки.

File/Save

/Save As.../{имя файла}

5. Создать программный код. Создание fig-файла производится автоматически при сохранении GUI-образа. Одновременно создается m-файл.

В созданном программном коде m-файла объектам GUI будет соответствовать "программная оболочка" функций на языке C++. Эти функции (за исключением функции работы таймера) будут запускаться на исполнение пользователем с панели GUI. В приложении 1 приведен пример созданной таким путем «программной оболочки».

Далее в эту оболочку вносится программный код прикладной программы на языке C++.

Если производится редактирование ранее уже созданного программного кода, например, при добавлении какого-либо объекта GUI, то для добавления в ранее созданный программный код изменений, связанных с добавленными элементами GUI нужно открыть файл <>.fig для редактирования командой:

>>guide ('<полное имя файла, включая путь>.fig')

в окне Command Window. Путь к файлу можно не указывать, если поместить его в папку MATLAB.

6. Добавить в программу функцию выхода.

Для этого в уже сгенерированную по п.5 функцию выхода программы необходимо внести программный код:

```
fclose('all');
close ('all');
```

7. Добавить в функцию, вызываемую кнопкой на панели GUI (в приведенном примере - Start), строки обращения к элементам ввода/вывода GUI, текст программы. Ниже приведен пример программирования ввода текста из окна ввода/вывода (TextBox) с преобразованием в число типа double, вывода значения с преобразованием в текст, чтения номера элемента из поля со списком (ComboBox), вывода числового значения в TextBox:

```
periods_number=str2double(get(handles.periods_number,'string'));
    set(handles.chastotan,'String',fr_int);
    regim=get(handles.regim,'Value');
    set(handles.lampa,'Value',1);
```

Примечания:

1. Элементы поля со списком нумеруются, начиная с 1. Так же в MATLAB нумеруются и элементы массивов.

2. В атрибутах окна, предназначенного для вывода текстовых сообщений задать атрибут Style –Edit.

3. Для имитации сигнального индикатора можно использовать элемент radiobutton.

Описания типов переменных не обязательны, но возможны, если требуется. В случае необходимости передавать значения параметров из одной функции в другую нужно использовать описания глобальных переменных, например:

```
global N
```

В результате будет получен полный программный код прикладной программы. Простой пример такой программы генерации и графического отображения массива случайных чисел

приведен ниже. Количество периодов сигнала задается переменной `freq`, получаемой с элемента GUI:

```
freq=str2double(get(handles.freq,'string'));
for i=1:1000
    y(i)=sin((2*pi*i*freq)/1000.0);
end
i=1:1000;
plot(i,y);
```

Редактирование программного кода, в том числе строк обращения к объектам GUI производится точно так же, как в любом C/C++.

10. Запустить программу.

Debug/Run

Если компилятор найдет ошибки, то устранить ошибки и повторить предыдущее действие.

Порядок выполнения работы

1. Изучить состав и функции библиотек программ MATLAB для работы с аппаратурой, математической обработки, создания средств графического пользовательского интерфейса.
2. Разработать программу генерации синусоидального сигнала.
3. Разработать программу генерации стандартных сигналов: гармонического, пилообразного, треугольного, прямоугольного с нормально распределенным и белым шумом. Выбор вида сигнала, амплитуду сигнала и уровень шума сделать регулируемыми.

Задание

1. Изучить основы программирования в MATLAB по описанию к лабораторной работе и пользуясь справочной системой.
2. Создать графический пользовательский интерфейс и программу генерации синусоидального сигнала. Количество периодов сигнала сделать регулируемым.
3. Разработать прикладную программу генерации различных стандартных сигналов:
 - ◆ гармонического (синусоидального);

- ◆ пилообразного;
- ◆ треугольного;
- ◆ прямоугольных импульсов.

без шума и с нормально распределенным и белым шумом. Выбор вида сигнала, амплитуду, частоту сигналов и среднеквадратическую величину шума сделать регулируемыми.

3.1. Для генерации нормально распределенного и белого шума использовать функции `randn` и `wgn`. Пример фрагмента программы генерации модельного сигнала с шумом приведен ниже.

```
%Генерация нормального и белого шума
%noise=randn(points_number);%нормально распределенный
noise=wgn(points_number,1,0);%белый
for i=1:points_number %генерация модельного сигнала с шумом
    x(i)=sin(2*3.14*kp*i/kt)+noise_sko*noise(i);
end
```

Здесь `points_number` – количество элементов массива (количество значений сигнала), `noise_sko` – среднеквадратическое значение шума.

3.2. На панели графического пользовательского интерфейса предусмотреть (рисунок 5.2):

- ◆ переключатель выбора вида сигнала;
- ◆ элемент ввода уровня шума;
- ◆ элемент ввода количества точек за период генерируемого сигнала;
- ◆ окно графического вывода генерируемого сигнала.

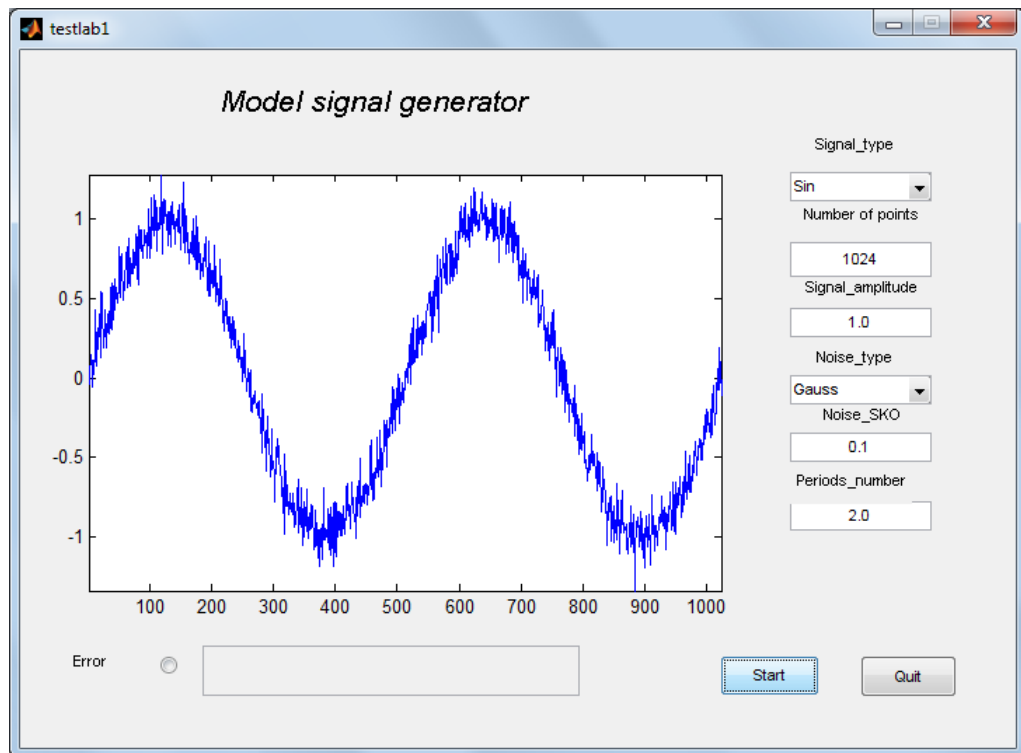


Рисунок 5.2 – Панель графического пользовательского интерфейса

3.3. Дополнить программу, предусмотрев в ней возможность накопления (суммирования с усреднением) генерируемых сигналов.

На панели графического пользовательского интерфейса дополнительно предусмотреть переключатель задания количества накоплений.

Содержание отчета

1. Задание к работе.
2. Текст программы с комментариями.
3. Вид панели интерфейса пользователя при генерации различных сигналов с различным шумом.

Контрольные вопросы

1. Перечислите основные стадии разработки алгоритмов ЦОС.
2. Какие две составляющие содержат программные средства, создаваемые для работы в среде MATLAB?
3. Укажите последовательность создания пользовательского интерфейса.

4. Укажите последовательность задания атрибутов элементов GUI.
5. Как можно произвести генерацию нормально распределенного и белого шума?

Лабораторная работа №6 «Обработка результатов однофакторного эксперимента в среде MATLAB. Подгонка кривых»

Цель работы: приобретение навыков обработки результатов однофакторного эксперимента в среде MATLAB/

Теоретические сведения

Обработка результатов однофакторного эксперимента включает 3 этапа:

- Выбор вида аппроксимирующей зависимости.
- Расчет коэффициентов аппроксимирующей зависимости.
- Проверку адекватности полученной зависимости экспериментальным данным.

Наиболее совершенным программным средством для решения данной задачи является пакет программ MATLAB. MATLAB является средой разработки программ обработки данных и, одновременно, содержит большое количество готовых программ, в частности, программу подгонки кривых CurveFitting.

Программа подгонки кривых CurveFitting предоставляет следующие виды функций для аппроксимации и интерполяции одномерных массивов данных:

а) экспоненциальную;

$$y = ae^{bx}$$

$$y = ae^{bx} + cd^{dx}$$

б) гауссиан

$$y = \sum_{i=1}^n a_i e^{\left[-\left(\frac{x-b_i}{c_i} \right)^2 \right]}$$

в) Фурье

$$y = a_0 + \sum_{i=1}^n a_i \cos(n\omega x) + b_i \sin(n\omega x)$$

г)полиномиальную;

$$y = \sum_{i=1}^{n+1} p_i x^{n+1-i}$$

д)показательную;

$$y = ax^b$$

$$y = a + bx^c$$

е)рациональную.

$$y = \frac{\sum_{i=1}^{n+1} p_i x^{n+1-i}}{x^m + \sum_{i=1}^m q_i x^{m-i}}$$

Критерием качества подгонки являются вычисляемые программой коэффициенты множественной детерминации (R-square и Adjusted R-square). Коэффициент R-square показывает насколько успешно разброс данных относительно аппроксимирующей кривой может быть объяснен наличием случайных погрешностей в данных. Максимально возможное значение R-square равно 1.

Коэффициент Adjusted R-square характеризует то же самое, но учитывает количество степеней свободы дисперсий воспроизводимости и адекватности. Поэтому Adjusted R-square лучше характеризует степень соответствия экспериментальных данных аппроксимирующей кривой в том случае, если мы увеличиваем степень аппроксимирующего полинома и хотим проверить, происходит ли при этом улучшение качества подгонки. Максимально возможное значение Adjusted R-square также равно 1.

$$SSE = \sum_{i=1}^n w_i (y_i - \hat{y}_i)^2 \quad SSE - \text{sum square error}$$

$$SST = \sum_{i=1}^n w_i (y_i - \bar{y})^2 \quad SST - \text{sum square total,}$$

$$w_i = \frac{1}{\sigma_i^2} - \text{коэффициент, выравнивающий степень разброса}$$

данных в опытах;

\hat{y} - значение полученное по аппроксимирующей зависимости

$$R - \text{square} = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

$$\text{Adjusted R - square} = 1 - \frac{SSE(n-1)}{SST(n-m)}$$

где n- количество результатов измерений;

m – количество членов в уравнении регрессии;

n-1 – количество степеней свободы дисперсии SSE;

n-m - количество степеней свободы дисперсии SST.

Порядок выполнения работы

1. Изучить теоретические основы подгонки кривых, положенные в основу программы Curve Fitting и описание библиотеки аппроксимирующих функций Fitting Library в MATLAB Help: CurveFittingToolbox/FittingData/ParametricFitting (Library Models, Custom Equations, Evaluation of goodness of Fitting и др.).

2. Выполнить подгонку кривых, выбрать наилучшую функциональную зависимость и обосновать сделанный выбор для зашумленного массива данных: файл censuN.mat) в папке E:\DSP_LAB_2012\Lab_4_MATLAB.

3. При работе с программой CurveFitting подгонку кривых производить в следующей последовательности:

а) Запустить MATLAB.

б) открыть окно Workspace (View/Workspace) и загрузить в это окно mat-файл данных по заданию преподавателя (census.mat, censu1.mat, censu2.mat и т.д.)

в) открыть панель CurveFitting для чего выбрать Start/Toolboxes/ CurveFitting в окне MATLAB;

г) на панели CurveFittingTool активизировать кнопку Data. В окнах Xdata, Ydata станут доступными составляющие cdate (X) и pop (Y) из файла census.mat;

д) произвести выбор `cdate` и `pop`, затем активизировать кнопку `CreateDataSet`;

е) активизировать кнопку `Fitting`;

ж) выбрать вид аппроксимирующей функции и активизировать кнопку `Apply`;

Результаты аппроксимации будут представлены в графической форме в окне графического вывода на панели `CurveFittingTool` (рисунок 6.1) и в численной форме в окне `Results` на панели `Tables of Fits` (рисунок 6.2).

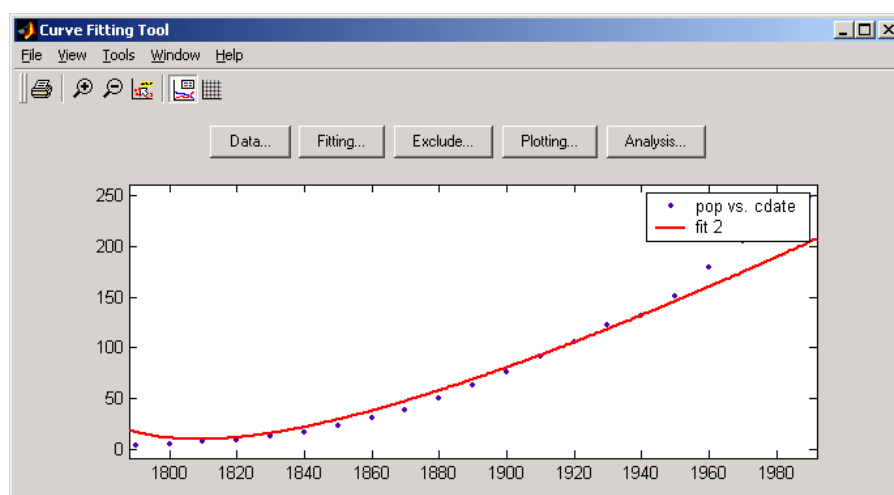


Рисунок 6.1 – Результаты аппроксимации на панели `CurveFittingTool`

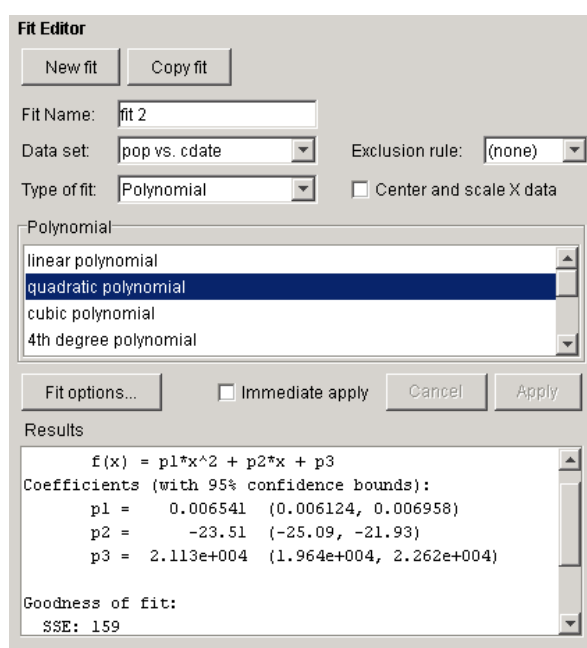


Рисунок 6.2 - Результаты аппроксимации в численной форме в окне `Results` на панели `Tables of Fits`

з) выбрать пункт меню View/Residuals/Line или View/Residuals/Scatter. В результате в окне графического вывода на панели CurveFittingTool будут одновременно выведен график ошибки интерполяции (пример на рисунке 6.3).

Окно графического вывода можно вывести на печать или выполнить в виде рисунка: File/Print to Figure.

Полученные в процессе аппроксимации результаты занести в таблицу:

Вид модели	Порядок модели	Качество подгонки (+ / -)	Диапазон погрешностей	R-square	Adjusted R-square	RMSE

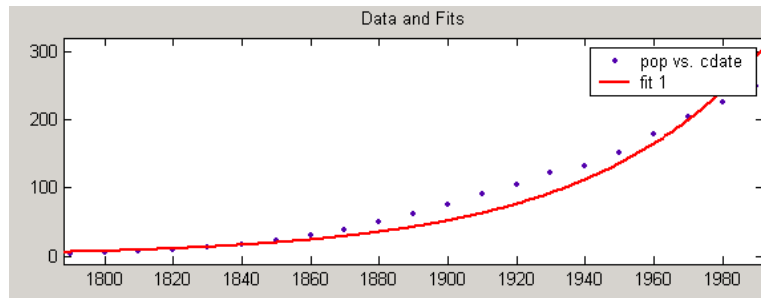
Исследовать таким образом все возможные виды моделей, порядок модели - до 5.

е) по полученным и отраженным в таблице данным выбрать три наилучшие варианта аппроксимирующих функций. Для этих вариантов получить значения график аппроксимирующей функции и график погрешности аппроксимации, аналитические выражения с численными значениями коэффициентов для аппроксимирующей зависимости.

