**Аннотация**

Необходимыми и достаточными условиями для определения любой отрасли знаний как науки является наличие: предмета исследования, метода исследования и средств для реализации этого метода.

Для кибернетики, как науки, предметом исследования являются системы любой природы и их управляемость, методом исследования – математическое моделирование, стратегией исследования – системный анализ, а средством исследования – вычислительные машины. Поэтому кибернетику можно определить как науку, изучающую системы любой природы, способные воспринимать, хранить и перерабатывать информацию для целей оптимального управления.

Сейчас с каждым годом возрастает роль средств, позволяющих рационально использовать ресурсы, выделенные для решения задач. Часть методов кибернетики предназначены для увеличения эффективности научного эксперимента на всех стадиях разработки, исследования, проектирования и эксплуатации производств.

Кибернетика включает такие понятия, как системы, информация, хранение и переработка информации, управление системами и оптимизация систем. При этом кибернетика широко пользуется методом математического моделирования и стремится к получению конкретных результатов, позволяющих анализировать и синтезировать изучаемые системы, прогнозировать их оптимальное поведение и обнаруживать каналы и алгоритмы управления.

**Введение**

В тех случаях, когда информации о рассматриваемом процессе недостаточно или процесс настолько сложен, что невозможно составить его детерминированную модель, прибегают к экспериментально-статистическим методам. При этом процесс рассматривают как “черный ящик”. Различают пассивный и активный эксперимент.

Пассивный эксперимент является традиционным методом, в соответствии с которым ставится большая серия опытов с поочередным варьированием каждой из переменных.

Активный эксперимент ставится по заранее составленному плану (планирование эксперимента), при этом предусматривается одновременное изменение всех параметров, влияющих на процесс, что позволяет сразу установить силу взаимодействия параметров и на этом основании сократить общее число опытов.

Использование принципов регрессионного и корреляционного анализа при обработке опытных данных позволяет найти зависимость между переменными и условия оптимума. В обоих случаях математической моделью является функция отклика, связывающая параметр оптимизации, характеризующий результаты эксперимента, с переменными, которые экспериментатор варьирует при проведении опытов.

Регрессионный анализ – один из наиболее распространенных статистических методов. Он используется при построении математической зависимости на основе экспериментальных данных. Благодаря регрессионному анализу возможны построения математической модели и статистический анализ результатов. В первом случае прибегают к различным вариантам метода наименьших квадратов.

В основе регрессионного анализа лежит несколько статистических предпосылок, выполнение которых гарантирует достоверность анализа полученной математической модели:

Выходная переменная – случайная величина с нормальным распределением, факторы – суть не случайные величины; практически это означает, что ошибки в управлениями факторами, по крайней мере на порядок меньше ошибок при измерении выходной переменной.

Связь между факторами отсутствует.

Дисперсии выходной переменной однородны (равноточные) в любой точке факторного пространства.

Исследуемый объект лишен динамических свойств (рассматриваются стационарные режимы объекта).

**1. Описание метода и алгоритма решения**

Ставится задача определения локального оптимума на объекте исследования, для этого предполагается использовать математическую модель, полученную с помощью полного факторного эксперимента.

Выбирают факторы и выходную переменную, задают области определения факторов и выходной переменной

ymin ≤ y ≤ ymax

X1min ≤ X1 ≤ X1max

X2min≤X2 ≤X2max

........

В области определения факторов выбирается точка Xi0, i = 1…n (нулевой уровень факторов), которая в предварительных исследованиях была признана наилучшей с точки зрения оптимума y. Задается интервал варьирования факторов ΔXi. Определяются верхние и нижние уровни факторов:

Xiв = Xi0 + ΔXi;

Xiн = Xi0 - ΔXi (1)

при условии, что ( Xiн ÷Xiв) < ( Ximin÷Ximax).



Кодируются факторы (переход к новой безразмерной системе координат x1, x2, …, xn)



В новой системе координат факторы принимают значения +1 и –1.

План проведения эксперимента (матрица планирования) записывается в виде таблицы. Фиктивная переменная x0 равна единице. В матрицу также записывают результаты проведения параллельных опытов (m опытов в каждой строке матрицы).



План эксперимента должен обладать ортогональностью:



Как следствие (4) план эксперимента обладает симметричностью:

 и нормировкой

Расчет коэффициентов уравнения регрессии. Коэффициенты рассчитываются по уравнению



где

и окончательно

где N – число строк матрицы планирования (число разных условий опыта); m – число параллельных опытов на каждой строке матрицы.



