Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования

Владимирский государственный университет

Н.Н.Барабанов, В.Т.Земскова

РАСЧЕТЫ ХИМИКО-ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ В СИСТЕМЕ MATLAB

Учебное пособие

Владимир 2011

УДК 66.01:51.001.57(075.8)

Рецензенты:

Кандидат технических наук, профессор кафедры экологии
Владимирского государственного университета
Н.А.Андрианов

Печатается по решению редакционного совета
Владимирского государственного университета

Барабанов Н.Н., Земскова В.Т.
Расчеты химико-технологических процессов в системе MATLAB: учеб. пособие / Н.Н.Барабанов, В.Т.Земскова; Владим. гос. ун-т. – Владимир: Изд-во Владим. гос. ун-та, 2011.
ISBN

Учебное пособие предназначено для студентов специальностей 240502 «Технология переработки пластмасс и эластомеров», 240304 «Химическая технология тугоплавких неметаллических и силикатных материалов, 240100 «Химическая технология и биотехнология» (бакалавры) по трем дисциплинам «Применение ЭВМ в химической технологии», «Математическое моделирование ППП» и «Системы управления ХТП».

Излагается материал по вопросам программирования в системе MATLAB, приводятся примеры расчета конкретных задач химической технологии.

Приведенный в пособии материал может быть использован на практических занятиях, при выполнении лабораторных работ, при курсовом и дипломном проектировании

Ил. 26. Табл. 5. Библиогр.: 3 назв.

ISBN

ВВЕДЕНИЕ

MATLAB – это высокопроизводительный язык для технических расчетов. Он включает в себя вычисления, визуализацию и программирование в удобной среде, где задачи и решения выражаются в форме, близкой к математической. *MATLAB* используется для:

- математических вычислений;
- создания алгоритмов;
- моделирования;
- анализа данных, исследования и визуализации;
- инженерной графики.

MATLAB – это интерактивная система. Язык *MATLAB* (*M*-язык) – это высокопроизводительный язык технического программирования. Основным элементом *M*-языка является не число, а массив (одно-, двух- и т.д. мерные массивы) то есть матрица. *M*-язык является языком команд, представляющих собой готовые алгоритмы тех или иных вычислений. Например, одной командой можно построить график сложной функции или решить систему линейных уравнений.

В языке системы *MATLAB* программы называют *M*-файлам и делят их на два типа: файлы сценария (или *Script*-файлы) и файлы-функции.

Создание и отладка *M*-файлов осуществляется при помощи текстового редактора *Debugger*, который вызывается посредством открытия чистой страницы или через команду *File+New+M-file* в командном окне *MATLAB*. Все файлы, создающиеся в системе *MATLAB* имеют расширения «имя«.*m*».

Внешним отличием Script-файла от файла-функции является то, что файл-функция всегда должна начинаться сто строки вида:

function[«МВД»]= «имя функции»(«МВХД>),

где МВД – массив выходных данных; МВХД – массив входных данных; «имя функции» – имя, которое используется при обращении к данной функции. Имя файла должно совпадать с именем функции.

При разработке *M*-файлов следует выполнять следующее:

- первая строка в *Script*-файле должна начинаться комментарием, который обозначается символом «%», который устанавливается в первой позиции командной строки. Эта строка является невыполняемой, в отличие от строк, содержащих операции;
- каждый оператор, для наглядности, рекомендуется записывать в отдельной строке текста *M*-файла, в конце оператора ставится символ «;»;
- длинный оператор, если он не вмещается в одну строку, можно переносить, при этом предыдущая строка должна заканчиваться тремя или более точками «….»;
- операторы в *M*-языке набираются строчными английскими буквами и должны начинаться с буквы;
- в *M*-языке нет оператора окончания текста программы;
- для вызова всего текста программы в окно необходимо набрать команду *type* «имя файла»;
- набранную программу необходимо сохранить, открыв в панели меню *File+Save as*; эту операцию следует выполнять после редактирования;
- разработанная программа в виде *Script*-файла может быть запущена на выполнение или непосредственно из редактора-отладчика через меню *Tools+Run* или через буфер обмена загрузкой его в рабочую область *MATLAB*. Если имеются синтаксические ошибки система *MATLAB* выдает сообщение об ошибках;
- в файлах-функциях строка комментария записывается после определения функции

  function y=summ(x);

  %Файл-функция для расчета…

1. ГРАФИЧЕСКАЯ ВИЗУАЛИЗАЦИЯ ЭКСПЕРИМЕНТАЛЬНЫХ И РАСЧЕТНЫХ ДАННЫХ

1.1. Одномерная графика

Построение графиков функций одной переменной и их оформление осуществляется с помощью функций, приведенных в табл. 1.

Таблица 1

Функция	Назначение	Форма записи
Plot	а) Построение графика в декартовых координатах б)Построение нескольких графиков на одном рисунке	*plot(x,y)* *plot(x1,y1,x2,y2,...)* Размерности *x1, y1,z2,y2,...* должны быть одинаковы
subplot	Расположение нескольких графиков в одном окне	*subplot(m,n,k)* *m* – число рядов подокон, *n* – число колонок в ряду, *k* – номер подокна в ряду
Polar	Построение графика в полярных координатах	*polar(teta, A), teta* – угол; *A* – модуль
Title	Заголовок графика	*title*(‘график функции’)
xlabel *ylabel*	Определение названия осей x и y	*xlabel(‘ось x’);* *ylabel(‘ось y’)*
grid on (off)	Нанесение координатной сетки (или отключение ее)	*grid on* *grid off*
Figure	Функция сохранения графиков в различных окнах; ставится перед вызовом функции plot	*Figure*
Gtext	Нанесение подписей на график с помощью мыши	*gtext*(‘любой текст’) *gtext(*‘функция sin *x*’) и т.д.
маркеры точек ‘•’,’*’,’0’,’+’	Маркер точек, который задает тип точки в узлах; тип маркера записывается в команде plot	*plot(x,y,’*’)*
Bar	Столбиковая диаграмма массива	*bar(x)*
Hist	Построение гистограммы заданного вектора	*hist(y,x); y* – вектор, по которому строится гистограмма; *x* – интервал на котором строится гистограмма с заданной шириной столбцов
Comet	Динамическое построение графиков	*comet(x,y)*

Пример 1. Построить графики функций $y_1=sin(x)$, $y_2=cos(x)$ на одном графике, командой *plot* при одинаковой размерности массивов расчетных данных

```
x=0:0.1:3*pi; % задание вектора аргумента x с помощью оператора «:»
y1=sin(x); %расчет функции y1
y2=cos(x); % расчет функции y2
plot(x,y1,x,y2);grid on; % построение двух графиков в одном окне
ylabel ('ось y');
xlabel ('ось x');
gtext (' функция y1');
gtext (' функция y2');
gtext ('графики функций y1=sin(x), y2=cos(x)');
```

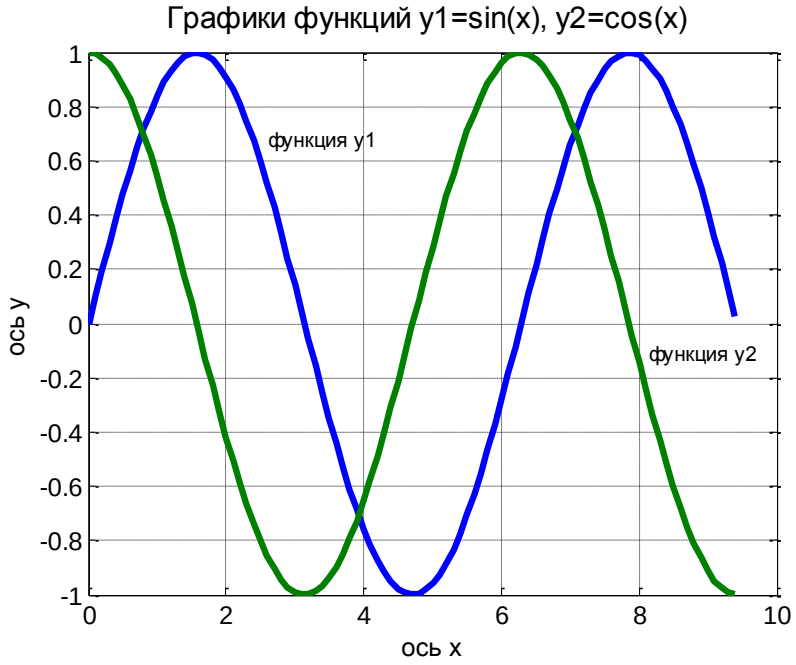


Рис. 1. Результаты построения графиков

Пример 2. Построить график функций при различных интервалах изменения аргументов командой *subplot.*

```
t1=0:0.1:5; t2=0:0.5:10;
y1=1-exp(-t1);
y2=(1-exp(-t2).*cos(t2));
subplot (1,2,1); plot (t1,y1); grid on;
subplot (1,2,2); plot (t2,y2); grid on;
```

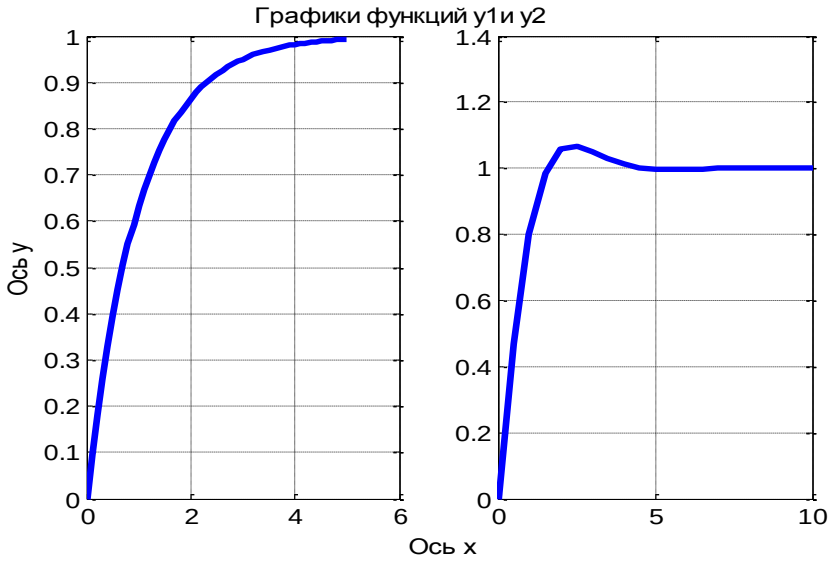


Рис. 2. Результаты построения графиков через функцию *subplot()*

1.2. Трехмерная графика

В системе *Matlab* для построения объемных графиков предусмотрены три функции *plot3(x,y,f), mesh(x,y,f), surf(x,y,f)*.