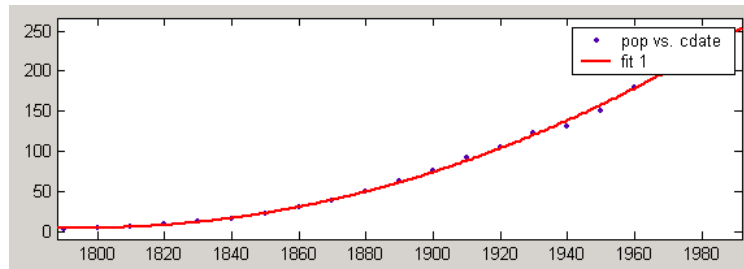
4. Оценку качества подгонки кривых и окончательный выбор аппроксимирующей зависимости производить последовательно:

1. Выбор «хороших» моделей.

1.1. Визуально, по степени согласованности графика полученной в результате подгонки аппроксимирующей кривой с отображенными в этом же окне графического вывода значениями y_i и n . Качество подгонки считается удовлетворительным, если значения y_i «вытянуты» вдоль аппроксимирующей кривой. На рисунке 6.3 приведены результаты подгонки, которые можно считать неудовлетворительными (А) и удовлетворительными (Б).



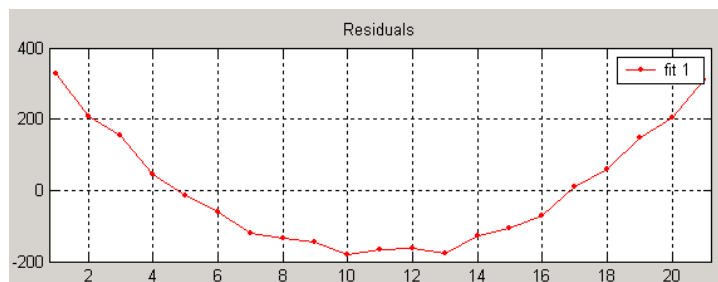
А



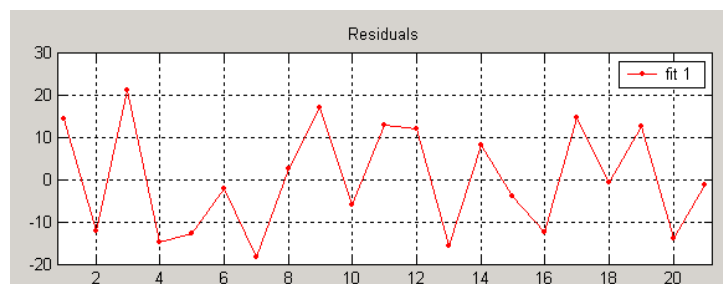
Б

Рисунок 6.3 - Результаты подгонки (неудовлетворительные (А) и удовлетворительные (Б))

1.2. Визуально, по графику разности между значениями y_i и вычисленными по аппроксимирующей функции значениями $f(x_i)$. Качество подгонки считается удовлетворительным, если график разности хорошо аппроксимируется «на глаз» функцией $y=0$. На рисунке 6.4 приведены результаты подгонки, которые можно считать неудовлетворительными (А) и удовлетворительными (Б).



А



Б

Рисунок 6.4 - Результаты подгонки (неудовлетворительные (А) и удовлетворительные (Б))

2. Выбор лучшей кривой в каждом классе аппроксимирующих зависимостей – по Adjusted R-square. Лучшей кривой в классе является та, которой соответствует максимальное значение Adjusted R-square. Если максимальное значение соответствует двум или более кривым – выбирается кривая с наиболее простым аналитическим описанием.

3. Окончательный выбор самой лучшей аппроксимирующей зависимости – выбрать самую простую модель из лучших с условием примерно тех же значений диапазона погрешностей экспериментальных данных от аппроксимирующей кривой и RMSE (Root Mean Squared Error).

Задание

1. Освоить технологию работы с программой CurveFitting пакета . MATLAB.

2. Научиться производить обоснованный выбор наилучшей аппроксимирующей зависимости из возможных.

3. В соответствии с порядком выполнения лабораторной работы сделать аппроксимацию экспериментальных данных.

Содержание отчета

1. Постановка задачи.

2. Результаты аппроксимации экспериментальных данных из файла census.mat и censusN.mat и показатели качества подгонки R-square и Adjusted R-square для всех основных классов аппроксимирующих функций (экспоненциальной, гауссиана, показательной, Фурье, полиномиальной, показательной, рациональной) в форме таблицы результатов.

3. Для трех наиболее подходящих аппроксимирующих зависимостей - график аппроксимирующей функции и график погрешности аппроксимации, аналитические выражения для аппроксимирующей зависимости.

4. Обоснование выбора наилучшей аппроксимирующей зависимости.

5. Выводы.

Контрольные вопросы

1. Сколько этапов включает в себя обработка результатов однофакторного эксперимента?
2. Какие виды функций для аппроксимации и интерполяции одномерных массивов данных предоставляет программа подгонки кривых CurveFitting?
3. Приведите формулу показательной функции.
4. Приведите формулу функции Фурье.
5. Приведите формулу рациональной функции.

Лабораторная №6.1

«Цифровая фильтрация шумов в среде MATLAB»

Цель работы: изучение методики разработки программ цифровой обработки сигналов, включающей различные способы улучшения отношения сигнал/шум (накопление, использование НЧ и ВЧ-фильтров, оптимального фильтра Колмогорова-Винера, прямого и обратного БПФ).

Теоретические сведения

Цифровая обработка сигналов решает задачи обнаружения и определения параметров информативных сигналов и изображений, искаженных шумами и помехами. Для этой цели используются различные средства:

- цифровые частотные фильтры (высокой частоты, низкой частоты, полосовые фильтры);
 - накопление (временная фильтрация);
 - сглаживание (медианные фильтры);
 - оптимальные фильтры (фильтр Колмогорова-Винера, LMS и RLS-фильтры);
 - адаптивные фильтры (функцию адаптивных фильтров могут выполнять фильтр Колмогорова-Винера, LMS и RLS-фильтры).
- фильтры для борьбы с шумами при нелинейных и нестационарных процессах (фильтр Гильберта-Хуанга).

Выбор способа борьбы с шумами должен производиться с учетом свойств и особенностей информативного сигнала и помехи. Чем в большей степени свойства сигнала и шума априори известны, тем может быть получен больший эффект от цифровой обработки. Кроме того, несмотря на обилие стандартных, доведенных до уровня готовых программ цифровой обработки, с учетом конкретных априори известных свойствах информативного сигнала и шума может оказаться полезным разработка новых методов и алгоритмов для борьбы с шумами.

Цифровой низкочастотный фильтр

Для фильтрации высокочастотного шума может быть применен фильтр низких частот (ФНЧ). Частотная характеристика ФНЧ выражается как

$$K(\omega) = \frac{1}{1 + j\omega},$$

где ω - частота среза НЧ-фильтра.

Для фильтрации сигнала нужно вычислить частотный спектр сигнала с помощью преобразования Фурье, затем перемножить частотный спектр сигнала и частотную функцию фильтра и выполнить обратное преобразование Фурье. Второй способ – вычислить импульсную переходную характеристику фильтра (реакцию на единичный импульс), затем выполнить операцию свертки входного сигнала с импульсной переходной характеристикой фильтра.

В программах обработки дискретизированных сигналов, представленных в форме числовых массивов, цикл

```
for k=1:NFFT
x(k) = A*sin(2*pi*KP1*k/NFFT);
end
```

создает KP1 периодов дискретизированного синусоидального сигнала в числовом массиве, содержащем NFFT значений, понятия частоты в этом случае отсутствует и появляется только в том случае, если задать шаг дискретности по времени. Аналогично этому, в результате выполнения быстрого преобразования Фурье

```
I=1:NFFT;
XX1=fft(x,NFFT);
```

мы получаем числовой массив, который может содержать NFFT элементов, причем информативной будет только первая половина массива, вторая будет зеркально отображать первую половину. В первой половине массива, содержащей NFFT/2 элементов будет представлен «частотный» спектр. Спектр будет отражать не частоту

сигнала, установить которую по числовому массиву, представляющему сигнал во временной области, невозможно, а количество периодов. Т.е. в приведенном выше примере «всплеск» в массиве частотной области будет в элементе с номером KP1. Таким образом, массив частотной области укажет на количество периодов сигнала во временной области.

Частотная характеристика линейного фильтра низких частот может быть вычислена следующим образом:

```
for i=1:NFFT
H(i)=1/((1+j*i/NC));
end
```

Здесь NC - полоса пропускания фильтра по уровню 0,7 амплитуды выражена в количестве отчетов спектра БПФ, пропускаемых фильтром. Остальные отсчеты в массиве частотного спектра будут ослабляться по амплитуде. Таким образом, понятие постоянной времени фильтра, равно как и полосы пропускания при дискретизированном представлении линейного фильтра отсутствует.

Ниже приведена программа фильтрации сигналов и временные диаграммы.

```
%Низкочастотный фильтр
A=20; %амплитуда сигнала
Q=5; %амплитуда шума
KP1=12;% - количество периодов первого сигнала
NFFT=1024;%количество точек расчета
Fs=1024;
for k=1:NFFT % генерация сигнала и шума
s(k) = A*sin(2*pi*KP1*k/NFFT);
q(k)=Q*(randn(size(Fs))); % шум
x(k)=s(k)+q(k); % суммирование сигнала и шума
end
figure
plot(x);
title('Зашумленный сигнал до фильтра');
Y=fft(x,NFFT)/NFFT; %БПФ сигнала с шумом
SS1=Y.*conj(Y)/Fs; %спектр мощности сигнала с шумом
```

```

figure
plot(f,2*abs(SS1(1:NFFT/2)));
title('Частотный спектр сигнала с шумом');
NC=12; %NC - полоса пропускания фильтра по уровню 0,7
амплитуды
% выражена в количестве отчетов спектра БПФ, пропускаемых
фильтром
% остальные отсчеты (в частотном спектре!) будут ослабляться
по амплитуде
for i=1:NFFT
H(i)=1/((1+j*i/NC)); %передаточная функция простого фильтра
%в частотной области
End
i=1:400;
figure
%plot(i,abs(H(1:400))); %вывод графика частотной
характеристики фильтра
semilogx(i,abs(H(1:400)));%то же, что и plot, но в
логарифмическом
%масштабе по X
title('Частотная хар-ка НЧ-фильтра');
i=1:NFFT;
XX1=fft(x,NFFT); %частотный спектр сигнала с шумом
Z=ifft(XX1.*H); %свертка зашумленного сигнала с частотной
хар-кой фильтра
i=1:NFFT;
figure
plot(i,2*Z(1:NFFT)); %вывод отфильтрованного сигнала
title('Сигнал после фильтра');
pause;
close all;

```

На рисунке 6.1.1 представлен результат работы программы.

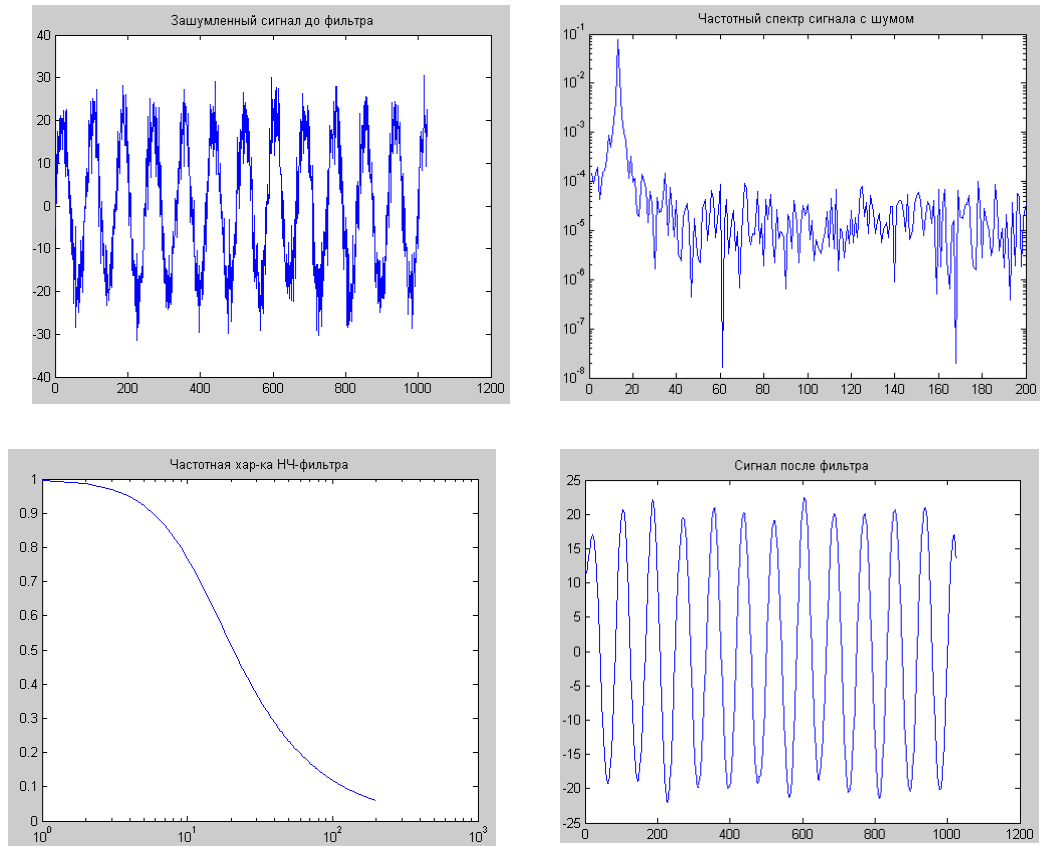


Рисунок 6.1.1 - Исходный сигнал с шумом (А), его частотный спектр (Б), частотная характеристика фильтра в полулогарифмическом масштабе (В) и сигнал после фильтра (Г)

Оптимальный фильтр Колмогорова-Винера

Фильтры низкой частоты, высокой частоты и полосовые фильтры эффективны в том случае, когда частотные спектры сигнала и шума не перекрываются.

Наилучшее разделение сигнала и шума цифровыми методами обеспечивает оптимальный фильтр Колмогорова-Винера.

Частотная характеристика фильтра Колмогорова-Винера:

$$H(\omega) = W_s(\omega) / [W_s(\omega) + W_q(\omega)],$$

где $W_s(\omega)$ и $W_q(\omega)$ - энергетические спектры (плотности мощности) сигнала и помех.

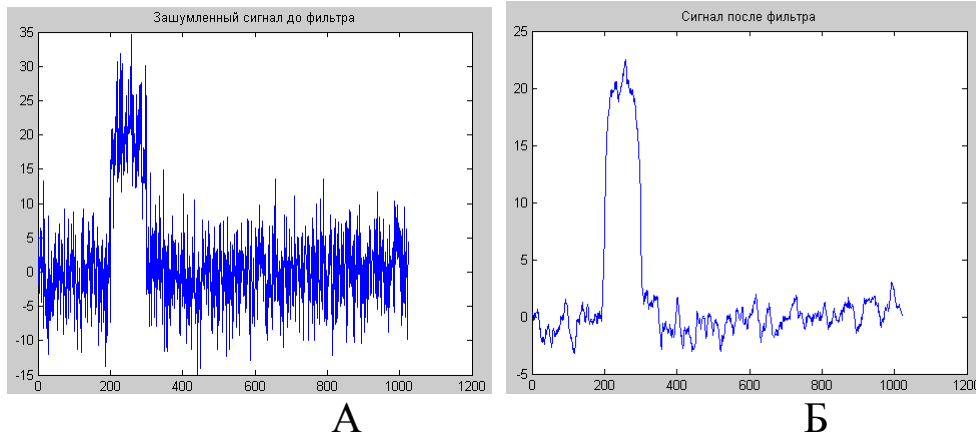


Рисунок 6.1.2 - Исходный сигнал с шумом (А) и после фильтра (Б)

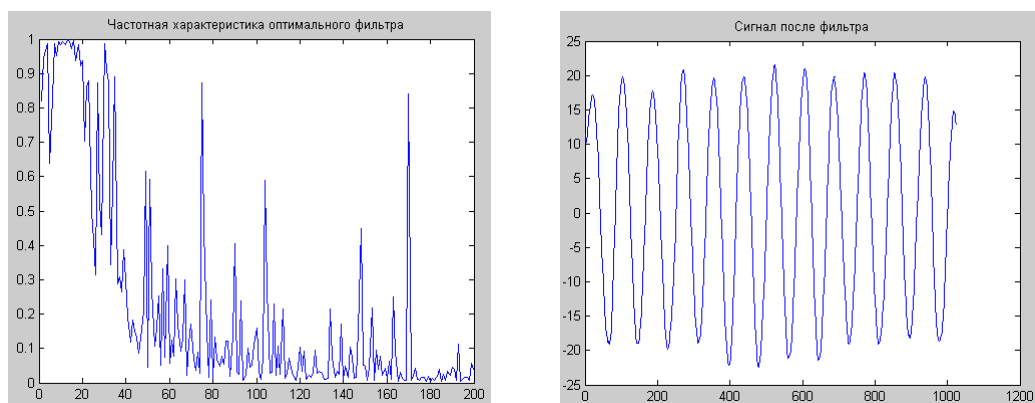


Рисунок 6.1.3 - Частотная характеристика оптимального фильтра (А) и сигнал после фильтра (Б)

Программа, реализующая оптимальный фильтр Колмогорова – Винера в среде MATLAB, приведена ниже.

```
%Программа оптимального фильтра Колмогорова - Винера
A=20; %амплитуда сигнала
Q=20; %амплитуда шума
Fs=1024;
NFFT=1024;%количество точек расчета
L=1024;
for k=1:NFFT %цикл вычисления сигнала и шума
% s1(k)=A*exp(-0.00003*(k-500)^2.0); %колоколообразный
сигнал
% s1(k)=A*sin(2*pi*12*k/1000.0)% синусоидальный сигнал
s1(k)=0;
if (k>400)&(k<600) % сигнал прямоугольной формы
s1(k)=A;
```

```

end
q(k)=Q*(randn(size(Fs))); % шум
x1(k)=s1(k)+q(k); % суммирование сигнала и шума
end
figure
plot(x1);
title('Зашумленный сигнал до фильтра');

Y=fft(s1,NFFT)/L; %БПФ сигнала без шума
f = Fs/2*linspace(0,1,NFFT/2);%вычисление шкалы для
частотной области
% figure
% plot(f,2*abs(Y(1:NFFT/2)));
SS1=Y.*conj(Y)/Fs; %спектр мощности сигнала без шума
% figure
% plot(f,2*abs(SS1(1:NFFT/2)));

Y1=fft(q,NFFT)/L; %БПФ шума
f = Fs/2*linspace(0,1,NFFT/2);
% figure
% plot(f,2*abs(Y1(1:NFFT/2)))
SS2=Y1.*conj(Y1)/Fs; %спектр мощности шума
% figure
% plot(f,2*abs(SS2(1:NFFT/2)));
for i=1:NFFT
H(i)=SS1(i)/(SS1(i)+SS2(i)); % передаточная функция
оптимального фильтра
%в частотной области
end
% i=1:40;
% figure
% plot(i,abs(H(1:40)));
i=1:NFFT;
h=ifft(H);
XX2=conv(x1,h);
figure
plot(i,XX2 (1:NFFT)); % вывод отфильтрованного сигнала
title('Сигнал после фильтра 2');
i=1:NFFT;

```



```

XX1=fft(x1,NFFT); %частотный спектр сигнала с шумом
Z=ifft(XX1.*H); %свертка зашумленного сигнала с частотной
характеристикой фильтра
figure
plot(i,Z (1:NFFT)); %вывод отфильтрованного сигнала
title('Сигнал после фильтра');

```

На рисунках 6.1.2, 6.1.3 представлен результат работы программы.