Построчные дисперсии по параллельным опытам на каждой строке матрицы рассчитываются по уравнению

где fu = mu –1.

Проверка однородности дисперсий осуществляется по



критерию Кохрена, расчетное значение которого определяют по уравнению:

где simax – максимальная из рассчитанных дисперсий параллельных опытов (построчных дисперсий); знаменатель – сумма всех дисперсий по уровням фактора.

Если выполняется условие

Gp<Gт (fi = mi – 1, f2 = p, q = 0,05),

то гипотеза об однородности дисперсий правомерна. Gт находят по таблице критерия Кохрена для степеней свободы fi (максимальная дисперсия), f2 (число уровней) и заданного уровня значимости q.

Вся проверка однородности дисперсий осуществляется при условии mi = m, p = N; индекс i заменяется индексом u.

Расчет ошибки опыта производится усреднением построчных дисперсий



для числа степеней свободы

f0 = N ( m – 1 )



Оценка значимости коэффициентов регрессии производится расчетом t-критерия по формуле



где bi – i-й коэффициент регрессии, расчетом дисперсий коэффициентов по формуле с учетом и оценкой по условию

tip>tт (f0 = N0 –1, q = 0,05) (14)



Если для какого-то коэффициента условие (14) не выполняется, то соответствующий фактор можно признать незначимым и исключить его из уравнения.

Проверка адекватности уравнения регрессии осуществляется по формулам.



кибернетика локальный оптимум регрессия

где yu – экспериментальные значения выходной переменной; yu – значения, рассчитанные по уравнению регрессии; fад = N - l, где l – число связей, равное числу коэффициентов уравнения, оставшихся после проверки их значимости; sад – дисперсия адекватности.

Поиск Fт производится для степеней свободы fад и f0.

Если расчетное значение критерия Фишера

Fp<Fт (fад = N – l, f0 = N0 – 1, q = 0,05) (19)

для степеней свободы fад, f0 и заданного уровня значимости, то уравнение считается адекватным.

**2. Программная реализация алгоритма**

**2.1 Описание программы**

Программа написана в среде разработки BorlandDelphi 5.0. Она является работоспособной и изменяемой. Проверка работоспособности и нормального функционирования была произведена на IntelPentiumIII 1200 с объемом оперативной памяти в 512 Мб.

Данная программа производит вычисления по заранее заложенным в нее данным. Однако можно легко изменить ее на ввод данных с терминала не нарушая общей структуры.

**2.2 Описание алгоритма**

Список используемых переменных

expResult: array[ 1..8, 1..10 ] of real;

матрица хранящая результаты эксперимента

yAverage: array[ 1..8 ] of real;

матрица хранящая построчные средние значения экспериментальных данных У

yExpResult: array[ 1..8 ] of real;

матрица хранящая значения экспериментальных данных Y. Эксперимент построен по полученой математической модели.

regCoeficient: array[ 0..3 ] of real;

матрица хранящая значения коэфициентов регрессии, полеченных по формулам 7-8.

tCritery: array[ 0..3 ] of real;

матрица хранящая значения t-критерия для каждого коефициента уравнения регрессии, расчитанных по формуле 12.

dSu: array[ 1..8 ] of real;

матрица хранящая значения построчных дисперсий, полеченных по формуле 9.

dSo: real;

ошибка эксперимента, формула 11.

dSbi: real;

среднеквадратическое отклонение коэфициентов регрессии, необходимы для нахождения критерия Стьюдента, формулы 13-15.

dSad: real;

дисперсия адекватности математичексой модели.

fP: real;

переменная хранящая значение критерия Фишера. Проверка адекватности математической модели, формула 19.

G: real;

переменная хранящая значение критерия Кохрена. Проверка однородности дисперсий на каждом уровне фактора, формула 10.

gipotesa1D: boolean;

переменная хранящая значение однородности дисперсий. Значение True соответствует тому факту, что дисперсии однородны, False – обратное утверждение.

regAd: boolean;

переменная хранящая значение адекватности математической модели. Значение True соответствует тому факту, что математическая модель адекватна, False – обратное утверждение.

decisionRegMean: array[ 0..3 ] of boolean;

матрица хранящая значения значимости коэфициентов регрессии. Значение True соответствует тому факту, что данный коэфициент значим, False – обратное утверждение.