Перед использованием этих команд необходимо по заданным векторам *x* и *y* создать матрицы *"XX"* и *"YY"*, состоящие из повторяющихся строк и столбцов *x* и *y* координат.

Эту операцию создает функция *meshgrid*, которая записывается в следующем формате:

[xx,yy]=meshgrid(x,y).

Script-файл построения графиков функции *f* с использованием команд *mesh()* и *surf()*:

```
%Пример: построить графики функций вида:
%r=sqrt(x.^2+y.^2); f=sin(r)./r;% при изменении x и y в интервале (-10 до +10)
x=[-10:0.513:10]; y=x; [x,y]=meshgrid(x,y);
r=sqrt(x.^2+y.^2);
f=sin(r)./r;
mesh(x,y,f);xlabel('x');ylabel('y');zlabel('f');box;
title('Построение объемного графика командой mesh');
figure;
```

```
surf(x,y,f); grid on;
xlabel('x');ylabel('y');zlabel('f');
box;title('Построение объемного графика командой surf');
figure;
% Построение контурных линий
c=contour(x,y,f,5);grid on;
clabel(c);
title('Контурные линии');
xlabel('x');ylabel('y');
```

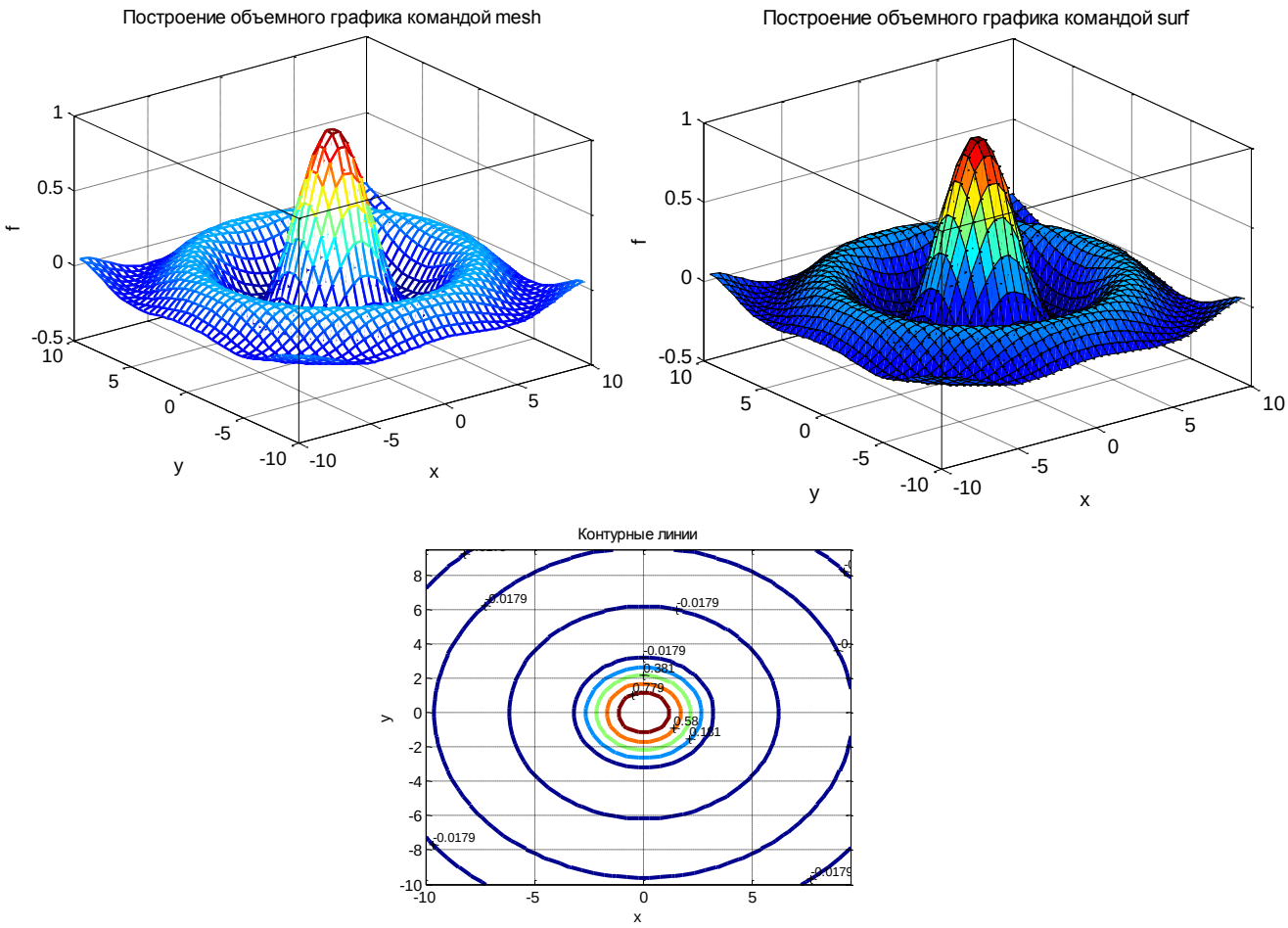


Рис. 3. Результаты построения объемных графиков

1.3. Специальные средства графики

Обработка данных в графическом окне

В позиции *Tools* графического окна имеется две команды для обработки данных графиков прямо в графическом окне:

Basic Fitting – основные виды аппроксимации (регрессии);

Data Statistics – статистические параметры данных.

Команда *Basic Fitting* открывает окно, дающее доступ к ряду видов аппроксимации и регрессии: сплайновой, эрмитовой и полиномиальной со степенями от 1 (линейная аппроксимация) до 10. В том числе со степенью (квадратичная аппроксимация) и 3 (кубическая аппроксимация). Команда *Data Statistics* открывает окно с результатами простейшей статистической обработки данных.

Пусть некая зависимость *y(x)* задана векторами координат ее точек:

```
>>X=[2,4,6,8,10,12,14];
>> Y=[3.76,4.4,5.1,5.56,6,6.3,6.7];
>> plot(X,Y,'o')
```

На рис. 4 показан пример выполнения полиномиальной регрессии (аппроксимации) для степеней полинома 1, 2 и 3. Иными словами, выполняется линейная, параболическая и кубическая регрессия.

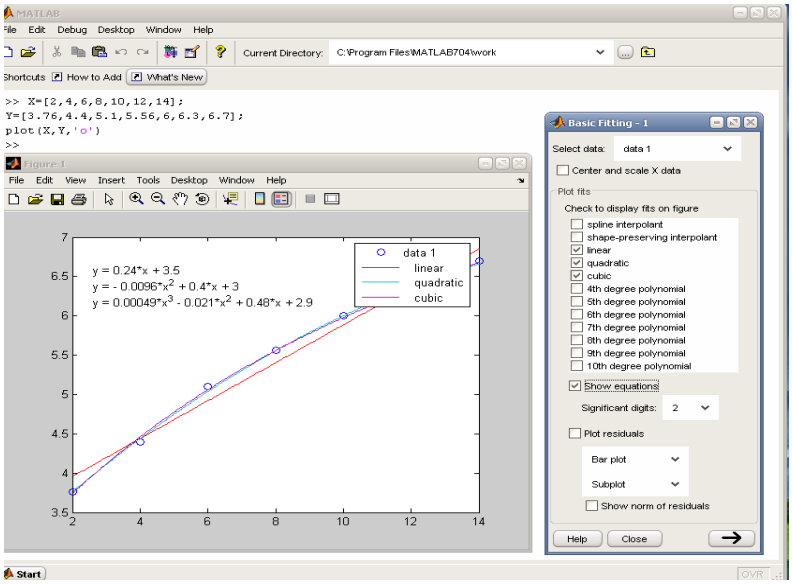


Рис. 4. Пример обработки табличных данных в графическом окне

ВНИМАНИЕ *При проведении полиномиальной аппроксимации надо помнить, что максимальная степень полинома на 1 меньше числа точек, то есть числа элементов в векторах X и У.*

Поясним, что же показано на рис.4. В левом верхнем углу сессии *MATLAB* видна запись исходных векторов и команды построения заданных ими точек кружками (окно слева). Исполнив команду *Tools* ► *Basic Fiting*, можно получить окно регрессии (оно показано справа). В этом окне птичкой отмечены три упомянутых выше вида полиномиальной регрессии. Установка птички у параметра Show equations выводит в графическом окне записи уравнений регрессии.