Фильтрация шумов с использованием прямого и обратного БПФ

В том случае, когда частотные спектры сигнала и шума не перекрываются, фильтрация шума может быть произведена путем выполнения БПФ, обнуления спектральных линий шума и последующего обратного БПФ.

%Программа БПФ. Сигнал содержит две частотные составляющие.

```

Fs = 1000;           % Sampling frequency
T = 1/Fs;           % Sample time
L = 1000;           % Length of signal
t = (0:L-1)*T;      % Time vector
% Sum of a 50 Hz sinusoid and a 120 Hz sinusoid
x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
y = x + 2*randn(size(t)); % Sinusoids plus noise
figure
plot(Fs*t(1:50),y(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time (milliseconds)')
%NFFT = 2^nextpow2(L); % Next power of 2 from length of y
NFFT=1024;
Y = fft(y,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2);

% Plot single-sided amplitude spectrum.
figure
plot(f,2*abs(Y(1:NFFT/2)))

```

```

title('Single-Sided Amplitude Spectrum of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')

```

На рисунке 6.1.4 представлен зашумленный сигнал, представляющий сумму двух синусоидальных сигналов разных частот (количество периодов во временном окне 50 и 120) и частотный спектр, полученный с помощью БПФ.

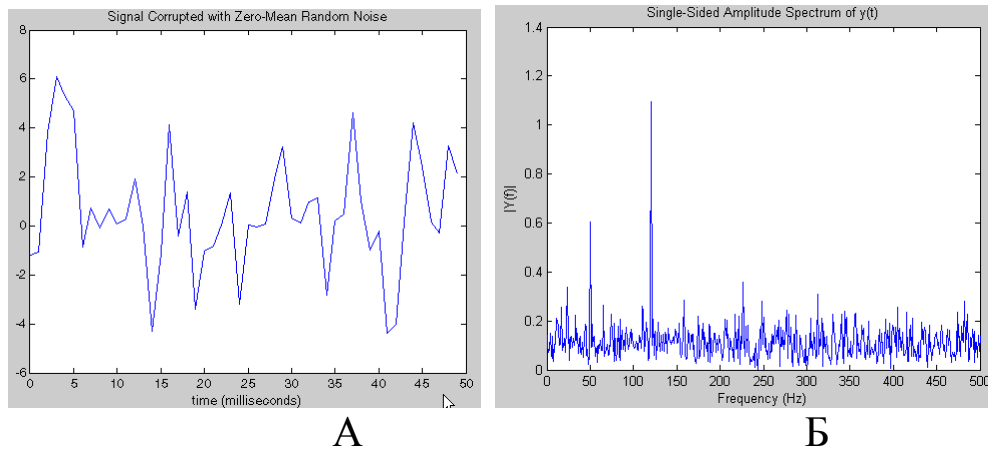


Рисунок 6.1.4 - Зашумленный сигнал, представляющий сумму двух синусоидальных сигналов разных частот (количество периодов во временном окне 50 и 120) (А) и частотный спектр, полученный с помощью БПФ (Б)

Если теперь обнулить участки спектра от 0 до 40, от 55 до 110 и от 125 до 500, а затем выполнить обратное преобразование БПФ (ifft), то получим спектр и сигнал, представленные на рисунке 6.1.5.

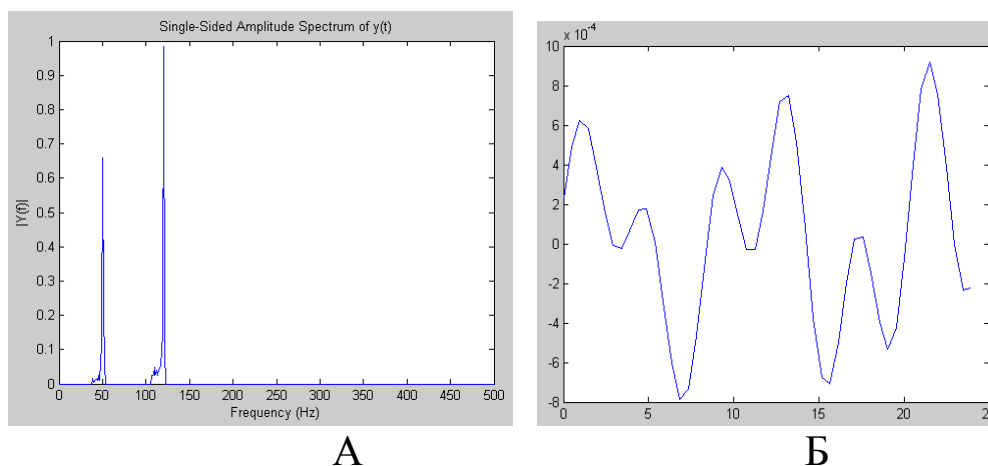


Рисунок 6.1.5 - Частотный спектр после удаления спектральных составляющих шума (А) и восстановленный с помощью обратного преобразования Фурье сигнал (Б)

Текст программы фильтрации помех с помощью прямого и обратного преобразования Фурье приведен ниже.

```

%Программа фильтрации помех с использованием БПФ-ОБПФ
Fs = 1000;           % Sampling frequency
T = 1/Fs;           % Sample time
L = 1000;           % Length of signal
t = (0:L-1)*T;      % Time vector
% Sum of a 50 Hz sinusoid and a 120 Hz sinusoid
x = 1*sin(2*pi*50*t) + sin(2*pi*120*t);
y = x + 0.2*randn(size(t)); % Синусоиды+шум
figure
plot(Fs*t(1:50),y(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time (milliseconds)')
%NFFT = 2^nextpow2(L); % Next power of 2 from length of y
NFFT=1024;
Y = fft(y,NFFT)/L;%вычисление спектра сигнала с шумом
for i=1:NFFT %"вырезание" спектральных составляющих шума
if (i>125)|(i<40)|((i>55)&(i<110))
    Y(i)=0;
end
end
f = Fs/2*linspace(0,1,NFFT/2);
figure
plot(f,2*abs(Y(1:NFFT/2)))
title('Single-Sided Amplitude Spectrum of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')

%восстановление исходного сигнала после "вырезания"
%спектральных составляющих шума
z=ifft(Y);
f = Fs/2*linspace(0,1,NFFT);
figure
plot(f(1:50),z(1:50));

pause;
clear all; %стирание всех окон графического вывода

```

Фильтр скользящего среднего

Пусть мы имеем массив N значений измеренного сигнала, представленный в цифровой форме:

$$\{f_1, f_2, \dots, f_N\}, \quad i = 1, 2, 3, \dots, N.$$

Для нахождения скользящего среднего в окрестности точки f_i берем среднее арифметическое от K предыдущих и K последующих точек, включая и f_i . Таким же образом производим обработку для всех значений i . В результате вычисляем новый массив g_i :

$$g_i = \frac{1}{2K + 1} (f_{i-K} + f_{i-K-1} + \dots + f_i + \dots + f_{i+K})$$

или

$$g_i = \frac{1}{2K + 1} \sum_{j=-K}^K f_{i+j}$$

Текст программы фильтрации помех с помощью фильтра скользящего среднего приведен ниже.

`%Фильтр скользящего среднего`

`A=20; %амплитуда сигнала`

`Q=1; %СКО шума`

`КР1=6;% - количество периодов первого сигнала`

`КР2=5;% - количество периодов второго сигнала`

`N=256;%количество точек расчета`

`for k=1:N % генерация сигнала и шума`

`s(k) = A*sin(2*pi*КР1*k/N);% + A*sin(2*pi*КР2*k/1000);`

`q(k)=Q*(randn(size(N))); % СКО шума, амплитуда равна 3Q`

`x(k)=s(k)+q(k); % суммирование сигнала и шума`

`end`

`figure`

`plot(x);`

```

hold on;
title('Зашумленный сигнал до фильтра');
S=x(1)+x(2)+x(3)+x(4)+x(5)+x(6)+x(7);
y(4)=S/7;
for i=1:N-7 %сглаживание зашумленного сигнала
    S=S-x(i)+x(i+7);
    y(i+4)=S/7;
end
i=1:N-7;
%figure;
plot(i,y(1:N-7),'r-');
title('Сигнал после фильтра');
xlabel('Номер отсчета'); % подпись по оси X
i=1:N-7;
DX(i)=x(i)-y(i);
figure
plot(i,DX(i:N-7)); %вывод погрешности отфильтрованного
сигнала
title('Погрешность отфильтрованного сигнала');
ylabel('Случайная погрешность'); % подпись по оси Y
pause;
clear all; %стирание всех окон графического вывода

```

Порядок выполнения работы

1. При исследовании эффекта улучшения отношения сигнал/шум и уменьшения погрешности обработанного сигнала в сравнении с исходным благодаря накоплению построить зависимость отношения сигнал/шум и разности исходного и обработанного сигналов от количества накоплений.

2. При исследовании эффекта улучшения отношения сигнал/шум и уменьшения погрешности обработанного сигнала в сравнении с исходным благодаря использованию фильтров (НЧ, Баттерворта, БПФ-ОБПФ) найти оптимальную полосу пропускания фильтра, при которой эффект будет наибольшим с точки зрения улучшения отношения сигнал/шум и уменьшения погрешности обработанного сигнала отдельно.

3. При исследовании эффекта улучшения отношения сигнал/шум и уменьшения погрешности обработанного сигнала в

сравнении с исходным благодаря использованию фильтра скользящего среднего (Smoothing) определить зависимость степени подавления шумов от ширины окна (при рамере окна 3, 5, 7 элементов) при различном уровне шумов и влияние параметров фильтра на изменения амплитуды и фазы сигнала на выходе фильтра.

4. При исследовании эффекта улучшения отношения сигнал/шум и уменьшения погрешности обработанного сигнала произвести сравнение эффективности исследуемых цифровых фильтров с оптимальным фильтром Колмогорова-Винера при обработке сигналов различной формы: гармонических, импульсных, колоколообразных при различных параметрах частоты и длительности.

Задание

Имеется набор экспериментальных данных в виде числового массива. Требуется спроектировать на внутреннем языке MATLAB программу цифровой обработки данных, реализующую различные способы улучшения отношения сигнал/шум:

1. Накопление.
2. НЧ-фильтр.
3. Фильтр скользящего среднего.
4. Фильтр Баттерворта.
5. Прямое и обратное БПФ.
6. Оптимальный фильтр Колмогорова-Винера.

и оценить сравнительный эффект улучшения отношения сигнал/шум и степень искажения сигнала в результате обработки.

Содержание отчета

1. Цель работы.
2. Задание к работе.
3. Таблица исходных данных.
4. Тексты программ фильтрации с комментариями.
5. Графики исходного и обработанного сигналов, график погрешности при различных параметрах обработки.
6. Выводы.

Контрольные вопросы

1. Какие задачи решает цифровая обработка сигналов?
2. Какие средства используются для обнаружения и определения параметров информативных сигналов и изображений, искаженных шумами и помехами?
3. Как выражается частотная характеристика ФНЧ?
4. Приведите алгоритм фильтрации сигнала.
5. В каких случаях эффективны ФНЧ, ФВЧ и полосовые фильтры?
6. Какой фильтр обеспечивает наилучшее разделение сигнала и шума цифровыми методами?
7. Как реализуется частотная характеристика фильтра Колмогорова-Винера?
8. Приведите алгоритм фильтрации шумов с использованием прямого и обратного БПФ?
9. Что представляет собой фильтр скользящего среднего?
10. Как уменьшается погрешность обработанного сигнала?

Лабораторная работа №6.2 **«Цифровая обработка сигналов в среде MATLAB»**

Цель работы: изучение методики разработки программ сложных видов цифровой обработки сигналов, включающей комбинацию ключевых операций (БПФ, корреляции, сплайн-аппроксимации и передискретизации).

Теоретические сведения

Одной из важнейших задач цифровой обработки зашумленных сигналов является обнаружение информативного сигнала в потоке данных, искаженных шумами и помехами, и определение его параметров. Для этого применяются различные методы, такие как временная фильтрация (накопление), оптимальная частотная фильтрация, прямое и обратное преобразование Фурье, корреляционная обработка. Каждая из этих операций позволяет выполнять преобразования исходного сигнала, например, переход сигнала из временной области в частотную или наоборот, причем при этом производится уменьшение уровня шумов в обработанном сигнале. В задачах обнаружения и определения параметров зашумленных сигналов усиление эффекта подавления шумов и увеличения точности определения параметров сигнала можно достичь, используя несколько методов цифровой обработки в комплексе. Примером может быть задача обработки эхо-сигнала спектрометра ЯМР.

Результатом БПФ дискретизированного эхо-сигнала спектрометра ЯМР определенной частоты является количество периодов сигнала во временном окне. Если частота отсчетов или интервал дискретности по времени при измерении сигнала известен, то по количеству периодов во временном окне можно установить и частоту измеряемого сигнала. Точность определения частоты в спектре входного сигнала вполне определена и зависит от количества периодов p сигнала. Если количество периодов целое, то частота с помощью БПФ находится абсолютно точно (при отсутствии зашумленности сигнала). Если же количество периодов не является целым, то появляется погрешность определения частоты. Максимальное значение погрешности равняется $1/p$. В некоторых, практически важных случаях, например, при обработке

эхо-сигналов (рисунок 6.2.1 А) импульсных спектрометров ЯМР, количество периодов анализируемого сигнала во временном окне принципиально ограничено величиной около 10. В этом случае погрешность определения частоты с помощью БПФ достигает $1/10$, т. е. 10%.

Влияние шума в регистрируемом сигнале во временной области можно значительно уменьшать за счет многократного повторения эксперимента и синхронного накопления эхо-сигналов (рисунок 6.2.1 Б).

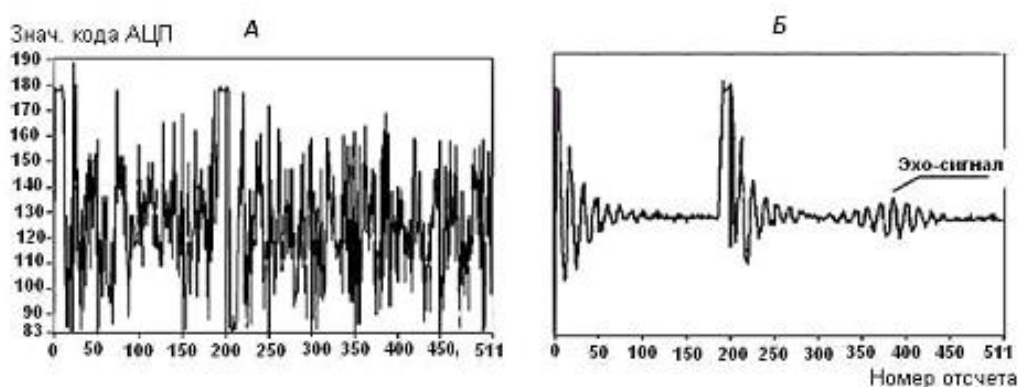


Рисунок 6.2.1 - Исходный (А) и накопленный сигнал (Б) в борате железа

Однако увеличение количества накоплений позволяет улучшать отношение сигнал/шум без искажения формы и уменьшения амплитуды накопленного эхо-сигнала лишь до некоторого предела. После этого предела накопление уже не приносит ощутимого улучшения качества. В частности, при большом времени накопления на эхо-сигнал начинают влиять постепенные изменения параметров приборов, входящих в состав импульсного спектрометра ЯМР. При ограничении времени проведения анализа веществ количество возможных накоплений сигнала должно быть ограничено или вообще должно отсутствовать. Проблема обработки эхо-сигналов в условиях значительных шумов возникает еще и в том случае, когда имеется необходимость исследования веществ малой концентрации. Поэтому задача выполнения частотного анализа зашумленных эхо-сигналов актуальна.