Список процедур и функций

function RandomNorm( mF, dF: real ): real;

фунция возвращает величину с заданой дисперсией и мат. ожиданием и нормальным распределением.

function CalculateX( level: byte ): real;

функция возвращающая значение х на заданном уровне эксперимента.

procedure MakeExperiment;

процедура осуществляет эксперимент при заданных условиях.

procedure CalculateYAverage;

процедура вичисляющая построчные средние значения Y.

procedureCalculateRegCoeficients;

процедура вычисляющая коєфициенты регресии.

procedureCalculatedSu;

процедура подсчитывает дисперсию dSu.

procedure Check1D;

процедура проверки однородности дисперсий.

procedure CalculatedSo;

процедура вычисления ошибки эксперимента.

procedure CalculateRegMean;

процедура осуществяющая проверку значимости коэфициентов регрессии.

procedure MakeDecision;

процедура принятия решений, по результатам проверки критерия Стьюдента.

function CalculateL: byte;

функция возвращающая количество значимых коефициентов регрессии, необходима для проверки адекватности уравненя регрессии по критерию Фишера.

procedure CalculateYExp;

процедура подсчета експериментального значения Y. Эксперимент проводится по полученному уравнению регрессии.

procedure CheckRegAd;

процедура проверки адекватности уравнения регрессии.

procedure FillPlaneMatrix;

процедура вывода данных: план эксперимента.

procedure FillExpMatrix;

процедура вывода данных: результат эксперимента.

procedure FillYAverage;

процедура вывода данных: построчние средние значения Y.

procedure FillRegCoeficient;

процедура вывода данных: коефициенты регрессии.

procedure FillDSu;

процедура вывода данных: построчные дисперсии.

procedure FillExpMistake;

процедура вывода данных: ошибка експеримента.

procedure FillRegCoeficientCritery;

процедура вывода данных: значимость коэйициентов регресии.

procedure FillYExp;

процедура вывода данных: Y полученый по уравнению регрессии.

procedure FillFCritery;

процедура вывода данных: критерий Фишера.

procedure FillresultTables;

процедура объеденяющая вывод данных.

**3. Инструкция пользователя**

Для управления данным программным продуктом используется всего одна кнопка “Произвести расчет”, так как программа работает с заданными начальными условиями и данными.

Соответственно все результаты работы программы можно просмотреть через закладки.

Последующие рисунки дают полную наглядность программы, обеспечивающуюся благодаря графическому интерфейсу.

Рисунок 1

Рисунок 2

Рисунок 3

Рисунок 4

**4. Результаты работы программы**

Рисунок 5

**Список литературы**

1. А.Г. Бондарь, Г.А. Статюха. «Планирование эксперимента в химической технологии». “Вища школа”. Киев 1976.
2. А.Г. Бондарь, Г.А. Статюха, И.А. Потяженко. «Планирование эксперимента при оптимизации процессов химической технологии». “Вища школа”. Киев 1980.
3. В.В. Кафаров. «Методы кибернетики в химии и химической технологии».

**Приложение**

**Листинг программы**

unitmain;

interface

uses

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

Menus, ComCtrls, Buttons, StdCtrls, Grids, Tabnotbk;

type

TFMain = class(TForm)

MainMenu1: TMainMenu;

F1: TMenuItem;

Exit1: TMenuItem;

SpeedButton1: TSpeedButton;

TNotebook: TTabbedNotebook;

Label1: TLabel;

SGridPlaneMatrix: TStringGrid;

SGridY: TStringGrid;

SGridYAverage: TStringGrid;

Label2: TLabel;

SGridRegCoef: TStringGrid;

Label3: TLabel;

Label4: TLabel;

SGridDSu: TStringGrid;

LblExpMistake: TLabel;

Label5: TLabel;

SGridCritery: TStringGrid;

LblFCritery: TLabel;

SGridYExp: TStringGrid;

Label6: TLabel;

StatusBar1: TStatusBar;

procedure Exit1Click(Sender: TObject);

procedure SpeedButton1Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FMain: TFMain;

implementation

uses ffe\_typeconst;

{$R \*.DFM}

function RandomNorm( mF, dF: real ): real;

begin

RandomNorm:= mF + dF \* random;

end;

function CalculateX( level: byte ): real;

var

xResult: real;

xU: real;

i: byte;

begin

xResult:= 0;

i:= 0;

if planingMatrix[ i, level ] = 1 then

xU:= u[ i, 2 ]

else

xU:= u[ i, 1 ];

xResult:= xResult + p[ i ] \* xU;

for i:= 1 to 3 do

begin

if planingMatrix[ i, level ] = 1 then

xU:= u[ i, 2 ]

else

xU:= u[ i, 1 ];

xResult:= xResult + p[ i ] \* xU;

xResult:= xResult + p[ i ] \* xU \* xU;

end;