По команде *Tools* ► *Data Statistics* выводится окно с рядом статистических параметров для данных, представленных векторами *X* и *У*. Отметив птичкой тот или иной параметр в этом окне (оно показано на рис. 5 под окном графики), можно наблюдать соответствующие построения на графике, например вертикалей с минимальным, среднем, срединным, и максимальным значением *y* и горизонталей с минимальным, среднем, срединным, и максимальным *x*.

Оценка погрешности аппроксимации

Средства обработки данных из графического окна позволяют строить столбиковый или линейчатый графики погрешностей в узловых точках и наносить на эти графики норму погрешности. Норма дает статистическую оценку среднеквадратической погрешности, и чем она меньше, тем точнее аппроксимация. Для вывода графика погрешности надо установить птичку у параметра *Plot residuals* (График погрешностей) и в меню ниже этой опции выбрать тип графика – смотри рис. 6. На рис. 6 приведены данные по полиномиальной аппроксимации степени 1, 2, 3 и 6. Последний случай предельный, поскольку максимальная степень полинома должна быть на 1 меньше числа точек (их 7). В этом случае регрессия вырождается в обычную (без статистической обработки) полиномиальную аппроксимацию. При ней линия графика аппроксимирующей функции точно проходит через узловые точки, а погрешность в них равна 0 (точнее, ничтожно мала).

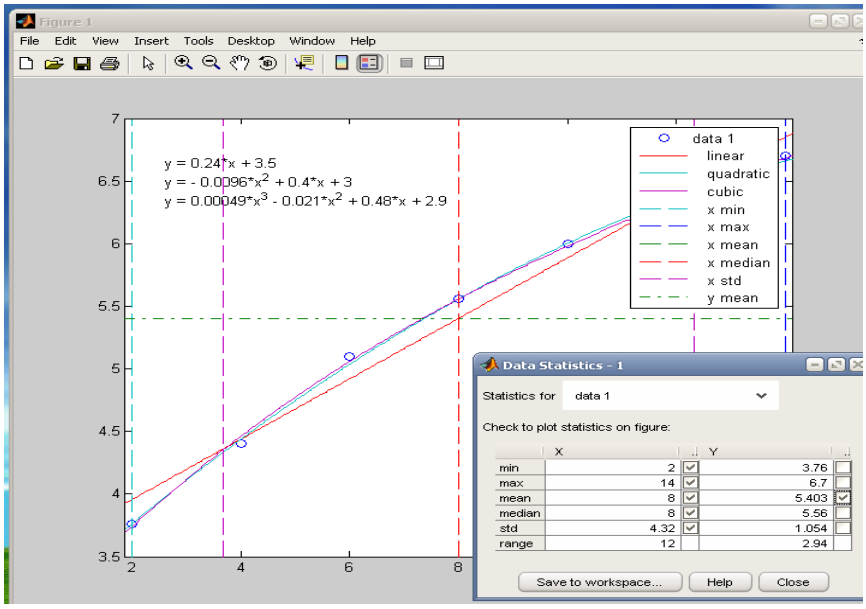


Рис. 5. Пример получения статистических данных о графике

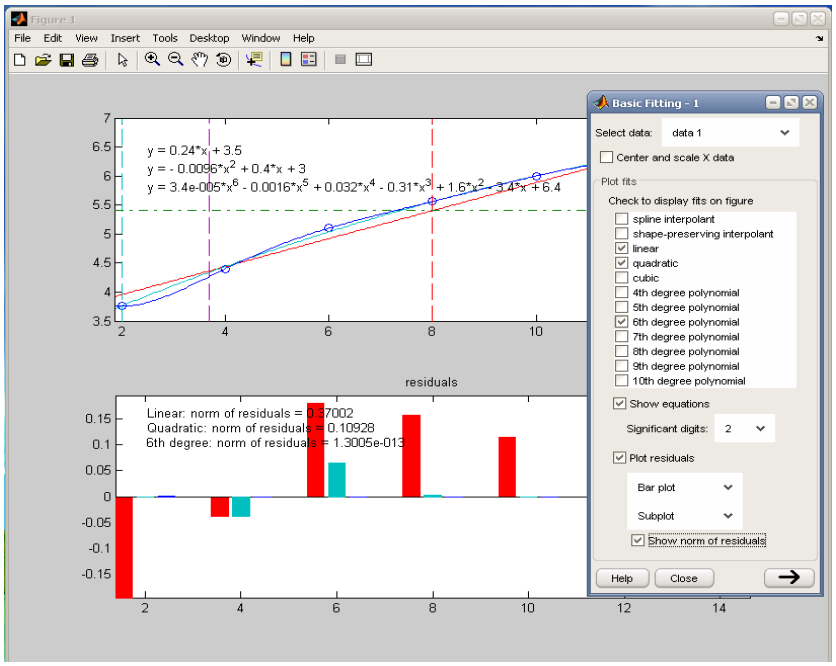


Рис. 6. Пример вывода данных обработки со столбцовым графиком погрешности

Рис. 7 демонстрирует построение графика погрешности отрезками линий. Кроме того, опцией *Separate figure* (Разделить фигуры) задано построение графика погрешности в отдельном окне – оно расположено под графиком узловых точек и функций аппроксимации.

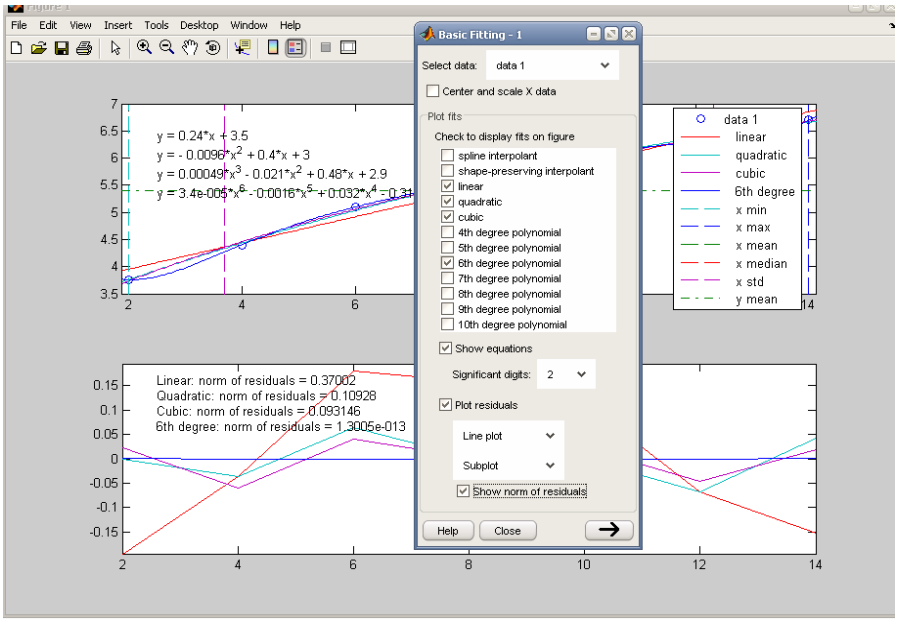


Рис. 7. Пример обработки табличных данных с выводом графиков погрешностей в отдельном окне

Таким образом, интерфейс графического окна позволяет выполнять эффективную обработку данных наиболее распространенными способами.

Расширенные возможности окна

На рис. 7 окно приближения кривых *Basic Fitting* представлено в упрощённом виде. В левом нижнем его углу можно заметить кнопку с жирной стрелкой ►, указывающая на возможность расширения окна до двух и даже трёх панелей. На рис.8 показана расширенная до трёх панелей окно *Basic Fitting.*

Первая панель для задания и типа приближения и вывода данных о погрешности уж была описана. Вторая панель *Numerical Results* (численные результаты) содержит список приближений в котором можно задать выбранное приближение, например, линейное, если задана позиция списка *linear*. После выбора типа приближения для него выводятся выражения для приближения, значения коэффициентов и значения нормы.

В третьей панели *Find Y=f(X)* для выбранного приближения кривой можно найти значения *Y* по заданным значениям *X*. Соответ-

ствующие точки *Y(X)* помещаются на графике жирными ромбами – см. рис.9. Пример дан для задания вектора *X*=2.5:2.5:14.