Непосредственное использование БПФ для зашумленного сигнала не позволяет получить точное значение количества периодов во временном окне и частоты измеряемого сигнала в том

случае, когда количество периодов не является целым, когда анализируемый сигнал занимает только часть временной области, модулирован по амплитуде и зашумлен. В практически важных случаях частотного анализа сигналов спектрометров ЯМР форма и начальная фаза эхо-сигналов известны. Это дает возможность создать эталонные сигналы, соответствующие ожидаемому эхо-сигналу по форме и начальной фазе, и произвести их корреляционное сравнение. Коэффициент корреляции эхо-сигнала с эталонным сигналом будет равен единице, если частоты эхо-сигнала и эталонного сигнала равны и эхо-сигнал не зашумлен. Поэтому при отсутствии шумов найти частоту эхо-сигнала можно производя корреляционное сравнение с эталонными сигналами, частоту эталонных сигналов подбирать до выполнения условия, когда коэффициент корреляции будет равен единице. Однако коэффициент корреляции уменьшается как при разнице частот эхо и эталонного сигналов, так и при совпадении частот, но из-за наличия шума. Поэтому таким способом определить частоту зашумленного сигнала невозможно.

Усовершенствование алгоритма определения частоты сигнала достигается за счет сочетания положительных качеств корреляционного подхода и БПФ с целью повышения точности определения частоты дискретизированного сигнала в условиях, когда обрабатываемый сигнал во временной области зашумлен, количество периодов не является целым, сигнал модулирован по амплитуде, количество периодов сигнала мало.

Идея предлагаемого алгоритма заключается в том, что в небольшой окрестности от предполагаемой частоты сигнала или количества периодов (приближенное значение частоты сигнала может быть найдено с помощью быстрого преобразования Фурье) вычисляются коэффициенты корреляции с несколькими эталонными сигналами в некоторой окрестности от приближенного значения, затем с помощью сплайн-интерполяции и передискретизации строится функция, выражающая зависимость коэффициента корреляции от частоты эталонов и находится максимум этой функции, по положению максимума определяется частота эталонного сигнала. Функция, построенная таким образом, имеет вид параболы с явно выраженным максимумом (рисунок 6.2.2) как в случае незашумленного так и зашумленного сигнала, что и позволяет определить частоту эхо-сигнала более точно, чем это

позволяет сделать БПФ. При наличии шума форма функции сохраняется, уменьшается лишь абсолютное значение максимума.

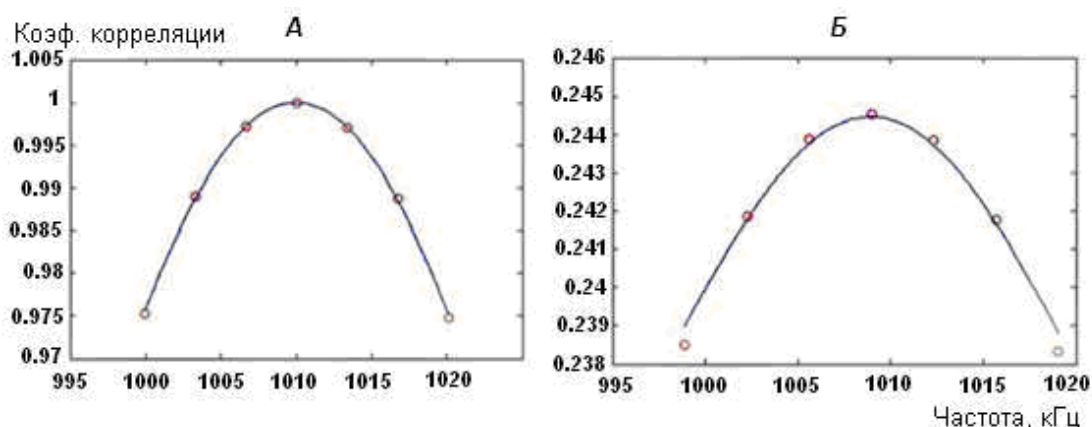


Рисунок 6.2.2 - Зависимость коэффициента корреляции от частоты при отсутствии шума (А) и при отношении сигнал/шум 1/3 (Б)

Точное значение частоты равно 1010. Аппроксимирующие кривые построены с помощью функции сплайн-аппроксимации `splaps` в MATLAB.

Точность определения частоты сигнала тем выше, чем ближе начальное приближение к истинной частоте. Поэтому применяется итерационное вычисление, на каждом этапе итерации в качестве начального приближения используется уточненное значение частоты, полученное на предыдущем этапе. В качестве первого приближения берется частота, определенная с помощью БПФ.

Количество итераций для расчета частоты с заданной точностью с помощью описанного алгоритма зависит от того, насколько близко к искомой частоте будет находиться начальное приближение.

Порядок выполнения работы

1. Использовать в качестве основы программу, приведенную в приложении 4.

2. Расчеты сделать по результатам 1-й и 2-й итерации отдельно. В качестве приближенного значения количества периодов на 1-й итерации взять результат, полученный с помощью БПФ, для 2-й итерации – уточненное в результате 1-й итерации значение

количества периодов, величину окрестности вблизи уточненного значения количества периодов, в которой создаются эталонные сигналы (параметр `prosch` в программе) уменьшить в m раз, $m \leq 10$.

3. При исследовании эффекта увеличения точности определения количества периодов и частоты сигнала по сравнению с БПФ за счет дополнительных операций цифровой обработки произвести измерения количества периодов и частоты сигнала в диапазоне количества периодов от K до $K+1$ с шагом 0.05 и вычислить максимальную, среднюю и среднеквадратическую погрешности определения количества периодов и частоты сигнала на этом интервале.

4. При исследовании эффекта увеличения точности определения количества периодов и частоты сигнала по сравнению с БПФ за счет дополнительных операций цифровой обработки при зашумленности сигнала произвести измерения количества периодов и частоты сигнала в диапазоне количества периодов от K до $K+1$ с шагом 0.05 при значениях амплитуды шума от 0 до 1 с шагом 0.1 и вычислить максимальную, среднюю и среднеквадратическую погрешности определения количества периодов и частоты сигнала на этом интервале для каждого уровня шума.

5. При исследовании зависимости погрешности определения количества периодов и частоты сигнала от количества периодов сигнала с помощью БПФ и комбинированного способа выполнить п.2 и 3 при значениях $K=2, 3, \dots, 10$, и построить графики зависимости максимальной, средней и среднеквадратической погрешностей от количества периодов сигнала.

Задание

1. Имеется набор экспериментальных данных в виде числового массива. Требуется спроектировать на внутреннем языке MATLAB программу цифровой обработки данных, реализующую точное определение количества периодов сигнала и частоту во временной области с использованием нескольких ключевых операций ЦОС: БПФ, корреляции, сплайн-аппроксимации и передискретизации.

2. Исследовать эффективность использования комбинированного использования нескольких операций цифровой обработки и итерационного процесса для определения количества

периодов и частоты сигнала по сравнению с использованием для решения этой задачи быстрого преобразования Фурье. Исследование провести для различных условий:

- Для различного количества итераций.
- При целом и нецелом количестве периодов сигнала, меньшем 10.
- При различном уровне зашумленности сигнала

Содержание отчета

1. Цель работы.
2. Задание к работе.
3. Текст разработанной программы.
4. Графики зависимости максимальной, средней и среднеквадратической погрешностей при различном уровне шума от количества периодов сигнала для БПФ и исследуемого способа комбинированной обработки.
5. Выводы.

Контрольные вопросы

1. Какие методы применяются для цифровой обработки зашумленных сигналов?
2. Как можно значительно уменьшить влияние шума в регистрируемом сигнале?
3. Перечислите алгоритм определения частоты сигнала.
4. Перечислите алгоритм цифровой обработки сигнала в среде MATLAB.
5. Как осуществляется генерация модельного сигнала в системе MATLAB?

Лабораторная работа №6.3 **«Работа с речевыми данными в среде MATLAB»**

Цель работы: освоить навыки записи и считывания данных из речевого сигнала в системе MATLAB; исследовать особенности мужского и женского голоса.

Теоретические сведения

Характеристики голоса

Разложим голос на составляющие, основными параметрами являются: частота, сила, длительность и тембр, которые, как величины, можно анализировать и по отдельности. В действительности, однако, подобный анализ не представляет собой реального выражения голоса, поскольку эти качества образуют единый неделимый комплекс.

Высота издаваемого звука зависит от числа колебаний голосовых складок в 1 секунду. Голосовые складки способны приходить в колебательные движения не только целиком, всей своей массой, но и отдельными участками. Только этим можно объяснить то, что одни и те же голосовые складки могут колебаться с различной частотой: примерно от 80 до 10 000 колебаний в секунду и даже больше.

Тоновый диапазон человеческого голоса представлен последовательностью тонов, которые могут быть произведены голосовым аппаратом в пределах границ между самым низким и самым высоким звуками. Человеческий голос обычно включает в себя тоны от 64 до 1300 герц. В двух формах проявления человеческого голоса - пении и разговоре - качества голоса представлены несколько различно. Разговорный голос составляет лишь 1/10 от общего диапазона голоса. Тоновый охват певческого голоса значительно шире разговорного и зависит от вокального образования. Сила подаваемого звука определяется интенсивностью напряжения голосовых складок и величиной давления воздуха в подвязочном пространстве. И тот и другой процесс регулируются центральной нервной системой. Контроль осуществляется с помощью слуха. Если же взаимоотношения между этими процессами нарушаются, например, при крике ужаса, то

превалирование давления внутри трахеи вызывает звук, который характеризуется отсутствием чистой тональности.

Ниже представлен диапазон человеческого голоса (в Герцах):

Бас - 75-300.

Баритон - 100-400 - Тенор - 120-500.

Контральто - 170-780.

Меццо-сопрано - 200-900.

Сопрано - 230-1000.

Колоратурное сопрано - 260-1400.

Для сравнения приведен частотный диапазон музыкальных инструментов:

Контрабас - 40-300.

Виолончель - 65-880.

Альт - 130-1240.

Скрипка - 210-2800.

Фагот - 60-630.

Кларнет - 140-1980.

Гобой - 230-1480.

Флейта - 240-2300.

Туба - 45-320.

Тромбон - 80-500.

Валторны - 60-740.

Труба - 160-990.

Таким образом, человеческий голос имеет диапазон звучания от 75 до 110 Герц, который так или иначе перекрывает (заглушается, смешивается) с любым музыкальным инструментом (оптимальная точка - 300 Герц).

Также необходимо учитывать силу звучания (динамический диапазон) данных инструментов.

Динамический диапазон гитары составляет 15 дБ; органа - 35 дБ; рояля - 45 дБ; женский голос - 20-35 дБ; мужской голос - 20-45 дБ, эстрадного оркестра-45-55 дБ, симфонический оркестр 60-75 дБ.

Сообщение, передаваемое с помощью речевого сигнала, - дискретно, т.е. может быть представлено в виде последовательности символов из конечного их числа. Звуковые символы, из которых составлен речевой сигнал, называются фонемами.

Речь с физической точки зрения состоит из последовательности звуков с паузами между их группами. Схема речеобразования у человека представлена на рисунке 6.3.1.



Рисунка 6.3.1 - Схема речеобразования у человека

При нормальном темпе речи паузы появляются между отрывками фраз. Как правило, слова произносятся слитно, хотя слушающий воспринимает слова по отдельности. При замедленном темпе речи, например, при диктовке, паузы могут делаться между словами и даже их частями. Предлоги, союзы звучат всегда слитно с последующим словом.

Частотный диапазон речи находится в пределах 70 - 1400 Гц.

Один и тот же звук речи разные люди произносят по-разному. Произношение звуков речи зависит от ударения, соседних звуков и т.п. Но при всем многообразии в их произношении звуки являются физическими реализациями (произнесением) ограниченного числа обобщенных звуков речи (фонем). Фонема - это то, что человек должен произнести, а звук - то, что человек фактически произносит. Фонема по отношению к звуку речи играет ту же роль, что и образцовая буква по отношению к ее рукописной форме в конкретном написании.

Звуки речи делятся на звонкие и глухие. Звонкие звуки образуются с участием голосовых связок, в этом случае находящихся в напряженном состоянии. Под напором воздуха, идущего из легких, они периодически раздвигаются, в результате чего создается прерывистый поток воздуха. Импульсы потока воздуха, создаваемые голосовыми связками, с достаточной точностью могут считаться периодическими. Соответствующий период повторения импульсов называют периодом основного тона голоса T_0 - а обратную величину $1/T_0$ - частотой основного тона. Если связки тонкие и сильно напряжены, то период получается коротким и частота основного тона высокой; для толстых, слабо напряженных связок - низкой. Частота основного тона для всех голосов лежит в пределах 70 - 450 Гц. При произнесении речи она непрерывно изменяется в соответствии с ударением,

подчеркиванием звуков и слов, а также с проявлением эмоций (вопрос, восклицание, удивление и т.д.). Изменение частоты основного тона называется интонацией. У каждого человека свой диапазон изменения основного тона (обычно он бывает немногим более октавы) и своя интонация. Последняя имеет большое значение для узнавания говорящего.

Основной тон, интонация, устный почерк и тембр голоса служат для опознавания человека, и степень достоверности такая же высокая, как по отпечаткам пальцев. Импульсы основного тона имеют пилообразную форму, и поэтому при их периодическом повторении получается дискретный спектр с большим числом гармоник (до 40), частоты которых кратны частоте основного тона. Огибающая спектра основного тона имеет спад в сторону высоких частот с крутизной около 6 дБ/окт, поэтому для мужского голоса уровень составляющих на частоте 3000 Гц ниже их уровня на частоте 100 Гц примерно на 30 дБ. При произнесении глухих звуков связки находятся в расслабленном состоянии, поток воздуха из легких свободно проходит в полость рта. Встречая на своем пути различные преграды в виде языка, зубов, губ, он образует завихрения, создающие шум со сплошным спектром.

Согласные по способу образования делятся на сонорные (л, ль, р, рь, м, мь, и, нь, й), щелевые (ж, з, зь, в, вь, ш, с, сь, ф, фь, х, хь), взрывные (б, бь, д, дь, г, гь, и, иь, т, ть, к, кь) и аффрикаты (ц, ч - комбинация глухих взрывных и щелевых). Гласных фонем всего шесть: а, о, у, э, и, ы (гласные е, я, ё, ю - составные из и краткого или мягкого знака и гласных э, а, о, у).

Звонкие звуки речи, особенно гласные, имеют высокий уровень интенсивности, глухие - самый низкий. Громкость речи непрерывно изменяется, особенно резко при произнесении взрывных звуков. Динамический диапазон уровней речи находится в пределах 35 - 45 дБ. Гласные звуки речи имеют в среднем длительность около 0,15 с, согласные - около 0,08 (звук и - около 30 мс).

Звуки речи неодинаково информативны. Так, гласные звуки содержат малую информацию о смысле речи, а глухие согласные наиболее информативны (например, в слове «посылка» последовательность «о, ы, а» ничего не говорит, а «п, с, лк» дает почти однозначный ответ о смысле). Поэтому разборчивость речи снижается при действии шумов, в первую очередь из-за маскировки глухих звуков.

Известно, что для передачи одного и того же сообщения по телеграфу и по речевому тракту требуется различная пропускная способность. Для телеграфного сообщения достаточна пропускная способность не более 100 бит/с, а для речевого - около 100000 бит/с (полоса равна 7000 Гц, динамический диапазон 42 дБ, т.е. требуется семизначный код, откуда имеем $2 \cdot 7000 \cdot 7 = 98\,000$ бит/с), т.е. в 100 раз большая.

Образование звуков речи происходит путем подачи команд к мускулам артикуляционных органов речи от речевого центра мозга. Общий поток сообщений от него составляет в среднем не более 100 бит/с. Вся остальная информация в речевом сигнале называется сопутствующей.

Запись и считывание данных из речевого сигнала в MATLAB

В MATLAB предусмотрены средства для воспроизведения и записи звука (речи), а также для работы со звуковыми файлами формата wav.

Чтение wav-файлов. Для считывания wav-файлов в MATLAB используется функция wavread. В простейшем случае она может быть использована следующим образом:

$y = \text{wavread}(\text{'filename'})$, где 'filename' - имя звукового файла (расширение wav указывать не обязательно). В имя файла необходимо включить полный путь, за исключением тех случаев, когда файл находится в текущем (для MATLAB) каталоге или в одном из каталогов, входящих в список поиска MATLAB. Другой способ, не требующий указания имени файла, - полный путь, который заключается в определении местонахождения файла на жестком диске с помощью меню MATLAB.

В результате вызова функции в переменную y будет помещено все содержимое указанного файла. Строки матрицы y соответствуют отсчетам сигнала, столбцы - каналам, которых в wav-файле может быть один (моно - канал) или два (стереоканал).