CalculateX:= xResult;

end;

procedure MakeExperiment;

var

xValue: real;

i, j: byte;

begin

for i:= 1 to 8 do

begin

xValue:= CalculateX( i );

for j:= 1 to 10 do

expResult[ i, j ]:= c1 \* xValue + c2 \* RandomNorm( mF, dF );

end;

end;

procedure CalculateYAverage;

var

i, j: byte;

begin

for i:= 1 to 8 do

begin

yAverage[ i ]:= 0;

for j:= 1 to 10 do

yAverage[ i ]:= yAverage[ i ] + expResult[ i, j ];

yAverage[ i ]:= yAverage[ i ] / expNum;

end;

end;

procedure CalculateRegCoeficients;

var

i, j: byte;

begin

CalculateYAverage;

for i:= 0 to factorNum do

begin

regCoeficient[ i ]:= 0;

for j:= 1 to planeNum do

regCoeficient[ i ]:= regCoeficient[ i ] + planingMatrix[ i, j ] \* yAverage[ j ];

regCoeficient[ i ]:= regCoeficient[ i ] / planeNum;

end;

end;

procedure CalculatedSu;

var

i, j: byte;

begin

for i:= 1 to planeNum do

begin

dSu[ i ]:= 0;

for j:= 1 to expNum do

dSu[ i ]:= dSu[ i ] + sqr( expResult[ i, j ] - yAverage[ i ] ) / ( expNum - 1 );

end;

end;

function FindMax: real;

var

i: byte;

fResult: real;

begin

fResult:= 0;

for i:= 1 to planeNum do

if dSu[ i ] > fResult then

fResult:= dSu[ i ];

FindMax:= fResult;

end;

procedure Check1D;

var

dSum: real;

i: byte;

begin

dSum:= 0;

for i:= 1 to planeNum do

dSum:= dSum + dSu[ i ];

G:= FindMax / dSum;

gipotesa1D:= false;

if G < 0.2926 then

gipotesa1D:= true;

if gipotesa1D = false then

begin

ShowMessage ('Дисперсии не однородны');

end;

end;

procedure CalculatedSo;

var

i: byte;

begin

dSo:= 0;

for i:= 1 to planeNum do

dSo:= dSo + dSu[ i ];

dSo:= dSo / planeNum;

end;

procedure CalculateRegMean;

var

i: byte;

begin

dSbi:= dSo / planeNum / expNum;

for i:= 0 to factorNum do

tCritery[ i ]:= abs( regCoeficient[ i ] ) / sqrt( dSbi );

end;

procedure MakeDecision;

var

i: byte;

begin

for i:= 0 to factorNum do

begin

decisionRegMean[ i ]:= false;

if tCritery[ i ] > 2.26 then

decisionRegMean[ i ]:= true;

end;

end;

function CalculateL: byte;

var

i: byte;

xResult: byte;

begin

xResult:= 0;

for i:= 0 to 6 do

if decisionRegMean[ i ] then

inc( xResult );

CalculateL:= xResult;

end;

procedure CalculateYExp;

var

xResult: real;

level, i: byte;

begin

for level:= 1 to planeNum do

begin

xResult:= 0;

for i:= 0 to 3 do

xResult:= xResult + regCoeficient[ i ] \* planingMatrix[ i, level ];//xU;

yExpResult[ level ]:= xResult;

end;

end;

procedure CheckRegAd;

var

i: byte;

begin

CalculateYExp;

dSad:= 0;

for i:= 1 to planeNum do

dSad:= dSad + sqr( yAverage[ i ] - yExpResult[ i ] );

dSad:= dSad \* expNum / ( planeNum - CalculateL );

fP:= dSad / dSo;

regAd:= false;

if fP < fisherCritery[ planeNum - CalculateL ] then

regAd:= true;

if regAd = false then

begin

ShowMessage( 'Ренресионная модельне адекватна' );

end;

end;

procedure FillPlaneMatrix;

var

i, j: byte;

begin

FMain.SGridPlaneMatrix.Cells[ 0, 0 ]:= 'x0';

FMain.SGridPlaneMatrix.Cells[ 1, 0 ]:= 'x1';

FMain.SGridPlaneMatrix.Cells[ 2, 0 ]:= 'x2';