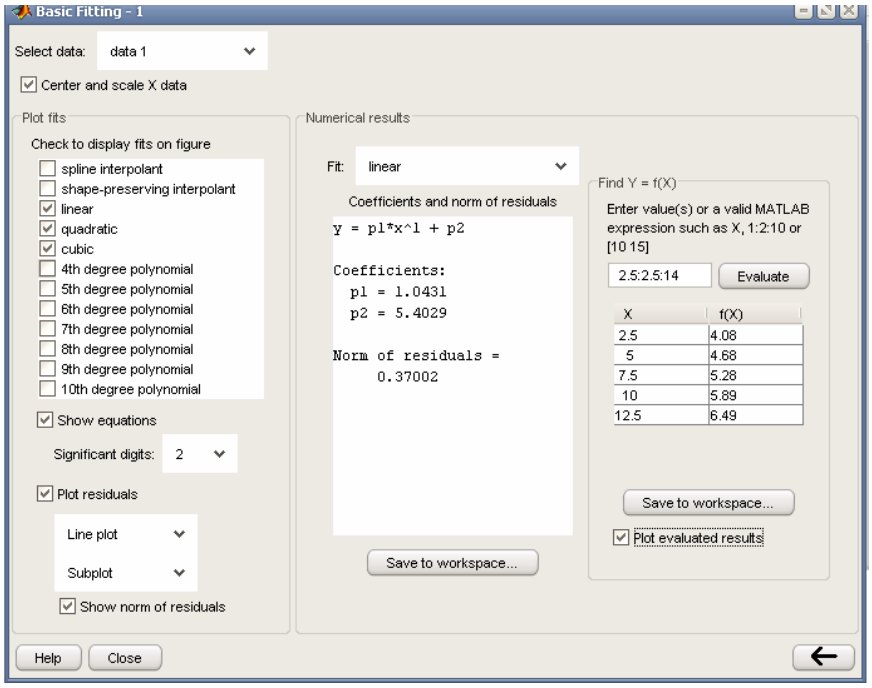


Рис. 8. Окно Basic Fitting с тремя панелями

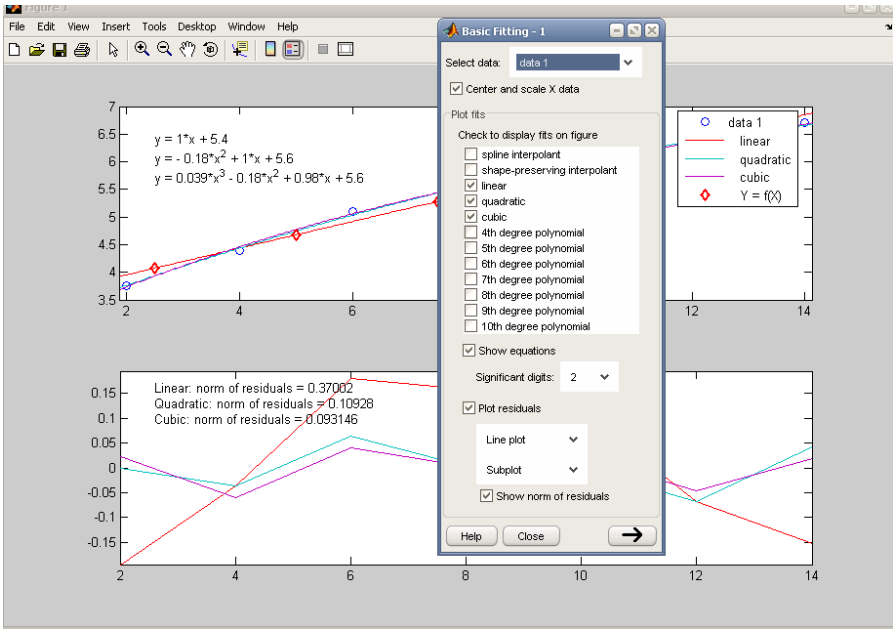


Рис. 9. Результаты приближения с указанием заданных точек графика линейного приближения и выводом данных о погрешности

2. МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ СИСТЕМЫ *MATLAB*

2.1. Математические операторы

Командная строка начинается со знака », после которого можно записать любое выражение в соответствии с математическими операциями, приведенными в табл. 2.

Таблица 2.

Операция	Описание
a+b	Сложение
a-b	Вычитание
a=b	Оператор присваивания
*a.*b*	Поэлементное умножение
*a*b*	Матричное умножение
a./b	Правое поэлементное деление
a.\b	Левое поэлементное деление
a/b	Правое матричное деление
a\b	Левое матричное деление
a.^b	Поэлементное возведение в степень
a^b	Матричное возведение в степень
a<b	Меньше
a>b	Больше
a<=b	Меньше или равно
a>=b	Больше или равно
a~=	Не равно
a==b	Равно
a.'	Транспонирование
[a b]	Горизонтальная конкатенация (объединение)
[a;b]	Вертикальная конкатенация

В системе *MATLAB* имеются стандартные элементарные функции вещественного аргумента, краткий перечень которых приведен в табл. 3.

Таблица 3

Обозначение	Описание
a^x	степенная функция
x^a	показательная функция
Sqrt(x)	квадратный корень
Exp(x)	Экспонента
Log(x)	натуральный логарифм
log10(x)	десятичный логарифм
Abs(x)	Модуль
Round(x)	обычное округление по правилам математики
sign(x)	знак числа
Log2(x)	логарифм по основанию 2
Sin(x)	Синус
cos(x)	Косинус
Tan(x)	Тангенс
cot(x)	Котангенс
Atan(x)	Арктангенс
Acot(x)	Арккотангенс
Perms(a)	вычисляет число перестановок
Системные переменные	
I, j	мнимая единица
pi	число Пи (3,14159265)
Eps	погрешность (по умолчанию=2^{-52})
realmin	минимальное значение вещественного числа (2^{-1022})
realmax	максимальное вещественное число (2^{+1023})
Inf	бесконечность (∞)
NaN	неопределенность (0/0,∞/∞)
Ans	переменная, хранящая результат последней операции

2.2. Управляющие операторы

M-язык содержит следующие операторы:

Оператор присваивания **(=)** записывают в следующей форме: имя_переменной=«выражение».

Условный оператор *if…end*. Его конструкция может иметь вид:

а) *if*<условие> <действие>; *end*;

б) *if…else…end*;

в) *if*<условие 1> <действие 1>

elseif <условие 2>

<действие 2>

elseif <условие 3>

<действие 3>

…… и т.д.

else <действие>

end.

Оператор переключения switch работает следующим образом: переход к <действию 1> будет в том случае, если указанная переменная равняется заданному <значению 1>; переход к <действию 2> , если она равна <значению 2> и т.д. Если переменная не совпадает ни с одним значением, то выполняется <действие>. Конструкция оператора имеет вид:

switch <переменная>

case <значение 1>

<действие 1>

case <значение 1>

<действие 2>

……

Otherwise <действие> *end*.

Оператор цикла с предусловием *while…end*

while <условие>

<действие>

end.

Оператор цикла с параметром *for…end*

for <имя переменной>=<НЗ>:<Ш>:<КЗ>

<действие> *end*,

где НЗ, Ш, КЗ – начальное значение, шаг, конечное значение управляющей переменной.

2.3. Числовые массивы

Одномерные массивы могут быть представлены в виде вектор-строки или в виде вектор-столбца. Элементы в вектор-строке разделяются запятыми или пробелами, в вектор-столбце – точкой с запятой и записываются в квадратных скобках.

Пример**:** *a*=[4,5,6] – вектор-строка; *b*=[4;5;6] – вектор-столбец.

Определение числа элементов в одномерном массиве находится функцией *length* (имя). Например: *length* (a).

Одномерный массив можно сформировать с помощью операции двоеточие. Форма записи имеет вид: *c*=[нз:ш:нк], где нз, ш, нк – начальное значение, шаг изменения, конечное значение. Массив создается в виде вектор-строки. Если шаг не указывается, то по умолчанию он равен единице.

Пример: сформировать одномерный массив от 1 до 10 с шагом 0,5: *t*=[1:0.5:10].

В двухмерных массивах элементы в строках разделяются пробелами или запятыми, строки – точкой с запятой: *c*=[4,5;7,9;3,1]

Размер созданного массива (матрицы) можно узнать с помощью функции [*m,n*]=*size(c),*где *m* – число строк, *n* – число столбцов, *c* – матрица: *[m,n] = size(c).*

Результатом работы этой функции будет два числа: первое показывает число строк *m*=3, второе – число столбцов *n*=2.

Многомерные числовые массивы – это массивы с размерностью больше двух. Например, можно создать массив с помощью оператора двоеточие. *MATLAB*-программа будет выглядеть следующим образом:

```
x=[1:4;5:8;9:12];
disp(' Матрица  x=');disp(x);
disp('Размер матрицы  ');disp( size(x));
```

```
 Матрица  x=  1    2    3    4
              5    6    7    8
              9   10   11   12
```

Размер матрицы: 3, 4, где *disp* – вывод информации на экран.

Вызов элементов 2-ой строки с использованием знака двоеточия:

A=*x*(2,:)

ans=

5 6 7 8

Аналогично вызов 2-го столбца

B=*x*(:,2);

B=

2

6

10

Замена любого элемента, строки, столбца массива.

Пример: Необходимо заменить в выше представленной матрице *x* второй столбец на [4; 5; 6;].

x(:,2)= [4;5;6;]

Удаление любого элемента, строки, столбца массива осуществляется с помощью пустого оператора – [].

Пример: необходимо удалить первую строку матрицы x:

x(1,:)=[]. В результате получим новую матрицу *x*1, в которой останется две строки и три столбца:

*x*1= [5 6 7 8

9 10 11 12]

Объединение матриц – конкатенация (вертикальная). Пример: объединить матрицы *x* и *x*1:

[*x*1;*x*]

ans = 5.00 6.00 7.00 8.00

9.00 10.00 11.00 12.00

1.00 2.00 3.00 4.00

5.00 6.00 7.00 8.00

9.00 10.00 11.00 12.00

Вычисление ранга и определителя квадратной матрицы. Функция *rank (A)* вычисляет ранг матрицы *A*.

Пример. Вычисление ранга матрицы:

a=[1 2 3; 5 1 4; 0 1 6];

r=rank(a);

disp ('Ранг матрицы *a*='); disp(r); % вывод данных на экран;

Функция *det(a)* вычисляет определитель квадратной матрицы *a*, методом Гаусса.

Пример. Вычисление определителя матрицы *a*

a=[1 2 3; 5 1 4;0 1 6];

d= det(a);

disp('Определитель матрицы *a*=');*disp(d);*

disp('Определитель матрицы *a*=');*disp(d);*

Определитель матрицы *a*= -43.00.