Помимо отсчетов сигнала в wav-файлах хранится и служебная информация, которая содержит следующие параметры:

- частоту дискретизации, для определения которой в указанную функцию необходимо включить второй выходной параметр:

$$[y, Fs] = \text{wavread}(\text{'filename'}),$$

где F_s - частота дискретизации, Гц;

- число бит на отсчет, для определения которого необходимо добавить еще один выходной параметр:

$$[y, F_s, \text{bils}] = \text{wavread}('filename');$$

- число отсчетов и каналов записи. Для получения данной информации необходимо вызвать функцию `wavread` с двумя входными параметрами: именем файла и текстовой строкой 'size': `wavesize=wavread('filename*', 'size')`.

При вызове такой функции из wav-файла извлекается служебная информация, которая возвращается в виде двухэлементного вектор-строки, первый элемент которого содержит число отсчетов, второй - число каналов;

- продолжительность звучания сигнала (в секундах), которую можно определить следующим образом:

`wavesize(1) / F_s`, где 1 указывает на первый параметр вектора `wavesize`.

Имеются и возможности считывания данных из wav-файла не целиком, а отдельными фрагментами. Для этого используется второй входной параметр функции `wavread`. Если этот параметр является числом, будет считано соответствующее количество отсчетов, начиная с первого:

$$y = \text{wavread}('filename', N).$$

Если нужный фрагмент расположен не в начале файла, придется указать его начало и конец:

$$y = \text{wavread}('filename', [n1, n2]).$$

В результате в переменную y будут считаны отсчеты с номерами от n_1 до n_2 включительно (нумерация отсчетов начинается с единицы).

Чтобы узнать объем памяти (в килобайтах), требуемый в MATLAB для хранения записи, необходимо использовать следующую функцию:

`prod (wavesize)*8/1024.`

Для просмотра речевого (звукового) сигнала выведем его в виде графика с помощью следующей функции: `plot (y)`.

Если необходимо вывести график по каналам стереозаписи, то применяют следующие функции:

`subplot (2, 1, 1); plot (:, 1); subplot (2, 1, 2); plot (:, 2)` или просто `plot(y)`.

Если сигнал имеет большую длину, то можно использовать следующую функцию (фрагменты выводятся друг под другом):

`strips (x, N),`

где `x` - вектор отсчетов сигнала (двумерный массив не допускается), `N` - число отсчетов в каждом фрагменте (этот параметр можно опустить, по умолчанию размер фрагмента составляет 200 отсчетов).

Запись wav-файлов. Для записи вектора (или матрицы) на диск в виде wav-файла используется функция `wavwrite'`.

`wavwrite (y, Fs, N, 'filename')`, где `y` - записываемые данные, `Fs` - частота дискретизации, Гц, `N` - число бит на отсчет (8 или 16), `'filename'` - имя создаваемого файла. Параметры `Fs` и `N` можно опускать, при этом используются значения по умолчанию: `Fs = 8 000` Гц, `N= 16`.

Записываемые данные должны быть вещественными и лежать в диапазоне от -1 до 1. Значения, выходящие из этого диапазона, будут обрезаны и сделаны равными.

Воспроизведение звуковых файлов. Помимо работы с wav-файлами можно воспроизводить вектор и матрицу в звуковом виде с использованием следующих функций:

`sound`, синтаксис которой записывается следующим образом:

`sound (y, Fs, bits),`

где `y` - вектор или двухстолбцовая матрица сигнала, `Fs` - частота дискретизации, Гц, `bits` - число бит на отсчет (8 или 16).

Параметры `Fs` и `bits` можно опускать, при этом их значения будут приниматься по умолчанию.

Выходных параметров у функции нет. После вызова она передает вектор y звуковой карте для воспроизведения и сразу же, не дожидаясь окончания звука, возвращает управление MATLAB;

wavplay, синтаксис которой имеет следующий вид:

wavplay (y , F_s , 'mode'), где параметр mode управляет режимом воспроизведения, который может принимать два значения:

'sync' - синхронный режим, означающий что функция вернет управление интерпретатору MATLAB только после окончания звука;

'async' - асинхронный режим, при котором функция передает данные для воспроизведения звуковым драйверам Windows и сразу же возвращает управление системе MATLAB. не дожидаясь окончания звука.

Параметры F_s и mode можно опускать, при этом их значения принимаются по умолчанию: $F_s = 11025$ Гц и 'mode' = 'async'.

Запись звука (речи). Функция wavrecord позволяет записать звук в переменную MATLAB с помощью звуковой карты компьютера:

$$y = \text{wavrecord}(n, F_s, \text{ch}, \text{'dtype'}),$$

где n - число записываемых отсчетов, F_s - частота дискретизации, Гц. ch - число каналов записи, 'dtype' - тип записываемых данных.

Возвращаемый результату - матрица, каждый столбец которой соответствует одному каналу записи. При стереозаписи первый столбец - левый канал, второй - правый канал.

Для параметра dtype возможны следующие значения:

'double' - 16-битная запись, данные масштабируются к диапазону от -1 до 1 и представляются в восьмибайтовом формате с плавающей запятой;

'single' - 16-битная запись, данные масштабируются к диапазону -1...1 и представляются в четырехбайтовом формате с плавающей запятой;

'uint16' - 16-битная запись, данные представляются в двухбайтовом целочисленном формате (диапазон от -32 768 до 32 767);

'uint8' - 8-битная запись, данные представляются в однобайтовом беззнаковом целочисленном формате (диапазон от 0 до 255, нулевому напряжению на входе соответствует значение

«128»).

Входные параметры F_s , ch , $dtype$ можно опускать, при этом их значения будут приниматься по умолчанию: $F_s = 11\,025$ Гц, $ch = 1$, $dtype = 'double'$.

Спектрограмма

Если спектр сигнала меняется во времени, то для оценки спектра целесообразно использовать спектрограмму сигнала. Спектрограммой (`spectrogram`) сигнала называется его мгновенный спектр, зависящий от времени. Для вычисления спектрограммы вектор сигнала разбивается на сегменты (в общем случае с перекрытием). Для каждого сегмента вычисляется спектр с помощью функции `fft`. Набор спектров всех сегментов и образует спектрограмму. Для вычисления спектрограммы служит функция `spectrogram`.

Синтаксис вызова функции:

$$[S, F, T] = \text{spectrogram}(x, \text{window}, \text{overlap} - \text{lap}, \text{nfft}, F_s),$$

где x - вектор сигнала; `window` - вектор весовой функции (если вместо вектора используется целое число, то используется весовая функция по умолчанию - функция Хэмминга соответствующей длины); `overlap` - величина перекрытия соседних сегментов сигнала; `nfft` - число точек преобразования Фурье; F_s - частота дискретизации. S - матрица, каждая колонка которой содержит $(\text{nfft}/2+1)$ отсчетов спектра для данного момента времени (если `nfft` - нечетное число, количество отсчетов равно $(\text{nfft}+1)/2$). Число колонок $k = \text{nx}((\text{nx} - \text{overlap}) / (\text{length}(\text{window}) - \text{overlap}))$, где nx - длина вектора сигнала. Параметр F - вектор частот, T - вектор моментов времени, его длина равна k .

Если выходные параметры функции не указываются (`spectrogram(x, window, overlap, nfft, F_s)`), то строится трехмерный график спектральной плотности мощности в координатах: время, частота, уровень.

Обязательным входным параметром функции является вектор значений сигнала x , остальные параметры имеют значения по умолчанию, которые используются, если в качестве параметра указана пустая матрица (`[]`) или если несколько последних

параметров при вызове опущены.

```
[V, fs, b]=wavread('c:\1\female1f.wav'); spectrogram(V, 256, 128, [], fs, 'yaxis');
```

(рисунок 6.3.2).

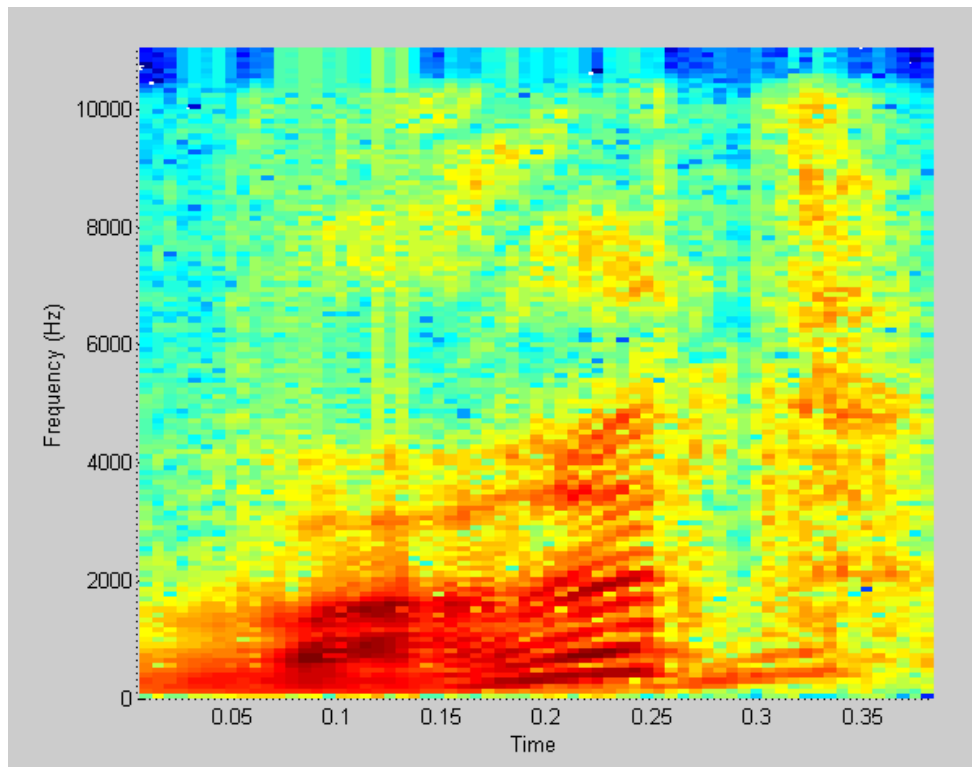


Рисунок 6.3.2 - Спектрограмма файла female1f.wav

Здесь во второй строке примера для параметра `nfft` используется значение по умолчанию - максимальное из двух чисел: 256 и 2^k . Значение k таково, что выполняется условие $2^k > \text{window}$.

Операции над звуковыми файлами

Система MATLAB позволяет читать и записывать файлы формата `wav`. Чтение осуществляется с помощью функции `wavread`, а запись - с помощью функции `wavwrite`.

```
[V, f, b] = wavread (*.wav');
```

Звуковая информация из файла считывается в матрицу `V`, состоящую из двух столбцов, в скалярную переменную `f` считывается значение частоты дискретизации, а в переменную `b` - число двоичных разрядов.

Добавим к матрице `V` белый шум.

```
[m, n] = size(V); VI = randn (m, n); s = 0.05; V2 = V + 5* VI;
```

Здесь в первой строке определяется число строк и столбцов матрицы V . Вторая строка с помощью функции `randn (m, n)` создает матрицу VI , которая содержит случайные числа, распределенные по закону Гаусса (по нормальному закону), и которая имеет тот же размер, что и матрица V . Матрица VI умножается на коэффициент s и складывается с матрицей V , образуя матрицу $V2$ зашумленного звука. Функция `sound` осуществляет воспроизведение звука, получая в качестве аргумента вещественный вектор или матрицу размерами $N \times 2$ (для стереозвука), содержащие последовательности измерений громкости звука.

```
sound (V, f, b) sound (V2, f, b)
```

Значения элементов матрицы отсчетов должны быть ограничены диапазоном от -1.0 до $+1.0$. Вне этого диапазона значения матрицы ограничиваются (имеет место клиппированный звук). Аналогичная функция `soundsc` перед воспроизведением звука обеспечивает автоматическое масштабирование значений матрицы до диапазона: $-1 \dots +1$ - клиппирования звука не происходит.

```
soundsc (V, f, b) soundsc (V2, f, b)
```

Чтобы сохранить результаты экспериментов со звуком в звуковом файле, следует применить функцию `wavwrite`. В данном случае результаты эксперимента с зашумлением сохраним в файле 'Sound1.wav':

```
wavwrite (V2, f, b, 'Sound1.wav');
```

Здесь первым аргументом является матрица (для монофонического звука - это вектор) звуковых отсчетов, вторым - частота дискретизации, третьим - разрядность отсчетов, а последним - имя файла. Если не указывать пути к файлу, то он будет записан в текущий рабочий каталог пакета MATLAB.

Исследование мужского и женского голосов

Для примера возьмем два образца женского и два образца мужского голосов. В пакете MATLAB построим их спектрограммы, амплитудный и фазовый спектры (рисунки 6.3.3 – 6.3.10).

Амплитудный спектр строим следующим образом:

```
[s, fs] = wavread (c:\1\female1f.wav);
spec = abs (fft(s));
spec = spec (1:end/2);
(freq, spec) = (fs/2)*(1:length(spec))/length(spec);
xlabel ('Frequency: Hz');
ylabel ('Amplitude spectrum');
```

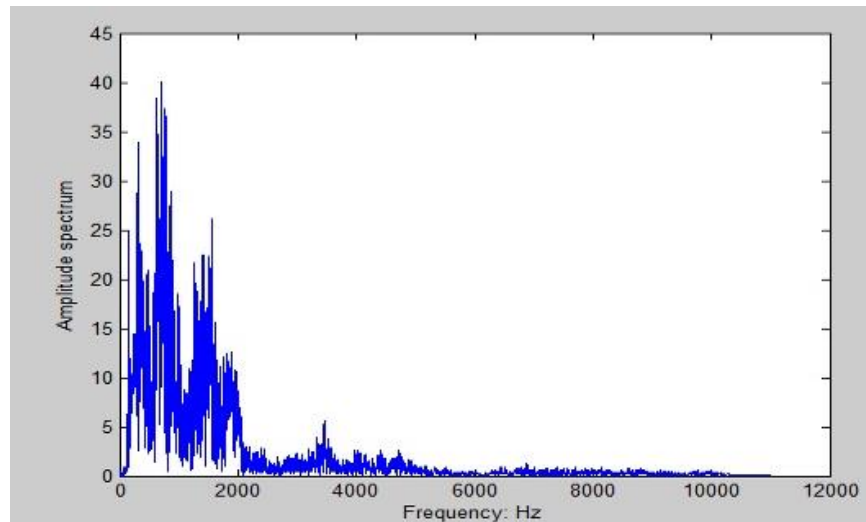


Рисунок 6.3.3 - Амплитудный спектр female1f.wav

Фазовый спектр

```
[s, fs] = wavread (c:\1\female1f.wav);
spec = phase (fft(s));
spec = spec (1:end/2);
(freq, spec) = (fs/2)*(1:length(spec))/length(spec);
xlabel ('Frequency: Hz');
ylabel ('Phase spectrum');
```

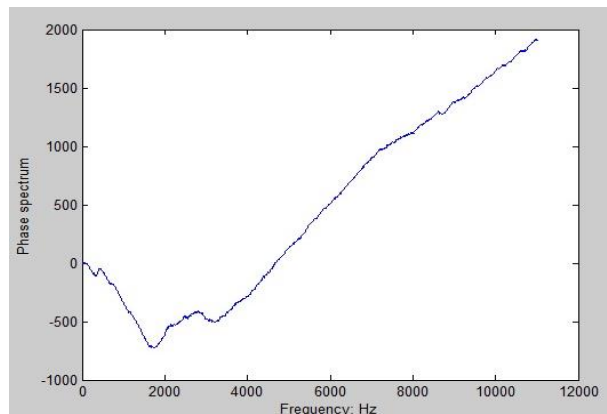


Рисунок 6.3.4 - Фазовый спектр female1f.wav

Аналогичные операции проделываем с остальными образцами голоса.