FMain.SGridPlaneMatrix.Cells[ 3, 0 ]:= 'x3';

for i:= 0 to factorNum do

for j:= 1 to planeNum do

FMain.SGridPlaneMatrix.Cells[ i, j ]:= FloatToStr( planingMatrix[ i, j ] );

end;

procedure FillExpMatrix;

var

i, j: byte;

begin

for i:= 1 to expNum do

FMain.SGridY.Cells[ i, 0 ]:= IntToStr( i );

for i:= 1 to planeNum do

FMain.SGridY.Cells[ 0, i ]:= IntToStr( i );

for i:= 1 to expNum do

for j:= 1 to planeNum do

FMain.SGridY.Cells[ i, j ]:= FloatToStrF( expResult[ j, i ], ffFixed, 6, 3 );

end;

procedure FillYAverage;

var

i: byte;

begin

for i:= 0 to 7 do

FMain.SGridYAverage.Cells[ i, 0 ]:= FloatToStrF( yAverage[ i + 1 ], ffFixed, 6, 3 );

end;

procedure FillRegCoeficient;

var

i: byte;

begin

for i:= 0 to 3 do

FMain.SGridRegCoef.Cells[ i, 0 ]:= FloatToStrF( regCoeficient[ i + 1 ], ffFixed, 6, 3 );

end;

procedure FillDSu;

var

i: byte;

begin

for i:= 0 to 9 do

FMain.SGridDSu.Cells[ i, 0 ]:= FloatToStrF( dSu[ i + 1 ], ffFixed, 6, 3 );

end;

procedure FillExpMistake;

begin

FMain.LblExpMistake.Caption:= 'Ошибка опыта: ' + FloatToStr( dSo );

end;

procedure FillRegCoeficientCritery;

var

i: byte;

begin

for i:= 0 to 3 do

if decisionRegMean[ i ] then

FMain.SGridCritery.Cells[ i, 0 ]:= 'значим'

else

FMain.SGridCritery.Cells[ i, 0 ]:= 'не значим'

end;

procedure FillYExp;

var

i: byte;

begin

for i:= 0 to 7 do

FMain.SGridYExp.Cells[ i, 0 ]:= FloatToStrF( yExpResult[ i + 1 ], ffFixed, 6, 3 );

end;

procedure FillFCritery;

begin

FMain.LblFCritery.Caption:= 'Критерий Фишера: ' + FloatToStr( fP );

end;

procedure FillresultTables;

begin

FillPlaneMatrix;

FillExpMatrix;

FillYAverage;

FillRegCoeficient;

FillDSu;

FillExpMistake;

FillRegCoeficientCritery;

FillYExp;

FillFCritery;

end;

procedure TFMain.Exit1Click(Sender: TObject);

begin

Close;

end;

procedure TFMain.SpeedButton1Click(Sender: TObject);

begin

MakeExperiment;

CalculateRegCoeficients;

CalculatedSu;

Check1D;

CalculatedSo;

CalculateRegMean;

MakeDecision;

CheckRegAd;

FillResultTables;

end;

procedure TFMain.FormCreate(Sender: TObject);

begin

FillPlaneMatrix;

end;

end.

unit ffe\_typeconst;

interface

const

planingMatrix: array[ 0..3, 1..8 ] of shortint =

( ( +1, +1, +1, +1, +1, +1, +1, +1 ),

( +1, +1, +1, +1, -1, -1, -1, -1 ),

( +1, +1, -1, -1, +1, +1, -1, -1 ),

( +1, -1, +1, -1, +1, -1, +1, -1 )

);

fisherCritery: array[ 1..6 ] of real =

( 5.12, 4.26, 3.86, 3.63, 3.48, 3.37 );

p: array[ 0..3 ] of real = ( 1, 2, 0.5, -1 );

u: array[ 0..3, 1..2 ] of shortint =

( ( 1, 1 ),

( -5, 10 ),

( -7, 2 ),

( 2, 13 ) );

mF: real = 0;

dF: real = 0.8;

expNum: byte = 10;

planeNum: byte = 8;

factorNum: byte = 3;

c1: real = 1.2;

c2: real = -0.8;

var

expResult: array[ 1..8, 1..10 ] of real;

yAverage: array[ 1..8 ] of real;

yExpResult: array[ 1..8 ] of real;

regCoeficient: array[ 0..3 ] of real;

tCritery: array[ 0..3 ] of real;

dSu: array[ 1..8 ] of real;

dSo: real;

dSbi: real;

dSad: real;

fP: real;

G: real;

gipotesa1D: boolean;

regAd: boolean;

decisionRegMean: array[ 0..3 ] of boolean;

implementation

end.

program ffe;

uses

Forms,

main in 'main.pas' {FMain},

ffe\_typeconst in 'ffe\_typeconst.pas';

{$R \*.RES}

begin

Application.Initialize;

Application.CreateForm(TFMain, FMain);

Application.Run;

end.