Нахождение максимального (минимального) элемента массива осуществляется командами *max()* и *min().* Формат обращения к командам:

- *Amax=max(a), Amin=min(a)* , если *a* - вектор, находится максимальный или минимальный элемент в числовом массиве; если *a* – матрица, находится вектор-строка максимальных или минимальных элементов в каждом столбце;
- *Amax=max(max(a)), Amin=min(min(a))* – находится максимальный или минимальный элемент матрицы *a*.

Нахождение среднего значения массива осуществляется командой *mean(),* формат обращения к которой имеет вид:

AS=mean(a) – если *a* - вектор, находится среднее значение элементов в числовом массиве; если *a* – матрица, находится вектор-строка средних значений элементов в каждом столбце;

Нахождение средне-квадратичного отклонения элементов массива осуществляется командой *std*(). Формат обращения аналогичен команды *mean().*

Примеры:

1. Числовой массив задан в виде вектора *a*=[1 3 7 4 8 10 2];

Amax=max(a); disp('Amax='); disp(Amax); *Amax*=10.

2. Числовой массив задан в виде матрицы

a = 8.00 1.00 6.00
3.00 5.00 7.00
4.00 9.00 2.00

Найти максимальные значения элементов матрицы по столбцам *Amax=max(a)*

Amax = 8.00 9.00 7.00

3. Найти максимальное значение в матрице *a*

Amax=max(max(a))

Amax = 9.00.

4. Найти среднее значение матрицы *a* по столбцам:

A=mean(a)

A = 5.00 5.00 5.00

Нахождение произведений или сумм элементов массива осуществляется командами *prod()* и *sum()* соответственно, формат обращения к которым аналогичен рассмотренному выше.

Пример: найти произведение элементов матрицы *a* по столбцам

prod(a)

ans = 96.00 45.00 84.00.

2.4.Работа с полиномами

Определение корней полинома.

Команда *roots(p)* – находит корни полинома любого порядка вида $p=c_1x^n+c_2x^{n-1}+...+c^n$, причем значение корней *p* располагаются в вектор-строку.

Пример: %*Script-file* для нахождения корней полинома вида

$p=4x^4-7x^2+11x-1$

p=[4 0 -7 1]; %коэффициент полинома: при x^3 равен 0.

k= roots(p);

disp ('корни полинома='); *disp (k);*

корни полинома= -1.3 1.24 0.14

Следует отметить, что функция *roots(p)* находит как вещественные, так и комплексные корни полинома.

Умножение полиномов *p* и *q* осуществляется командой *conv(p,q),* которая находит коэффициенты нового полинома *c*.

Пример.

Вычислить произведение 2-х полиномов

$p=3x^3-7x^2+x+1$, $q=x^2+x-3$.

Решение:

p=[3 -7 1 1]; *q*=[1 1 -3];

c= conv(p,q);

disp ('коэффициенты полинома *c*='); *disp(c);*

коэффициенты полинома *c*=

3.00 -4.00 -15.00 23.00 -2.00 -3.00

Результатом является полином 5-ой степени вида:

$$c=3x^5-4x^4-15x^3+23x^2-2x-3$$

В математике произведение двух полиномов называют сверткой векторов.

Деление полиномов *p/q* осуществляется командой *deconv (p,q).* Формат обращения к этой функции имеет вид

[c,r]= deconv (p,q),

где *c* – полином частное, *r* – полином остаток.

Пример.

p=[3 -7 1 1]; *q*=[1 1 -3]; *[c, r]=deconv(p,q);*

c= 3 –10

r= 0 0 20 –29.

Функция *polyval(p,x)* осуществляет вычисление значений полинома по заданному значению аргумента *x; p* – заданный вектор коэффициентов; *x* – заданный вектор аргумента.

Пример.

p=[3 -7 1 1]; x=[0:1:3];

pr=polyval(p,x)

pr = 1.00 -2.00 -1.00 22.00

Функция *polyfit(x, y, n)* – позволяет найти коэффициенты полинома, которым может быть описана функция *y=f(x); n* – порядок полинома.

Пример.

X=[0:0.1:2];

y=3*x*.^2-*x*+5; % расчет значений y

p= polyfit(x, y, 2); % расчетные коэффициенты полинома

disp(p);

Функция *polyder(p)* осуществляет вычисление производной полинома *p* и возвращает вектор коэффициентов полинома, являющейся производной исходного полинома.

Пример.

P=[3 –7 1 1];

d=polyder(p)

d= 9 –14 1

Таким образом производная от полинома *p* будет $p'=9x^2-14x+1$. Если в качестве аргумента функции *polyder* задать векторы коэффициентов двух полиномов *p* и *q*, то есть *polyder(p)*, то результатом выполнения этой функции будет вектор коэффициентов полинома, являющейся производной произведения заданных полиномов.

2.5. Решение систем линейных алгебраических уравнений (СЛАУ)

Для нахождения корней СЛАУ в системе *MATLAB* используются уникальные по своему назначению операции: *x=a\b*; *x = inv (a) ·b*.

Матричная форма записи **СЛАУ** имеет вид:

AX=B,

где *A*– заданная квадратная матрица коэффициентов уравнения;

B – вектор-столбец свободных членов;

X– вектор-столбец искомых корней.

Пример: решить систему уравнений вида:

$$\begin{cases} 2\tilde{o}_1 + \tilde{o}_2 - 3\tilde{o}_3 = 1 \\ \tilde{o}_1 - \tilde{o}_2 + 2\tilde{o}_3 = 18 \\ 7\tilde{o}_1 + 5\tilde{o}_2 + \tilde{o}_3 = 3 \end{cases}$$

Программа решения с использованием перечисленных выше функций будет иметь вид:

A= [2, 1, -3; 1, -1, 2; 7, 5, 1] ;
B = [1; 18; 3];
*X*1=*A**B* ;
X 2= *inv (A)* · *B;*
*disp([X*1 *X*2*]);*

6.71	6.71
-9.02	-9.02
1.13	1.13

Решение алгебраических и трансцендентных уравнений в среде *MATLAB* осуществляется с помощью встроенной функции: *solve ().*

Технология решения уравнений с помощью встроенных функций предельно проста. Рассмотрим ее и приведем примеры.

Функция *solve ()* представляется в следующем виде: *solve (' f (x) ', x)*

где:*' f (x) '* — решаемое уравнение, записанное в одиночных кавычках; *x* — искомое неизвестное.

Уравнение *f (x)* = 0 записывается в произвольной форме. При этом если знак равенства отсутствует, то программа воспринимает уравнение в виде *f (x)* = 0. Аргумент x при решении уравнения можно опустить.

Функция *sums ()*, определяющая имя символьной переменной и обязательная при решении систем уравнений, здесь может быть опущена.

Рассмотрим технологию определения корня уравнения с помощью функции *solve ()* на примере: пусть необходимо решить следующее уравнение*:* $\sin x + x - 1 = 0$. Программа решения уравнения имеет вид:

» *Y = solve (' sin (x) + x - 1 = 0 ')*

После нажатия клавиши «*Enter*» получим следующее решение: $x = 0.510973$.

2.6. Численные методы решения с использованием специальных функций *MATLAB*

1. Вычисление интегралов методом квадратур осуществляется функциями *quad* и *quad8.* Обращение к этим функциям имеет следующий вид:

int=quad(‘имя функции’, *a,b);* или *int=quad8*(‘имя функции’, *a,b);*

где <имя функции> - имя файла, в котором определена интегрируемая функция; *a* и *b* – интервалы изменения аргумента функции.

При решении интегральных уравнений вначале необходимо создать файл-функцию по алгоритму, изложенному выше и сохранить ее в рабочем каталоге под определенным именем (например, *fun*1), в командной строке *MATLAB* выполнить следующую команду *int=(‘fun*1’, *a*,*b*).

Пример 1. Вычислить определенный интеграл

$$y = \int_{-1}^{+1} \frac{1}{\sqrt{1+x^2}} dx;$$

Решение: создать файл-функцию

*function f=fun*1*(x);*

% Нахождение интеграла функцией *quad.*

F=1./*sqrt*(1+*x*.^2);

Этот файл должен быть сохранен под именем *fun*1.*m*. В командной строке *MATLAB* выполним команду:

x=[-1,+1];

int=quad(‘fun1’, x);

disp (‘int=’); disp (int);

int= 1.7627

Пример 2. Вычислить определенный интеграл

$$f = \int_{-10}^{+10} xe^{-x} dx$$

Файл функция имеет вид:

function f=fun2(x);

f=x.*exp(-x);

В командной строке набрать: int=quad8(‘fun2’, [-10,+10]);

2. Нахождение максимумов и минимумов функций, заданных *M*-файлом.

Нахождение минимума осуществляют функции:

- *fminbnd* – находит минимум функции одной переменной, определенной на интервале *a-b*;
- *fminsearch* – находит минимум функции нескольких переменных при заданной начальной точки поиска.

Формат обращения к этим функциям имеет вид:

xmin=fminbnd(‘имя функции’, x_1, x_2);

xmin=fminsearch(‘имя функции’, x_0),

где x_1, x_2 – интервал в котором ищется минимум; x_0 – вектор начальной точки поиска, в окрестности которой ищется локальный минимум.

Для нахождения максимума функций одной или нескольких переменных с использованием перечисленных команд перед оптимизируемой функцией необходимо поставить знак минус.

Примеры.

1.Минимизация функции 2-х переменных *F=f (x, y),* например, нахождение минимальной поверхности *f* заданного объема *V*=3м^3.
Создадим *M*- файл функцию:

```
function f=mins(v);
x=v(1)
y=v(2);
f=2.*(x.*y+3./x+3./y);
```

В командной строке набрать:

```
v=[1.50 1.50];
xmin=fminsearch(@mins, v);
disp('xmin='); disp(xmin).
```

3. Нахождение нулей функции одной переменной. Для этого используется функция *fzero*, формат обращения к которой имеет вид:

n=fzero('имя функции', *x*0),

где *x*0 – заданное значение аргумента, в окрестности которого ищется нуль.