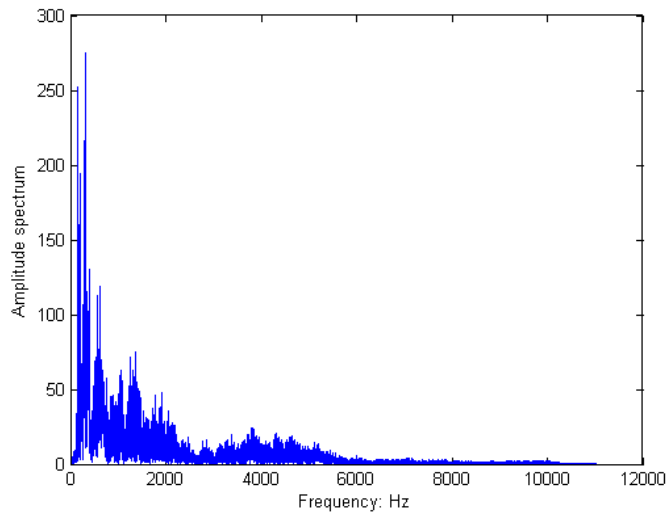


Рисунок 6.3.5 - Амплитудный спектр female2a.wav

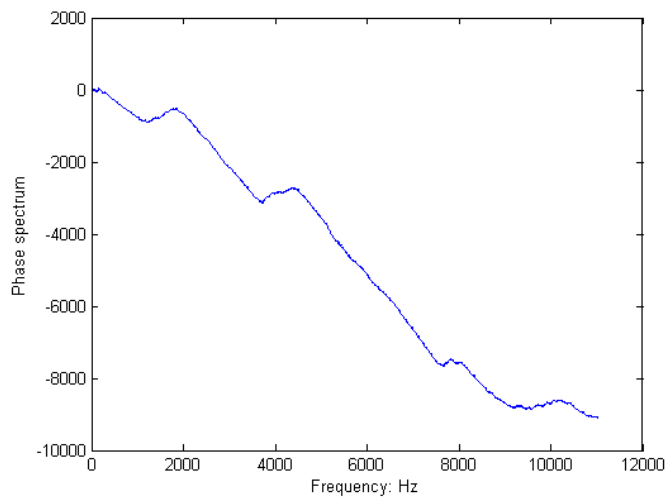


Рисунок 6.3.6 - Фазовый спектр female2a.wav

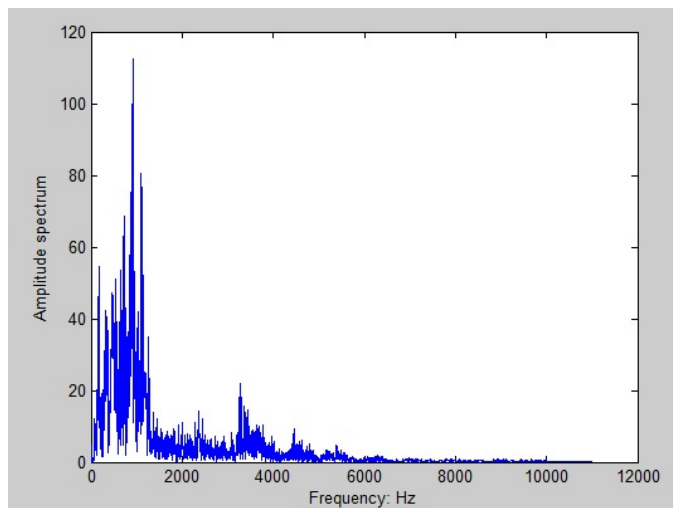


Рисунок 6.3.7 - Амплитудный спектр male1f.wav

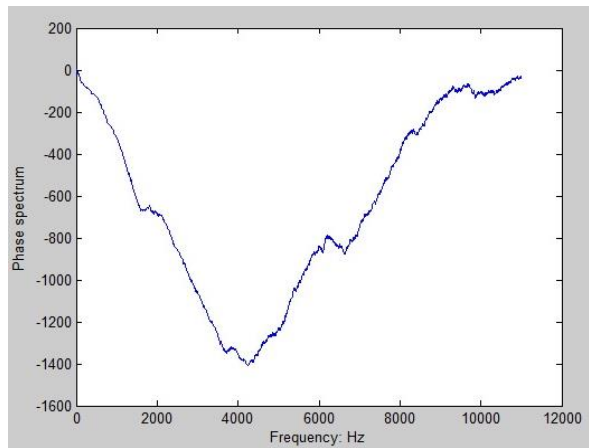


Рисунок 6.3.8 - Фазовый спектр male1f.wav

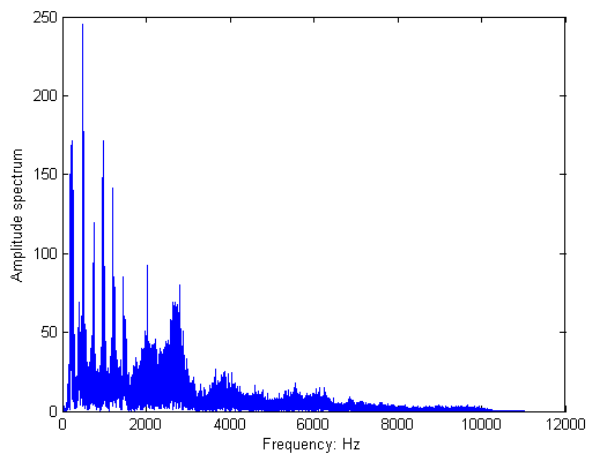


Рисунок 6.3.9 - Амплитудный спектр male2a.wav

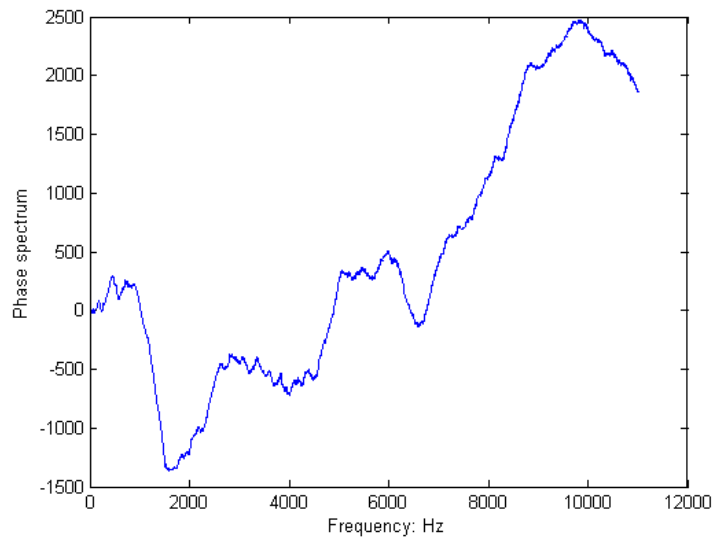


Рисунок 6.3.10 - Фазовый спектр male2a.wav

Построим спектрограммы голосов (рисунки 6.3.11-6.3.14)
`[V, fs, b]=wavread('c:\1\female1f.wav'); spectrogram(V,
 256,128, [], fs, 'yaxis');`

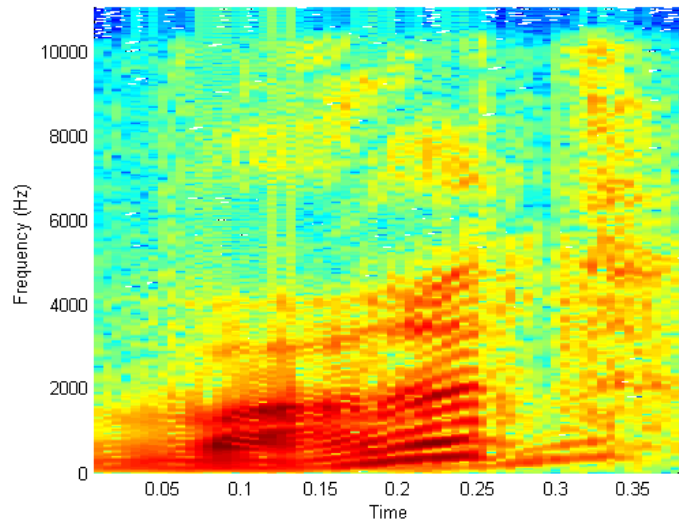


Рисунок 6.3.11 - Спектрограмма female1f.wav

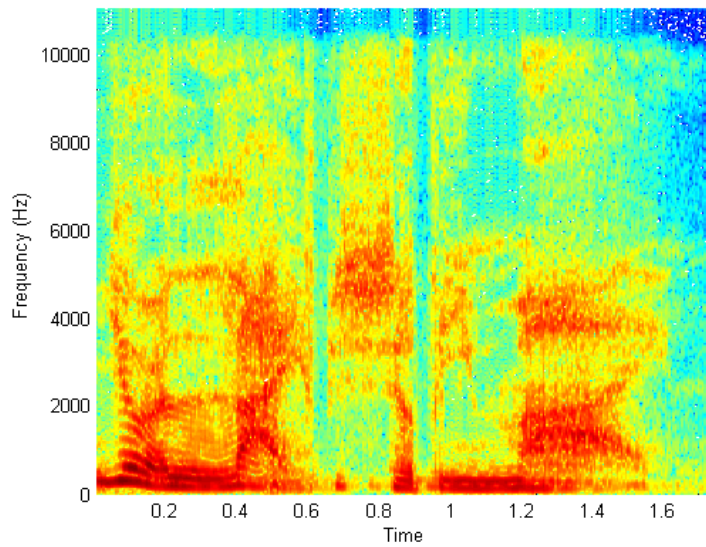


Рисунок 6.3.12 - Спектрограмма female2a.wav

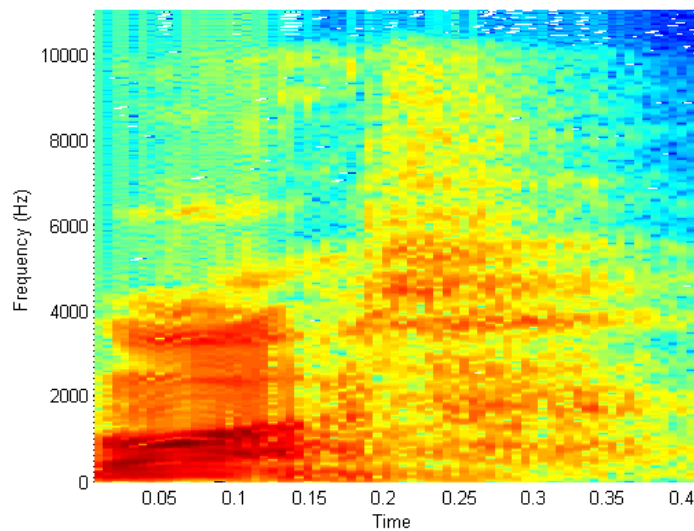


Рисунок 6.3.13 - Спектрограмма male1f.wav

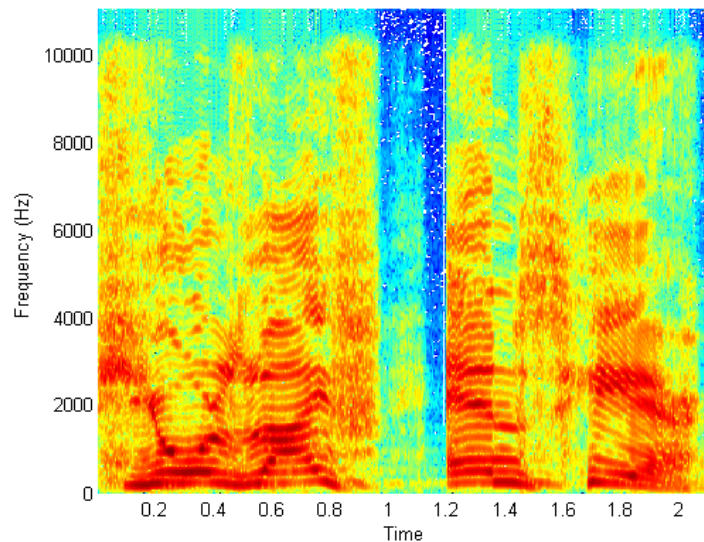


Рисунок 6.314 - Спектрограмма male2a.wav

Порядок выполнения работы

1. С помощью микрофонной гарнитуры введите в компьютер речевой сигнал (свою фамилию). Для этого удобно использовать программу «Звукозапись» из раздела «Стандартные – Развлечения».
2. С помощью команды `wavread` импортируйте речевой сигнал в среду Matlab. Определить параметры: F_s , bits.
3. Постройте график сигнала с помощью команды `plot` и `subplot`.
4. Определите время звучания и объем памяти для записанного сигнала.
5. Произведите обработку сигнала женского и мужского голоса.

Задание

1. Изучить теоретический материал.
2. Выберите данные фрагмента сигнала, где $n1=5000$, а $n2$ рассчитывается по формуле $n2=1000N$, где N – номер студента в журнале.
3. Повторите пп. 3 и 4 порядка выполнения работы для фрагмента сигнала, полученного по пп. 2.

4. С помощью микрофонной гарнитуры и команды *wavrecord* введите в компьютер речевой сигнал (Фамилию студента), применив $F_s = 8000; 11025$.

5. Запишите вектор (или матрицу) по пп. 4 полученной записи на диск в виде wav-файла и mat-файла.

Содержание отчета

1. Цель работы.
2. Задание к работе.
3. Текст разработанной программы.
4. Таблицы и рисунки, полученные в результате выполнения лабораторной работы.
5. Выводы.

Контрольные вопросы

1. Приведите схему речеобразования у человека?
2. В каком частотном диапазоне находится женский голос?
3. В каком частотном диапазоне находится мужской голос?
4. Что такое фонема?
5. На какие группы делятся звуки речи?
6. На какие группы делятся согласные звуки?
7. Перечислите алгоритм записи данных из речевого сигнала в системе MATLAB.
8. Перечислите алгоритм считывания данных из речевого сигнала в системе MATLAB.
9. Каков синтаксис вызова функции спектрограммы?
10. Каковы особенности женского и мужского голоса?
11. С помощью каких программ можно осуществлять анализ и обработку речи?

Лабораторная работа №7 **«Базовые средства фильтрации шумов на изображении в системе MATLAB»**

Целью работы: изучить особенности работы и графические возможности системы MATLAB, способы выполнения команд в MATLAB; освоить возможности анализа и улучшения изображений, предоставляемые системой MATLAB и пакетом программ Image Processing Toolbox (IPT).

Теоретические сведения

Главной целью обработки изображений, является обнаружение объектов и идентификация этих объектов.

Целью обработки может являться также улучшение качества изображения для лучшего визуального восприятия, геометрические преобразования (масштабирование, поворот, в общем), нормализация изображений по яркости, контрастности, резкости, выделение границ изображений, автоматическая классификация и подсчет однотипных объектов на изображении, сжатие информации об изображении.

К основным видам искажений изображений, затрудняющих идентификацию, можно отнести:

- Недостаточную контрастность и яркость, связанную с недостаточной освещенностью объекта;
- Наличие шумов на изображении, вызванных различными факторами;
- Искажения, вызванные «зернистостью» из-за квантования изображения;
- Искажения, связанные с перемещением видеорегистратора или объекта во время экспозиции и связанную с этим «смазанность» изображения;
- Математические основы и методы обработки изображений применительно как к объектам живой, так и неживой природы.

Элементарные операции с изображениями

К элементарным операциям, используемых при обработке изображений, можно отнести:

- 1) Чтение и запись файлов с изображениями;
- 2) Преобразование графических форматов;
- 3) Улучшение качества изображений (путем выравнивания гистограммы, изменения яркости палитры, корректировки яркости и контрастности)
- 4) Масштабирование изображений;
- 5) Выполнение геометрических операций с изображениями (кадрирование, изменение размеров, поворот);
- 6) Функциональные преобразования (двумерное прямое и обратное дискретное косинусное преобразование, двумерное быстрое преобразование Фурье, прямое и обратное преобразование Радона);
- 7) Корреляционное сравнение изображений;
- 8) Фильтрация шумов на изображениях (медианная, ранговая, адаптивная и др.);
- 9) Выполнение операций с пикселями (построение контурных графиков объектов, вычисление признаков объектов, построение гистограммы изображения, построение профиля изображения);

Для обработки изображений может быть использована программная среда MATLAB, в которой имеется большое количество библиотечных функций для работы с изображениями и Visual C++ совместно с библиотекой обработки изображений Open CV (Computer Vision).

Чтение и запись файлов с изображениями

В пакете расширения Image Processing программы MATLAB используются различные форматы изображений: BMP, JPEG, PCX, PNG, TIFF, CUR, HDF, ICO, XWD и представляющих их файлов. Соответствующие типы файлов могут быть прочитаны с помощью функции `imread`. Функция `A=imread(filename, fmt)` читает из файла с именем `filename` полутоновое или полноцветное изображение и создает двумерный массив `A` если изображение полутоновое или трехмерный массив размера $m \times n \times 3$ – если изображение

полноцветное. Параметр `fmt` задает формат изображения и может принимать значения 'bmp', 'jpg', 'jpeg', 'pcx', 'png', 'tif', 'cur', 'hdf', 'ico', 'xwd'.

Функция `imwrite(A,filename,fmt)` записывает изображение `A` в файл с именем `filename` с расширением `fmt`.

Вывод изображения на экран

```
J=imread('moon.tif');
figure;
imshow(J);
```

Базовые средства фильтрации шумов на изображениях

Алгоритмов улучшения качества изображений известно достаточно много. Ряд из них ориентированы на уменьшения уровня шумов с целью обнаружения объектов на изображениях или идентификации объектов. Основные методы: фильтрация, прямое и обратное вейвлет-преобразование, деблюринг (устранение смазанности), накопление.

Применение фильтров

Для уменьшения шумов на изображениях предусмотрены несколько различных фильтров: усредняющий (`average`), медианный (`medfilt2`), Гаусса (`gaussian`).

Функции медианной фильтрации (`medfilt2`), ранговой фильтрации (`ordfilt2`) и адаптивной фильтрации (`wiener2`) наиболее эффективны для борьбы с зашумленностью изображения имеющего вид «снега».

Усредняющий и медианный фильтры относятся к классу фильтров низких частот. Они могут использоваться для уменьшения влияния шума на изображение, однако их применение приводит к размытию изображения и уменьшению его четкости.

Медианная фильтрация представляет собой нелинейную операцию, более эффективную, чем конволюция, когда целью является одновременно уменьшение шума и сохранение границ объектов на изображении.

Функция медианной фильтрации `J= medfilt2(I,[m n])` фильтрует матрицу исходного изображения, используя маску фильтра

размером $m \times n$. По умолчанию – маска имеет размер 3×3 пиксела. Центральный пиксел маски заменяют медианой всех ее пикселов. Маска применяется нерекурсивно ко всему изображению. На краях изображение дополняется нулевыми элементами при классе I – unit8 или единицами при классе I – double, поэтому в результате фильтрации крайние элементы изображения могут быть искажены.

Эффект улучшения изображения при использовании медианного фильтра показан на рисунке 7.1.

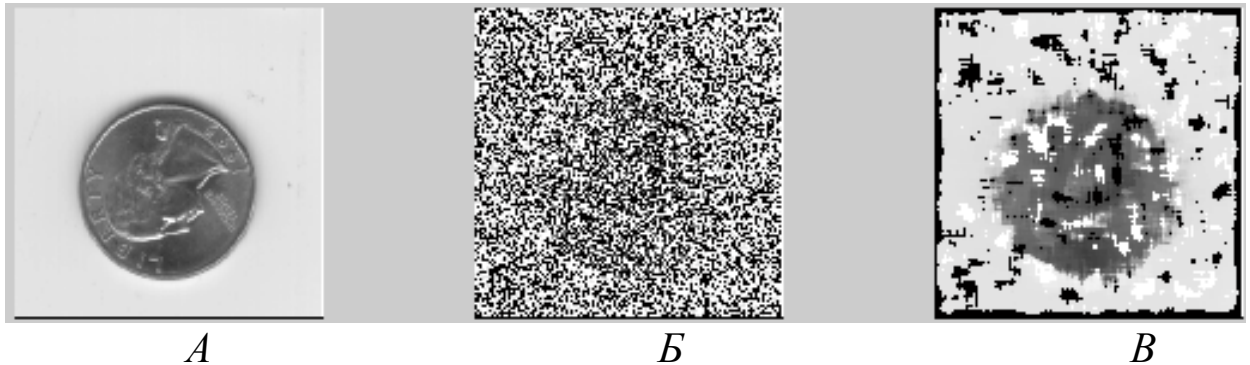


Рисунок 7.1 - Исходное изображение хорошего качества (А), зашумленное (Б) и полученное в результате фильтрации (В)

В данном случае применение фильтра решает задачу обнаружения объекта на изображении. На рисунке 7.2 показан пример улучшения качества изображения, выполняемого с целью достижения максимальной идентичности, оцениваемой по коэффициенту кросскорреляции, отфильтрованного изображения с исходным незашумленным изображением.

Усредняющий фильтр (average) функционирует таким же образом, как и медианный, но отличается тем, что центральный пиксел маски вычисляется как среднеарифметическое значение всех ее пикселов. Если медианный фильтр эффективнее при шуме типа «соль и перец», то усредняющий фильтр имеет преимущество при гауссовом белом шуме.

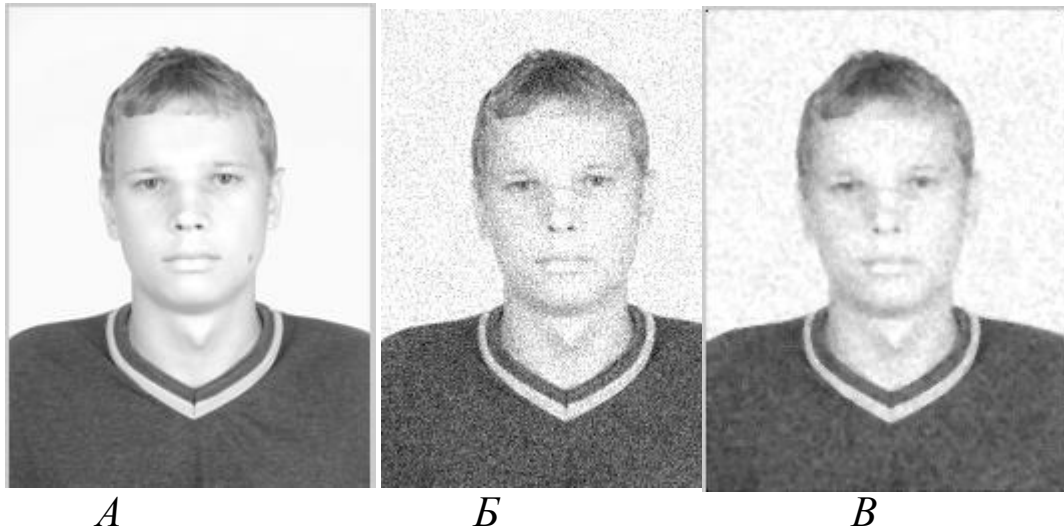


Рисунок 7.2 - Исходное (А), зашумленное (Б) и полученное в результате фильтрации (В) изображения. Вид шума – гауссовый белый, $\text{variance}=0.005$.

С точки зрения визуального восприятия зашумленное изображение выглядит лучше, чем отфильтрованное, однако отфильтрованное более близко к исходному при корреляционном сравнении. Коэффициент корреляции зашумленного изображения с исходным – 0.80, отфильтрованного с исходным – 0.92. Ниже приведен пример программ усредняющего и медианного фильтров.

```
%Усредняющий фильтр 'Average'
hsize=[3 3]; % маска фильтра
h= fspecial('average',hsize);
J=imfilter(I,h,'replicate');
figure;
imshow (J);
title('После усредняющего фильтра');
```

```
%Медианный фильтр 'medfilt2'
hsize=[3 3]; % маска фильтра
J= medfilt2(I,hsize);
figure;
imshow(J);
title('После медианного фильтра');
```

Фильтр Гаусса (gaussian) также относится к классу фильтров низких частот, но, по сравнению с усредняющим фильтром при гауссовом белом шуме, он меньше размывает изображение. Отфильтрованное изображение получается также близко к исходному при корреляционном сравнении. Коэффициент корреляции зашумленного изображения рис. 2Б при гауссовом белом шуме с $\text{variance} = 0.005$ с исходным – 0.80, отфильтрованного с исходным при использовании как усредняющего фильтра, так и фильтра Гаусса – 0.97. Ниже приведен пример программы фильтра Гаусса.

```
%Фильтр Гаусса 'gaussian'
hsize=[9 9];
sigma=0.99;
h= fspecial('gaussian',hsize,sigma);
J=imfilter(I,h,'replicate');
figure;
imshow(J);title('После фильтра Гаусса');
```

Однозначный выбор вида фильтра при обработке изображения может быть сделан с учетом решаемой задачи и только в случае, если точно известен характер шума. Если же характер шума неизвестен, нужно пробовать применить для обработки изображения последовательно все вышеупомянутые фильтры.

Улучшение отношения сигнал/шум путем накопления

Еще один возможный вид обработки изображений с целью улучшения отношения сигнал/шум – накопление, т.е. усреднение нескольких изображений. Благодаря тому, что изображение объекта остается неизменным, а шумовая составляющая изменяется, в результате накопления происходит улучшение отношения сигнал/шум. На рисунке 7.3 приведен пример эффекта улучшения отношения сигнал/шум в результате 10 накоплений.

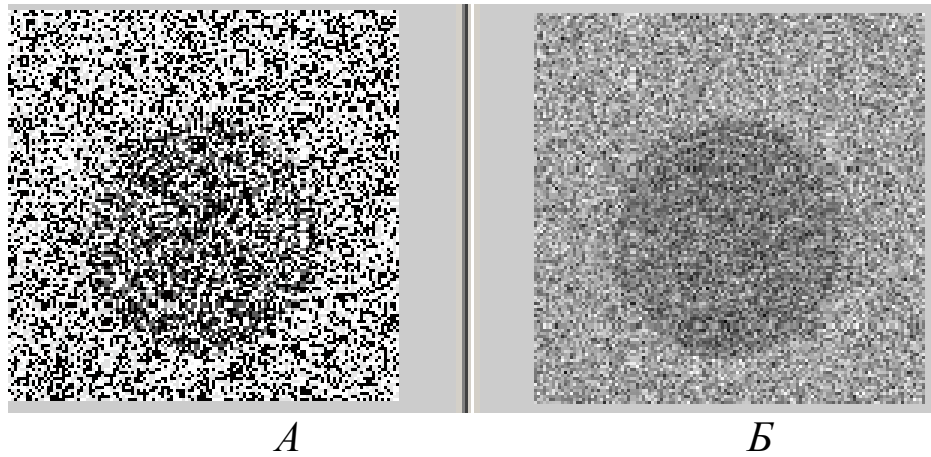


Рисунок 7.3 - Зашумленное изображение (А) и результат 10-ти накоплений (Б)

Эффект улучшения отношения сигнал/шум может быть достигнут при условии, если положение объекта в кадре остается неизменным. Поэтому необходимо устранение эффектов смещения объекта в кадре, связанного с качкой и перемещением корабля, на котором установлена система.

Таким образом, средства работы с фотоизображениями, представленные в MATLAB, являются достаточно полными и эффективными при решении задачи улучшения качества изображений. Кроме того, MATLAB является открытой системой, т.е. дает возможность модифицировать имеющиеся программы обработки и создавать другие.

Время выполнения базовых функций MATLAB для работы с изображениями на компьютере IBM PC класса Pentium-4 приведено в таблице 7.1.

Таблица 7.1 - Время выполнения функций для работы с изображениями, мсек

№№	ВЫПОЛНЯЕМАЯ ФУНКЦИЯ	Формат изображения	
		640x480	800x600
11	Чтение файла изображения	32	36
22	Отображение изображения	420	420
66	Ранговая фильтрация (Улучшение фокусировки)	130	220
77	Медианная фильтрация	100	140

Наложение шумов на изображение

Для выбора наилучшего метода фильтрации шумов и его параметров необходимо знать статистические свойства шума и параметры изображения, такие как яркость, контрастность, резкость, наличие или отсутствие смазанности. При этом возможно проведение модельного эксперимента, который позволит изучить эффективность того или иного метода фильтрации шумов при вариациях параметров изображения и параметров шума.

Для добавления шума к изображению MATLAB предоставляет функцию $J = \text{imnoise}(I, \text{type})$, где I – исходное изображение, type – класс шума:

- ‘gaussian’ – гауссовый белый шум;
- “salt & pepper” – «соль и перец», шум в виде включенных или выключенных пикселей;
- ‘speckle’ – мультипликативный шум.

Функция $J = \text{imnoise}(I, \text{type}, \text{parameters})$ позволяет дополнительно задать параметры шума, например:

- $J = \text{imnoise}(I, \text{'gaussian'}, m, v)$ добавляет гауссовый белый шум со средним значением m и отклонением v (по умолчанию $m=0, v=0.01$);
- $J = \text{imnoise}(I, \text{'salt \& pepper'}, d)$ параметр d задает плотность шума, по умолчанию $d=0.05$;
- $J = \text{imnoise}(I, \text{'speckle'}, v)$ добавляет мультипликативную компоненту шума, так что $J = I + n * I$, где n – равномерно распределенный шум со средним значением 0 и среднеквадратичным отклонением v (по умолчанию $v=0.04$). Этот шум не виден на темных участках изображения, но проявляется на его светлых участках.

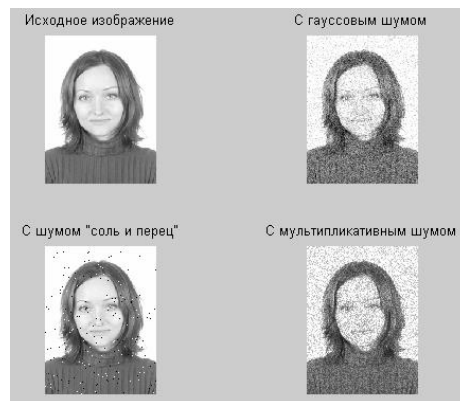


Рисунок 7.4 - Пример исходного и зашумленных изображений

%Программа производит зашумление изображения
m00200011.tif
%тремя различными способами

```
img1=imread ('m00200011.tif'); subplot(2,2,1);
imshow (img1);
title('Исходное изображение');
img2=imnoise(img1,'gaussian',0.03);subplot(2,2,2);
imshow (img2);
title('С гауссовым шумом');
img3=imnoise(img1,'salt & pepper',0.02);subplot(2,2,3);
imshow (img3);
title('С шумом "соль и перец"');
img4=imnoise(img1,'speckle',0.03);subplot(2,2,4);
imshow (img4);
title('С мультипликативным шумом');
```

Входное изображение должно быть представлено матрицей классов `unit8`, `unit16` или `double`. Выходной сигнал имеет тот же класс, что и входной. Файл входного черно-белого или цветного изображения может быть представлен в формате `.bmp`, `.jpg`, `.tif`, `.gif` и т.д.

Порядок выполнения работы

1. Загрузить изображение.
2. Установить уровень шума 0.005, вид шума – Гауссов, произвести кадрирование изображения, выделив центральную часть лица, и зафиксировать в таблице при выполнении программы максимальные значения коэффициентов корреляции зашумленного изображения и отфильтрованных разными фильтрами изображений. Затем, изменяя параметры фильтров, добиться получения наиболее высоких значений коэффициентов корреляции отфильтрованных сигналов с исходным и вспомогательным.
3. Повторить выполнение п. 2 при шуме `salt & pepper` с параметрами $d=0.01$, 0.05 и 0.1 и при мультипликативном шуме `speckle` с нулевым средним значением и среднеквадратичным отклонением $v=0.02$, 0.04 и 0.06 .

4. При выполнении п. 3 спроектировать на внутреннем языке MATLAB программу цифровой обработки изображений, реализующую уменьшение шумов на изображениях, используя стандартные базовые программы:

- average – усредняющий фильтр;
- medfilt – медианный фильтр;
- gaussian – фильтр Гаусса;
- ordfilt – ранговый фильтр;
- unsharp – фильтр повышения резкости;
- wiener2 – адаптивный фильтр Винера;

Программа должна позволять производить интерактивный выбор вида фильтра и параметров фильтра

Задание

1. Исследовать связь эффективности фильтрации зашумленных изображений с визуальным восприятием изображений и степенью сходства отфильтрованного изображения с исходным незашумленным изображением, найти оптимальные параметры фильтров;

2. Исследовать связь эффективности фильтрации от показателей яркости и контрастности исходного изображения;

3. Спроектировать на внутреннем языке MATLAB программу цифровой фильтрации изображений, содержащей средства интерфейса пользователя, позволяющие выбирать вид и задавать параметры фильтрации (приложение 6).

Содержание отчета

1. Цель работы.

2. Задание к работе.

3. Таблицы с показателями корреляционного сравнения изображений по п. 2 и 3 в соответствии порядка выполнения работы с комментариями.

4. Текст программы по п. 4 в соответствии порядка выполнения работы и экранная форма.

5. Выводы.

Контрольные вопросы

1. Какие основные виды искажений изображений, затрудняющих идентификацию, можно выделить?
2. Перечислите элементарные операции, производимые над изображениями.
3. Какие команды используются для чтения и записи файлов с изображением?
4. Какие фильтры применяются для обработки изображений?
5. Перечисли алгоритм наложения шума на изображение.

Литература

1. Ахмад, Х.М. Ведение в цифровую обработку речевых сигналов: учеб. пособие/ Х.М. Фхмад, В.Ф. Жирков; Владимир: Гос. ун-т. – Владимир: Издательство Владимир Гос. ун-т, 2007. – 192 с.
2. Дьяконов, В.П. Энциклопедия Mathcad 2001i и Mathcad 11/ В.П. Дьяконов. – М.:СОЛОН – Пресс, 2004. – 832 с.
3. Зубов, В.С. Программирование на языке TURBO PASCAL (версии 6.0 и 7.0)/ В.С. Зубов - М.: Информационно-издательский дом "Филинь", 1997.-304с.
4. Исследование алгоритмов обработки сигналов в системе MATLAB: метод. Указания к лабораторным работам/ Владим. гос. ун-т; сост. Е.К. Левин. - Владимир: Издательство Владимир Гос. ун-т, 2011. – 78 с.
5. Симонович, С.В. Информатика: Базовый курс/С.В.Симонович и др. – Спб.:Питер, 2004. – 640 с.
6. Каганов, В.И. Радиотехника + компьютер + Mathcad/В.И. Каганов. – М.:Горячая линия – Телеком, 2001. – 416 с.
7. Немирко, А.П. Обработка данных в среде MATLAB: Методические указания к практическим и лабораторным занятиям / Сост.: А. П. Немирко, Л. А. Манило. - Спб.: Изд-во СПбГЭТУ «ЛЭТИ»,2013. - 35 с
8. Очков, В.Ф. Mathcad 12 для студентов и инженеров/ В.Ф. Очков. – Спб.: БХВ – Петербург, 2005. – 464 с.
9. Сергиенко, А.Б. Цифровая обработка сигналов/ А.Б. Сергиенко. - Спб. Питер. 2002.-608 с.
10. Тутыгин, В.С.Цифровая обработка сигналов: Лаборат. Практикум/В.С. Тутыгин. - Спб.: СПбГПУ, 2012. - 58с.
11. Циммерман, Франклин Клиническая электрокардиография/ Франклин Циммерман. -М.: «Издательство БИНОМ», 1997.-448с.