Пример.

Нахождение нуля функции одной переменной

*f(w)=pi/4-atan(5*w);*

Создадим *M*- файл функцию

```
function ff=funff(w);
ff=pi/4-atan(w.*5);
```

В командной строке набрать обращение к *M*-файл функции

```
w1=fzero('funff',[0 pi]);
disp('корень w1=');disp(w1).
```

2.7. Символьные вычисления средствами *MATLAB*

Помимо численных расчетов в системе *MATLAB* можно производить мощные аналитические вычисления, находящиеся в пакете символьных вычислений *Symbolic Math Toolbox*.

Из большого перечня функций (более 200 шт.) можно перечислить следующие:

- алгебраические преобразования и упрощения выражений;
- вычисление пределов;

- вычисление производных;
- вычисление неопределенных интегралов;
- решение систем линейных и нелинейных алгебраических уравнений;
- решение систем обыкновенных дифференциальных уравнений.

Символьным объектом в системе *MATLAB* называют переменную, предназначенную для символьных преобразований.

Рассмотрим некоторые функции символьной математики:

sym – используется при объявлении одной переменной символьной, например:

sym – объявление символьной переменной *x.*

syms – одновременно объявляет несколько переменных символьными, например:

syms x y z; переменные разделяются пробелами.

simplify – упрощает заданное выражение, например:

```
syms x y ;
f= simplify((x^2-y^2)/(x-y));
disp('f='); disp(f);
f=x+y
```

expand – упрощает тригонометрические, логарифмические, экспоненциальные функции.

Формат обращения к этой функции:

Res=expand(S),

где *S* – символьное выражение, которое надо упростить;

res – результат упрощения.

Примеры:

```
 syms x y;
 res1=expand((x-y)^2);
res=x^2-2x*y+y^2
res2= expand(exp(logx))
res2=x
```

diff – дифференцирование символьных функций.

Формат обращения: *diff(y(x))*

Если функция задана неявно и необходимо продифференцировать ее *n* раз, формат обращения имеет вид:

D=diff(s,'v',n),

где *s* – дифференцированное выражение; *v* – переменная, по которой идет дифференцирование*; n* – порядок дифференцирования.

Пример:

```
syms x y;
S=x^3+y^2+sin(x*y);
D=diff(S, 'x', 2); % дважды дифференцированные по x
disp ('D='); disp(D); % ответ.
D=6x*y^2-sin(x*y)*y^2
```

solve – решает системы нелинейных алгебраических уравнений.

Формат обращения к функции *solve* имеет вид:

[x_1,x_2]-solve(уравнение1, уравнение2)

Пример: Найти корни системы уравнений вида:

$x_1^2+x_2^2=5$

$x_1x_2=2$

```
syms x1 x2;
[x1,x2]=solve(x1^2+x2^2-5, x1*x2-2)
```

ответ

x_1=	x_2=
[2]	[1]
[1]	[2]
[-1]	[-2]
[-2]	[-1]

Здесь решением системы уравнений являются первые соответствующих элементов столбцов x_1 и x_2, то есть первым решением будет пара x_1=2, x_2,=1

limit – функция вычисляет предел функции *f(x).* Формат обращения *lim=limit(F,x,n),*

где F – функция, в которой находится предел, x – аргумент функции F; n – конечное значение x.

Например, дана функция:

$$F = \frac{x^3 - 8}{2x - 4}$$

вычислить предел $\lim_{x \to 2} F$

```
% script – файл для нахождения предела f(x)
syms x;
f=(x^3-8)./(2*x-4);
lim=limit(f,x,2);
disp('предел='); disp (lim);
```

ответ lim=6.

Решение систем нелинейных алгебраических уравнений**.**

Команда (или функция) системы *MATLAB solve* позволяет решать нелинейные системы алгебраических уравнений. Формат обращения:

[x1,x2]=solve(f1,f2);

где x_1,x_2 – корни системы, f_1,f_2 – система уравнений.

Пример:

$f_1=x_1-x_2^3$;

$f_2=x_1-8$;

```
% script – файл решения;
 syms x1 x2; % объявление символьных переменных
 [x1,x2]=solve(x1-x2.^3, x1-8);
  x1                    x2
  x1=                   x2=
  [-2]                  [8]
[1-i*3^(1/2)]           [8]
[1+i*3^(1/2)]           [8]
```

Нахождение экстремумов функции.

Из курса математики известно, что для нахождения точки экстремума функции $f_1=f(x)$ нужно решить уравнение $f'(x)=0$. Это уравнение получено используя команду *diff(f1).*

Пример: Найти экстремум функции вида: $f_1=3x^4-4x^3-12x^2+2$:

```
syms x f1;
f1=3*x.^4-4*x.^3-12*x.^2+2;
pf1= diff(f1); %  вычисление производной
x=solve('pf1');
disp('Координата экстремума=');disp(x);
```

Координата экстремума x=0.

Примечание: для определения вида экстремума надо найти знак 2-ой производной в этих точках, если «-» – *max*, если «+» – *min*.

2 .8. Решение систем обыкновенных дифференциальных уравнений решателями *ODE*

Анализ поведения многих систем и устройств в динамике, а также решение многих задач в теории колебаний и в поведении упругих оболочек обычно базируются на решении систем обыкновенных дифференциальных уравнений (ОДУ). Их, как правило, представляют в виде системы из дифференциальных уравнений первого порядка в форме Коши:

$$\frac{dy}{d\tau} = y', y' = f(y, \tau)$$

с граничными условиями $y(\tau_{end}, \tau_0) = b$, где τ_{end}, τ_0 – начальные и конечные точки интервалов. Параметр τ не обязательно означает время, хотя чаще всего решение дифференциальных уравнений ищется во временной области. Вектор b задает начальные и конечные условия. Ниже коротко описаны численные методы решения обыкновенных дифференциальных уравнений (ОДУ) и некоторые вспомогательные функции, полезные для решения систем ОДУ. Дается представление о пакете расширения, решающем дифференциальные уравнения в частных производных.

Для решения систем ОДУ в *MATLAB* реализованы различные методы. Их реализации названы решателями ОДУ, перечень которых приведен ниже:

- *ode*45 — одношаговые явные методы Рунге-Кутта 4-го и 5-го порядка. Это классический метод, рекомендуемый для начальной пробы решения. Во многих случаях он дает хорошие результаты;
- *ode*23 — одношаговые явные методы Рунге-Кутта 2-го и 4-го порядка. При умеренной жесткости системы ОДУ и низких требованиях к точности этот метод может дать выигрыш в скорости решения;
- *ode*113 — многошаговый метод Адамса-Башворта-Мултона переменного порядка. Это адаптивный метод, который может обеспечить высокую точность решения
- *ode*23t*b* — неявный метод Рунге-Кутта в начале решения и метод, использующий формулы обратного дифференцирования 2-го порядка в последующем
- *ode*15*s* — многошаговый метод переменного порядка (от 1 до 5, по умолчанию 5), использующий формулы численного дифференцирования. Это адаптивный метод, его стоит применять, если решатель *ode*45 не обеспечивает решения;
- *ode*23*s* — одношаговый метод, использующий модифицированную формулу Розенброка 2-го порядка. Может обеспечить высокую скорость вычислений при низкой точности решения жесткой системы дифференциальных уравнений;
- *ode*23*t* — метод трапеций с интерполяцией. Этот метод дает хорошие результаты при решении задач, описывающих колебательные системы с почти гармоническим выходным сигналом.

Для решения жестких систем уравнений рекомендуется использовать только специальные решатели *ode* 15*s* , *ode*23*s*, *ode*23*t*, *ode*23*tb*.

В описанных далее функциях для решения систем дифференциальных уравнений приняты следующие обозначения и правила:

- *options* — аргумент, создаваемый функцией *odeset* (еще одна функция — odeget или *bvpget* (только для *bvp*4*c*)— позволяет вывести параметры, установленные по умолчанию или с помощью функции *odeset /bvpset*);
- *tspan* — вектор, определяющий интервал интегрирования [*t0 tfinal*]. Для получения решений в конкретные моменты времени *t0, tl,..., tfinal* (расположенные в порядке уменьшения или увеличения) нужно использовать *tspan = [t0 tl ... tfinal];*
- *y0* — вектор начальных условий;
- *p*1, *p*2,.,, — произвольные параметры, передаваемые в функцию *F;*
- *T, Y* — матрица решений *Y*, где каждая строка соответствует времени, возвращенном в векторе-столбце *T*.