Пример программного кода, генерируемого автоматически после создания панели и объектов GUI в среде MATLAB

```

%Служебная часть М-файла, создаваемая автоматически после
сохранения в файле
%панели интерфейса пользователя <имя>.fig
%В данном примере панель интерфейса пользователя содержит
кнопку управления
%PushButton, окно ввода/вывода текста EditText (tag – КР),
окно графического
%вывода Axes (tag – axis1)

function varargout = testlab2(varargin)
% TESTLAB2 M-file for testlab2.fig
% TESTLAB2, by itself, creates a new TESTLAB2 or raises the
existing
% singleton*.
%
% H = TESTLAB2 returns the handle to a new TESTLAB2 or the
handle to
%the existing singleton*.
%
%      TESTLAB2('CALLBACK',hObject,eventData,handles,...)
calls the local
%   function named CALLBACK in TESTLAB2.M with the given
input arguments.
%
%TESTLAB2('Property','Value',...) creates a new TESTLAB2 or
raises the
%   existing singleton*. Starting from the left, property value pairs
are
%   applied to the GUI before testlab2_OpeningFcn gets called.
An
%   unrecognized property name or invalid value makes property
application
%   stop. All inputs are passed to testlab2_OpeningFcn via
varargin.
%
```

```

%      *See GUI Options on GUIDE's Tools menu. Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help testlab2

% Last Modified by GUIDE v2.5 03-Feb-2012 21:54:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @testlab2_OpeningFcn, ...
                  'gui_OutputFcn', @testlab2_OutputFcn, ...
                  'gui_LayoutFcn', [] , ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before testlab2 is made visible.
function testlab2_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to testlab2 (see VARARGIN)

```

```

% Choose default command line output for testlab2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes testlab2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = testlab2_OutputFcn(hObject, eventdata,
handles)
% varargout    cell array for returning output args (see
VARARGOUT);
% hObject     handle to figure
% eventdata   reserved - to be defined in a future version of
MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function KP_Callback(hObject, eventdata, handles)
% hObject     handle to KP (see GCBO)
% eventdata   reserved - to be defined in a future version of
MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of KP as text
%         str2double(get(hObject,'String')) returns contents of KP as a
double

% --- Executes during object creation, after setting all properties.
function KP_CreateFcn(hObject, eventdata, handles)
% hObject     handle to KP (see GCBO)
% eventdata   reserved - to be defined in a future version of
MATLAB

```

```
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%    See ISPC and COMPUTER.
if    ispc    &&    isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in Start.
function Start_Callback(hObject, eventdata, handles)
% hObject    handle to Start (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
KP=str2double(get(handles.KP,'string'));
for i=1:1000
    y(i)=sin((2*pi*i*KP)/1000.0);
end
i=1:1000;
plot(i,y);
```

```
% --- Executes on button press in Quit.
function Quit_Callback(hObject, eventdata, handles)
% hObject    handle to Quit (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
fclose('all');
close ('all');
```

Пример программного кода в функции Start.

```

% --- Executes on button press in Start.
function Start_Callback(hObject, eventdata, handles)
% hObject   handle to Start (see GCBO)
% eventdata reserved - to be defined in a future version of
MATLAB
% handles   structure with handles and user data (see GUIDATA)
KP=str2double(get(handles.KP,'string'));
for i=1:1000
    y(i)=sin(2*pi*i*KP./1000.0);%KP – кол-во периодов
end
i=1:1000
plot(i,y);

```

После запуска программы Debug/Run диалоговое окно программы с результатами выполнения будет выглядеть, как показано на рисунке 5.3.

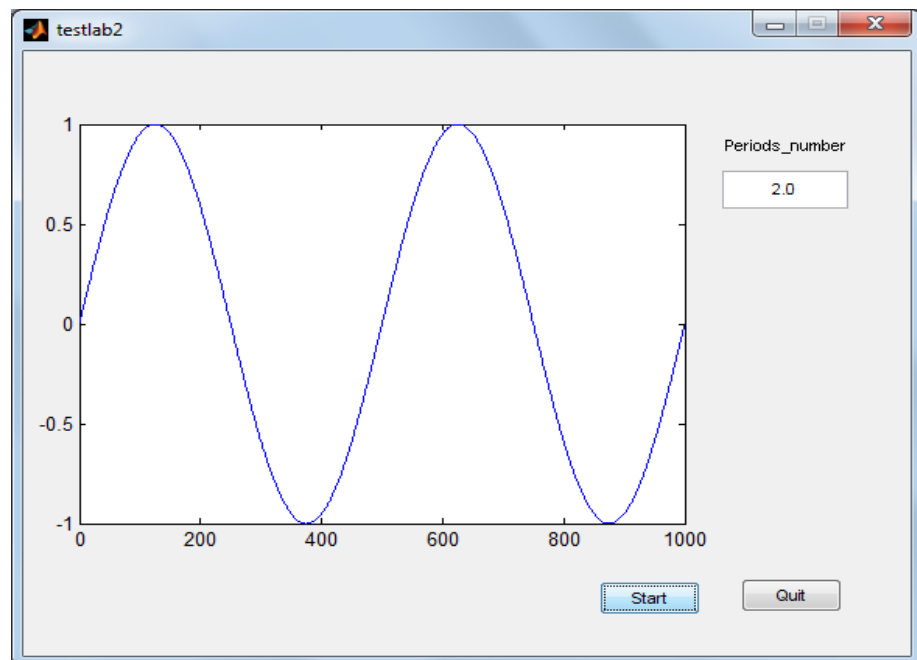


Рисунок 5.3 – Результат выполнения работы программы

Приложение 3

Построение и оформление графиков в MATLAB

```

i=1:N;
y=A*sin(6.28*КР*i/N); %создание N значений КР периодов
модельного sin %сигнала,
figure %создание окна графического вывода
%plot(i,y(1:50));%вывод графика y, кол-во точек – 50.
%plot(i,y) ;% вывод графика y (все N точек)
%plot (i,2*y(1:N),'r-');%отображение графика y линией
красного цвета
%semilogx(i,abs(y(1:200))),grid; %то же, что и plot, но в
логарифмическом
% масштабе по X
%loglog(i,abs(y(1:200))); % в логарифмическом масштабе по X
и Y
grid on; %отображение линий координатной сетки
%title('сигнал до фильтра');%заголовок графика
xlabel('номер отсчета'); % подпись по оси X
ylabel('амплитуда'); % подпись по оси Y
legend('до фильтра');%подпись легенды
axis tight; %диапазон X и Y по осям точно соответствует Xmax
и Ymax %(автомасштабирование)
hold on; % «удержание» окна вывода для следующего графика
hold off; %отмена «удержания»
close all; %закрытие всех открытых окон графического вывода

```


Текст базовой программы

```

%Комбинированное использование ключевых операций ЦОС
%Для повышения точности определения количества периодов
и частоты
%"короткого" сигнала используется комбинация
%БПФ, кросскорреляции, сплайн-
аппроксимации, передискретизации
kt=1024; % количество отсчетов
f=12; %частота сигнала
dt=2;%шаг дискретности по времени при измерении
k2=500; %декремент затухания огибающей сигнала
shum=0;%шум
kp=4.5;%количество периодов сигнала

%1. ГЕНЕРАЦИЯ МОДЕЛЬНОГО СИГНАЛА
for i=1:kt %обнуление массива сигнала
    y(i)=0;
end
for i=1:kt %генерация модельного сигнала
    if(i>0)&(i<=kt/2)
        y(i)=sin(2*3.14*kp*i/kt)*i*exp(i/k2)/1200;
    end
    if(i>kt/2)&(i<(kt))
        y(i)=sin(2*3.14*kp*i/kt)*(kt-i)*exp((kt-i)/k2)/1200;
    end
    y(i)=y(i)+shum*(2*rand(1)-1);
end
i=1:kt; %отображение модельного сигнала во временной
области
figure
plot(i,y);
axis tight;
title('Time domain')
xlabel('Sample number')
%2. ФУНКЦИОНАЛЬНОЕ ПРЕОБРАЗОВАНИЕ (БПФ)
bpfy=fft(y,kt);%БПФ
bpf=bpfy.*conj(bpfy)/kt;%БПФ

```

```

%f=1000*(0:256)/512;
figure
plot(i(1:257),bpf(1:257));
axis tight;
title('Frequency domain')
xlabel('frequency')
%нахождение макс. знач. функции БПФ для массива Y
C=max(bpf);
for i=1:kt %поиск количества периодов, соответствующих
максимуму БПФ
    if (bpf(i)==C)
        krbpf=(i-1);
        break
    end
end
kr_bpf=krbpf
%3. СОЗДАНИЕ ЭТАЛОНОВ И КРОССКОРРЕЛЯЦИЯ
fr=krbpf;
procch=0.1;%область поиска в процентах относит. kr_bpf
shagkor=fr*procch/3;%шаг поиска
k=0;
for iii=fr-fr*procch:shagkor:fr+fr*procch %цикл для создания 6
эталонов в окрестности приближенного
    %значения количества периодов, определенных с
помощью БПФ.
        k=k+1;
        xkor(k)=iii;
        kor(k)=0;
        for i=1:kt
            x(i)=0;
        end
    end
%Вычисление массивов эталонных сигналов
for i=1:kt
    if(i>0)&(i<=kt/2)
        x(i)=sin(2*3.14*iii*i/kt)*i*exp(i/k2)/1200;
    end
    if(i>kt/2)&(i<(kt))
        x(i)=sin(2*3.14*iii*i/kt)*(kt-i)*exp((kt-i)/k2)/1200;
    end
end

```

```

end
%вычисление средних значений модельного и эталонных
сигналов
    x_sr=mean(x);
    y_sr=mean(y);
    x_sko=0;
    y_sko=0;
%вычисление СКО и коэф. корреляции модельного и
эталонных сигналов
    for i=1:kt
        x_sko=x_sko+(x(i)-x_sr)*(x(i)-x_sr);
        y_sko=y_sko+(y(i)-y_sr)*(y(i)-y_sr);
        kor(k)=kor(k)+(x(i)-x_sr)*(y(i)-y_sr);
    end
    kor(k)=kor(k)/(sqrt(x_sko*y_sko));
end %конец цикла создания эталонов и вычисления массива
коэф. корр.
%СПЛАЙН-АППРОКСИМАЦИЯ И
ПЕРЕДИСКРЕТИЗАЦИЯ
    xx=1:k;
    xi=1:0.1:k;
    r1=sin(xx); %только для тестирования сплайн-
аппроксимации
    yint=interp1(xx,kor,xi,'spline');% сплайн-аппроксимация
коэф корреляции
    r1=kor;
    %%apr=csaps(xx,r1);
    apr=spaps(xkor,kor,0.000001);%%%%%%%%%%
    fnplt(apr)
    hold on
    plot(xkor,r1,'ro')%%%%%%%%%%
    hold off
%НАХОЖДЕНИЕ УТОЧНЕННОГО ЗНАЧЕНИЯ
КОЛИЧЕСТВА ПЕРИОДОВ СИГНАЛА
    cmax=max(yint); %нахождение максимума коэф. корр.
    for i=1:round((k-1)/0.1+1)
        if (yint(i)==cmax)
            kp_int=fr-fr*procch+(i-1)*shagkor/10 %уточненное
значение частоты по МАХ функции коэф. корр.

```

```
        end
    end
pause;
close all;
```

Текст базовой программы

```

%Программа производит зашумление изображения
%фильтрацию зашумленного изображения с помощью
%различных фильтров, определение коэффициента
корреляции
%зашумленного и отфильтрованных изображений с исходным

img=imread ('m00200021.tif');
I=imcrop(img);
figure;
imshow (img);
title('Исходное изображение');
img1=imnoise(img,'gaussian',0,0.005);% 0.005
figure;
imshow (img1);
title('Зашумленное изображение');

%Медианный фильтр 'medfilt2'
hsize=[3 3 ];
F4= medfilt2(img1,hsize);
figure;
imshow(F4);
title('После медианного фильтра');
pause;

%Усредняющий фильтр 'Average'
hsize=[3 3 ];
h= fspecial('average',hsize);
F1=imfilter(img1,h,'replicate');
figure;
imshow (F1);
title('После усредняющего фильтра');
pause;

ncorr = normxcorr2(I(:,:,1),img1(:,:,1));
figure, surf(ncorr), shading flat;
title('После зашумления');

```

```
max_c_noised = max(abs(ncorr(:)))
pause;
```

```
ncorr = normxcorr2(I(:,:,1),F4(:,:,1));
figure, surf(ncorr), shading flat;
title('После медианного фильтра');
    max_c_median = max(abs(ncorr(:)))
pause;
```

```
ncorr = normxcorr2(I(:,:,1),F1(:,:,1));
figure, surf(ncorr), shading flat;
title('После усредняющего фильтра');
max_c_average = max(abs(ncorr(:)))
pause;
```

```
% Фильтр Гаусса 'gaussian'
hsize=[9 9];
sigma=0.99;
h= fspecial('gaussian',hsize,sigma);
F2=imfilter(img1,h,'replicate');
figure;
imshow(F2);title('После фильтра Гаусса');
pause;
```

```
ncorr = normxcorr2(I(:,:,1),F2(:,:,1));
figure, surf(ncorr), shading flat;
title('После фильтра Гаусса');
max_c_gaussian = max(abs(ncorr(:)))
pause;
```

```
% Фильтр Лапласа 'laplacian'
alpha=0.5;
h= fspecial('laplacian',alpha);
F3=imfilter(img1,h,'replicate');
figure;
imshow(F3);title('После фильтра Лапласа');
pause;
```

```
ncorr = normxcorr2(I(:,:,1),F3(:,:,1));
```

```
figure, surf(ncorr), shading flat;  
title('После фильтра Лапласа');  
max_c_laplasiаn = max(abs(ncorr(:)))  
pause;
```

```
%Фильтр повышения резкости 'unsharp'  
alpha=0.2;  
h= fspecial('unsharp',alpha);  
F4=imfilter(img1,h);  
figure;  
imshow(F4);title('После фильтра unsharp');  
pause;
```

```
ncorr = normxcorr2(I(:,:,1),F4(:,:,1));  
figure, surf(ncorr), shading flat;  
title('После фильтра unsharp');  
max_c_unsharp = max(abs(ncorr(:)))  
pause;
```

```
%Ранговый фильтр
```

```
m=5;n=5;  
J = imnoise(img1,'gaussian',0,0.005);  
imshow(J);  
J6 = ordfilt2(J, 7, ones(4,4));  
figure;  
imshow(J6);title('После рангового фильтра');  
pause;
```

```
ncorr = normxcorr2(I(:,:,1),J6(:,:,1));  
figure, surf(ncorr), shading flat;  
title('После рангового фильтра');  
max_c_ordfilt2 = max(abs(ncorr(:)))  
pause;
```

```
close all;
```

Текст базовой программы

```

%Программа производит зашумление изображения
%фильтрацию зашумленного изображения с помощью
%различных фильтров, определение коэффициента
корреляции
%зашумленного и отфильтрованных изображений с исходным

img=imread ('m00200021.tif');
I=imcrop(img);
figure;
imshow (img);
title('Исходное изображение');
img1=imnoise(img,'gaussian',0,0.005);% 0.005
figure;
imshow (img1);
title('Зашумленное изображение');

%Медианный фильтр 'medfilt2'
hsize=[3 3 ];
F4= medfilt2(img1,hsize);
figure;
imshow(F4);
title('После медианного фильтра');
pause;

%Усредняющий фильтр 'Average'
hsize=[3 3 ];
h= fspecial('average',hsize);
F1=imfilter(img1,h,'replicate');
figure;
imshow (F1);
title('После усредняющего фильтра');
pause;

ncorr = normxcorr2(I(:,:,1),img1(:,:,1));
figure, surf(ncorr), shading flat;
title('После зашумления');

```



```
max_c_noised = max(abs(ncorr(:)))
pause;
```

```
ncorr = normxcorr2(I(:,:,1),F4(:,:,1));
figure, surf(ncorr), shading flat;
title('После медианного фильтра');
    max_c_median = max(abs(ncorr(:)))
pause;
```

```
ncorr = normxcorr2(I(:,:,1),F1(:,:,1));
figure, surf(ncorr), shading flat;
title('После усредняющего фильтра');
max_c_average = max(abs(ncorr(:)))
pause;
```

```
% Фильтр Гаусса 'gaussian'
hsize=[9 9];
sigma=0.99;
h= fspecial('gaussian',hsize,sigma);
F2=imfilter(img1,h,'replicate');
figure;
imshow(F2);title('После фильтра Гаусса');
pause;
```

```
ncorr = normxcorr2(I(:,:,1),F2(:,:,1));
figure, surf(ncorr), shading flat;
title('После фильтра Гаусса');
max_c_gaussian = max(abs(ncorr(:)))
pause;
```

```
% Фильтр Лапласа 'laplacian'
alpha=0.5;
h= fspecial('laplacian',alpha);
F3=imfilter(img1,h,'replicate');
figure;
imshow(F3);title('После фильтра Лапласа');
pause;
```

```
ncorr = normxcorr2(I(:,:,1),F3(:,:,1));
```

```
figure, surf(ncorr), shading flat;  
title('После фильтра Лапласа');  
max_c_laplasiаn = max(abs(ncorr(:)))  
pause;
```

```
%Фильтр повышения резкости 'unsharp'  
alpha=0.2;  
h= fspecial('unsharp',alpha);  
F4=imfilter(img1,h);  
figure;  
imshow(F4);title('После фильтра unsharp');  
pause;
```

```
ncorr = normxcorr2(I(:,:,1),F4(:,:,1));  
figure, surf(ncorr), shading flat;  
title('После фильтра unsharp');  
max_c_unsharp = max(abs(ncorr(:)))  
pause;
```

```
%Ранговый фильтр
```

```
m=5;n=5;  
J = imnoise(img1,'gaussian',0,0.005);  
imshow(J);  
J6 = ordfilt2(J, 7, ones(4,4));  
figure;  
imshow(J6);title('После рангового фильтра');  
pause;
```

```
ncorr = normxcorr2(I(:,:,1),J6(:,:,1));  
figure, surf(ncorr), shading flat;  
title('После рангового фильтра');  
max_c_ordfilt2 = max(abs(ncorr(:)))  
pause;
```

```
close all;
```