Перейдем к описанию функций для решения систем дифференциальных уравнений:

- [T.Y] = solver(@F,tspan,y0) — где вместо *solver* подставляем имя конкретного решателя — интегрирует систему дифференциальных уравнений вида *y'=F(t,y)* на интервале *tspan* с начальными условиями *y0*. *@F* — дескриптор ODE-функции. Каждая строка в массиве решений *Y* соответствует значению времени, возвращаемому в векторе-столбце *T;*
- [T.Y] = solver(@F, tspan, yO. options) — дает решение, подобное описанному выше, но с параметрами, определяемыми значениями аргумента *options*, созданного функцией *odeset*. Обычно используемые параметры включают допустимое значение относительной погрешности *RelTol* и вектор допустимых значений абсолютной погрешности *AbsTol* ;

- [T.Y] = solver(@F,tspan,yO,options,pl,p2...) — дает решение, подобное описанному выше, передавая дополнительные параметры p1, p2,... в *m*-файл *F* всякий раз, когда он вызывается. Используйте *options*=[], если никакие параметры не задаются;
- [T.Y.TE.YE.IE] = solver(@F.tspan,yO,options) — в дополнение к описанному решению содержит свойства *Events*, установленные в структуре *options* ссылкой на функции событий. Когда эти функции событий от (*t, y*, равны нулю, производятся действия в зависимости от значения трех векторов *value, isterminal, di rection* (их величины можно установить в m-файлах функций событий). Для *i*-й функции событий *value(i)* —значение функции, *isterminal (i)* — прекратить интеграцию при достижении функцией нулевого значения, *direction*^) = 0, если все нули функции событий нужно вычислять (по умолчанию), +1 — только те нули, где функция событий увеличивается, -1 — только те нули, где функция событий уменьшается. Выходной аргумент *TE* — вектор-столбец времен, в которые происходят события (*events)*, строки *YE* являются соответствующими решениями, а индексы в векторе *IE* определяют, какая из *i* функций событий (*event*) равна нулю в момент времени, определенный *TE*. Когда происходит вызов функции без выходных аргументов, по умолчанию вызывается выходная функция *odeplot* для построения вычисленного решения. В качестве альтернативы можно, например, установить свойство *OutputFcn* в значение ' *odephas*2' или '*odephas*3' для построения двумерных или трехмерных фазовых плоскостей.

Пример 1. Покажем применение решателя ОДУ на ставшем классическом примере — решении уравнения Ван-дер-Поля, записанного в виде системы из двух дифференциальных уравнений:

$y'_1 = y_2$;
$y'_2 = 100*(1-y_1)^2 * y_2 - y_1$
при начальных условиях
$y_1(0) = 0;\ y_2(0) = 1.$
Перед решением нужно записать систему дифференциальных уравнений в виде *ode*-функции. Для этого в главном меню выберем *File* > *New* > *M-File* и введем

```
function dydt = vdp100(t.y)
dydt = zeros(2.1); % a column vector
dydt(l) = y(2);
dydt(2) = 100*(1 -y(1^)2)*y(2) -y(1);
```

Сохраним *m*-файл-функцию.
Тогда решение решателем *ode*15*s* и сопровождающий его график можно получить, используя следующие команды, которые набираются в строке *Matlab*

```
» [T,Y]=odel5s(@vdp100.[0 30].[2 0]);
» plot(T.Y)
» hold on;gtext('yl'); gtext('y2')
```

Последние команды позволяют с помощью мыши нанести на графики решений $y_1 = y(1)$ и $y_2 = y(2)$ помечающие их надписи.

Пример 2. Решение системы дифференциальных уравнений с оператором «*for*»

$$\frac{dy(1)}{d\tau} = bT_p - 2by(1) + by(2)$$
$$\vdots$$
$$\frac{dy(i)}{d\tau} = by(i-1) - 2by(i) + by(i+1), i = 2, n-1$$
$$\vdots$$
$$\frac{dy(n)}{d\tau} = bT_p - 2by(n) + by(n-1)$$

При начальных условиях *y(i)*=20, *i*=1,*n* и функции внешнего воздействия *Tp* = 200.

M-файл функция для формирования правых частей исходной системы дифференциальных уравнений с использованием управляющего оператора «*for*» приведена ниже:

```
function dy=dif100(t,y,tp);
n=length(y);
a=1e-5;dl=l/n;b=a/dl^2;
dy(1,1)=b*tp-2*b*y(1,1)+b*y(2,1);
for i=2:n-1
dy(i,1)=b*y(i-1,1)-2*b*y(i,1)+b*y(i+1,1);
end;
dy(n,1)=b*y(n-1,1)-2*b*y(n,1)+b*tp;
```

Вызывающая программа в виде *Script*-файла имеет вид:

```
%Skript_fyle  для решения системы диф.уравнений с передачей параметров tp  и l в  m-fyle функцию  dif100
tp=200;n=10;l=0.05;
for i=1:n
y0(i)=20;
end;
tspan=[0:1:200];
[t,y]=ode15s(@dif100,tspan,y0,[],tp,l);
plot(t,y);grid on;
xlabel('tau');ylabel('y');
```

Многие технологические объекты описываются дифференциальными уравнениями в частных производных. К этим объектам могут быть отнесены процесс прессования, движение материала в экструдерах и литьевых машинах, в стекольной промышленности - нагрев изделий в переменном температурном поле.

Одним из методов решения дифференциальных уравнений в частных производных является метод Коши. Сущность этого метода заключается в следующем: исходное дифференциальное уравнение преобразуется в систему *n* обыкновенных дифференциальных уравнений с заменой непрерывных производных по координатам их конечностно-разностными представлениями.

Например, уравнение нестационарной теплопроводности

$$\frac{\partial T(h,\tau)}{\partial \tau} = a\frac{\partial^2 T(h,\tau)}{\partial h^2}$$

с заданными начальными и граничными условиями преобразуется в систему *n* обыкновенных дифференциальных уравнений

$$\frac{dT_j(\tau)}{d\tau} = \frac{a}{\Delta h^2}(T_{j-1}(\tau) - 2T_j(\tau) + T_{j+1}(\tau)),\ \text{j=1,n,}$$

где a – коэффициент температуропроводности, $\Delta h = H/n$ – размер сетки, H – полная толщина изделия, n – число элементарных слоев. Решение данной системы аналогично решению, приведенному в примере 2.

3. КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ РЕШЕНИЯ ЗАДАЧ УПРАВЛЕНИЯ

3.1. Функции *MATLAB* для создания передаточных функций звеньев системы

При расчете систем управления анализ и синтез проводят с использованием передаточных функций звеньев, входящих в систему управления.

В системе *MATLAB* для формирования передаточных функций имеются следующие функции:

1.Функция *tf(),* формат обращения к которой имеет вид:

W=tf(n,m), где: *n* – вектор коэффициентов числителя передаточной функции, *m* – вектор коэффициентов знаменателя передаточной функции.

Она служит для образования передаточной функции звеньев и системы в целом.

Пример 1.

Необходимо образовать передаточную функцию

$$W(S)=\frac{2S+5}{S^3+2S+1}$$

В нашем случае векторы коэффициентов числителя и знаменателя передаточной функции имеют вид: *n*=[2,5], *m*=[1,0,2,1]. Ноль в векторе *m* ставиться потому, что в знаменателе передаточной функции член S^2 отсутствует.

Процедуры образования передаточной функции *W(S)* имеют вид:

```
>>n=[2,5];
>> m=[1,0,2,1]
>> W=tf(n,m)
```

После нажатия клавиши *ENTER* на экране появиться передаточная функция в виде:

Transfer function:

$$\frac{2s+5}{s^3+2s+1}$$

Функцию *W(S)=tf(n,m)* можно также представить в следующем виде: *>>W(S)=tf*([2 5],[1 0 2 1]). Числа в векторах *n* и *m* отделяются друг от друга либо запятыми, либо пробелами, а сами векторы заканчиваются символом (;). Символ точка с запятой подавляет вывод на экран векторов при нажатии клавиши *ENTER*.

Формирование передаточных функций с запаздыванием.

Формат обращения к функции *tf* () имеет вид

wz=tf(n,m,'inputdelay',tzp),

где *n* – вектор коэффициентов числителя передаточной функции; *m* – вектор коэффициентов знаменателя передаточной функции; *tzp* – время запаздывания.

Пример 2. Необходимо образовать передаточную функцию:

$$Wz = \exp(-10s)\frac{2s^2+12s+15}{s^3+3s^2+7s+5}$$

Решение:

n=[2 12 15]; *m*=[1 3 7 5];*tzp*=10;

*ww=tf(n*2,*m*2,'*inputdelay'*,10)

Transfer function:

$$\exp(-10*s) * \frac{2s^2+12s+15}{s^3+3s^2+7s+5}$$

3.2. Операции с передаточными функциями звеньев

Сложение передаточных функций:

Сложение передаточных функций осуществляется с помощью оператора +.

Пример 3. Сложить передаточные функции

$$G1(S)=\frac{10}{S^2+2S+5} \quad \text{и} \quad G2(S)=\frac{2S^2+12S+15}{S^3+3S^2+7S+5}$$

Решение:

```
>> n1=[10];
>> m1=[1 2 5];
>> z1=tf(n1,m1)
 Transfer function:
    10
-------------
s^2 + 2 s + 5
>> n2=[2 12 15];
>> m2=[1 3 7 5];
>> z2=tf(n2,m2)
Transfer function:
  2 s^2 + 12 s + 15
---------------------
s^3 + 3 s^2 + 7 s + 5
 >> G=z1+z2
 Transfer function:
  2 s^4 + 26 s^3 + 79 s^2 + 160 s + 125
-----------------------------------------
s^5 + 5 s^4 + 18 s^3 + 34 s^2 + 45 s + 25
```

Аналогично осуществляются операции вычитания, умножения и деления передаточных функций с помощью операторов -, *, /.

Функция *series()* используется для образования передаточной функции системы, состоящей из последовательного соединения звеньев. Она имеет вид: *series(q1,q2),*

где *q1* и *q2* – передаточные функции последовательно соединенных звеньев.

Пример 4.

Структурная схема системы управления показана на рис.1

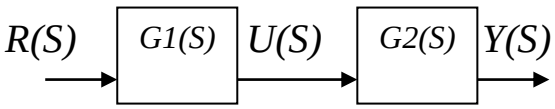


Рис.10. Структурная схема системы

Необходимо получить передаточную функцию системы

$$G(S) = \frac{Y(S)}{R(S)},$$

если передаточные функции звеньев имеют вид:

$$G1(S) = \frac{U(S)}{R(S)} = \frac{S+1}{S+2}, G2(S) = \frac{Y(S)}{U(S)} = \frac{1}{5S^2}$$

Решение:

```
>> n1=[1 1];
m1=[1 2];
q1=tf(n1,m1);
>> n2=[1];
m2=[5 0 0];
q2=tf(n2,m2);
>> G=series(q1,q2)
 Transfer function:
   s + 1
--------------
5 s^3 + 10 s^2
```

Функция *parallel()* используется для образования передаточной функции системы, состоящей из параллельных звеньев, и имеет вид:
parallel(q1,q2)
где *q1* и *q2* – передаточные функции параллельно соединенных звеньев.

Пример 5.

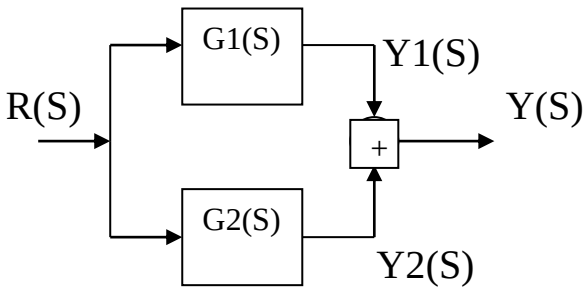


Рис.11. Структурная схема системы, состоящей из параллельных звеньев

Необходимо получить передаточную функцию системы

$$G(S) = \frac{Y(S)}{R(S)}$$

если передаточные функции звеньев имеет вид:

$$G1(S) = \frac{Y1(S)}{R(S)} = \frac{S+1}{S^2+3S+1}, G1(S) = \frac{Y2(S)}{R(S)} = \frac{S+2}{S^2+S+3}$$

Решение:

```
>> n1=[1 1];
m1=[1 3 1];
q1=tf(n1,m1);
n2=[1 2];
m2=[1 1 3];
q2=tf(n2,m2);
G=parallel(q1,q2)

Transfer function:
  2 s^3 + 7 s^2 + 11 s + 5
-----------------------------
s^4 + 4 s^3 + 7 s^2 + 10 s + 3
```

3.3. Функция *feedback()*

Функция *feedback()* применяется для образования передаточной функции замкнутой системы по известным передаточным функциям разомкнутой системы и цепи обратной связи. Она имеет вид:
feedback(q,qoc,±1), где: *qoc* - передаточная функция цепи обратной связи; ±1 – указывает вид обратной связи (-1-положительная, +1-отрицательная).

Пример 6.

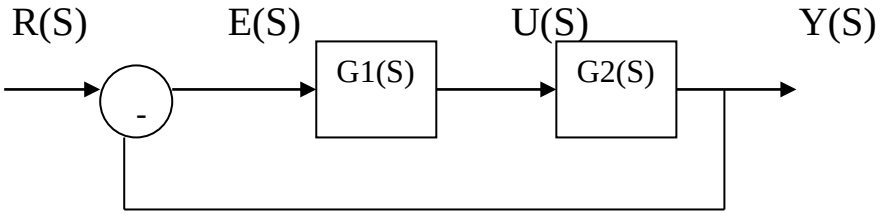


Рис.12. Структурная схема системы управления

Передаточная функция звеньев имеет вид:

$$G1(S) = \frac{S+1}{S+2}, G2(S) = \frac{1}{5S^2}$$

Передаточная функция цепи обратной связи образует связь с коэффициентом передачи, равным 1. Необходимо получить передаточную функцию замкнутой системы управления:

$$G(S) = \frac{Y(S)}{R(S)}$$

Передаточная функция G(S) определяется по выражению

$$G(S) = \frac{G1(S) \cdot G2(S)}{1 + G1(S) \cdot G2(S)}.$$

Из этого выражения и структурной схемы видно, что для получения передаточной функции замкнутой системы необходимо в начале образовать с помощью функции *tf*() звенья *G1(S)* и *G2(S)*, затем посредством функции *series()* образовать передаточную функцию разомкну-

той системы и после этих процедур использовать функцию *feedback()* для образования передаточной функции замкнутой системы.

Программа образования передаточной функции замкнутой системы управления имеет вид:

```
>> n1=[1 1];
m1=[1 2];
q1=tf(n1,m1)
 Transfer function:     s + 1
                        -----
                        s + 2
 >> n2=[1];
m2=[5 0 0];
q2=tf(n2,m2)
 Transfer function:    1
                       -----
                      5 s^2
 >> q=series(q1,q2)
Transfer function:     s + 1
                     --------------
                   5 s^3 + 10 s^2

>> feedback(q,[1])
 Transfer function:         s + 1
                      ----------------------
                   5 s^3 + 10 s^2 + s + 1
```

Пример 7.

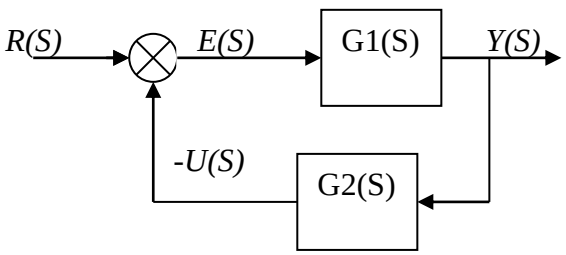


Рис.13. Структурная схема системы с гибкой отрицательной обратной связью

Передаточные функции имеют вид:

$$G1(S) = \frac{S+1}{S+2}, G2(S) = S + 0.5$$

Решение:

```
>> n1=[1 1];
m1=[1 2];
q1=tf(n1,m1);
>> n2=[1 0.5];
m2=[1];
q2=tf(n2,m2);
>> feedback(q1,q2,-1)

Transfer function:
     s + 1
-----------------
s^2 + 2.5 s + 2.5
```

3.4. Исследование переходных процессов в системах управления

Исследовать переходные процессы в системах управления можно следующими методами:

- непосредственным решением дифференциальных уравнений, описывающих динамику системы управления;
- с помощью преобразования Лапласа передаточной функции системы;
- с помощью встроенной функции *step()*.

Все эти методы могут быть реализованы в системе *MATLAB*. Для исследования переходных процессов с помощью преобразования Лапласа необходимо получить обратное преобразование Лапласа передаточной функции звена *Y(S)* и представить его графически, а затем по

виду графика определить вид переходного процесса (апериодический, колебательный) и его длительность. Для графического воспроизведения результата в *MATLAB* используется функция *ezplot()*, имеющая вид:

$$ezplot(Y(t), x_n, x_k),$$

где $Y(t)$ – функция, записанная в символьном виде (взята в кавычки); x_n, x_k – диапазон изменения аргумента, в нашем случае диапазон изменения t.

Пример 8.

Пусть обратное преобразование Лапласа имеет вид:

$$Y(t)=0.5e^{-0.5t}$$

Тогда программа воспроизведения графика в *MATLAB* будет иметь вид:

```
>> Y='0.5*exp(0.5*t)';
>> ezplot(Y,0,3)
```

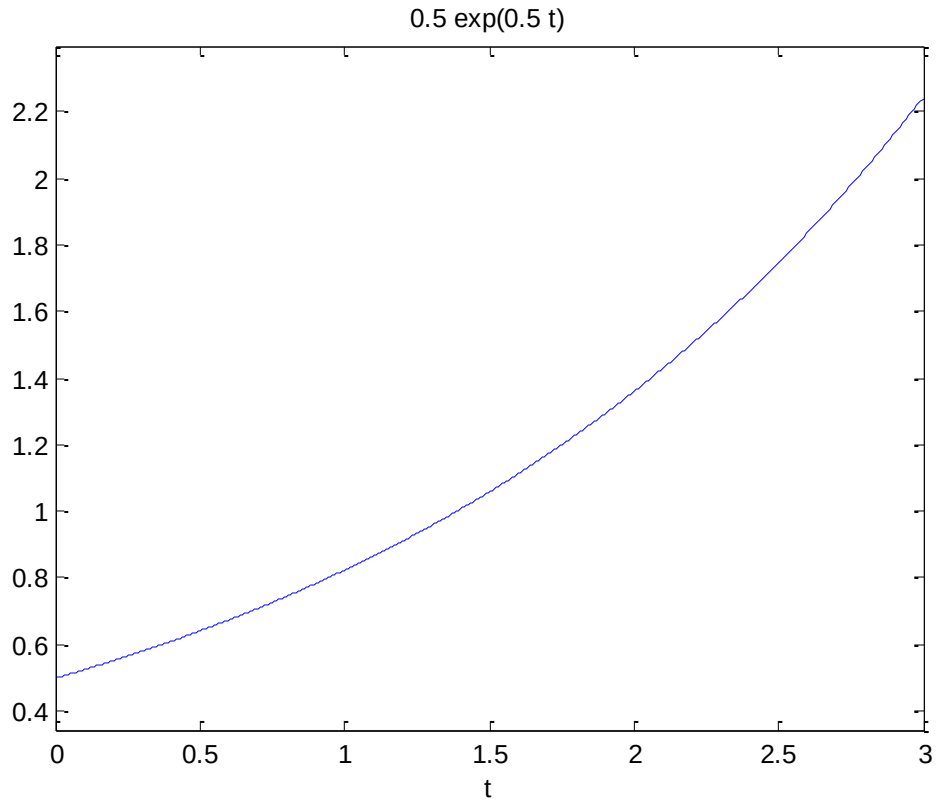


Рис.14. График переходного процесса системы

Функция *step()* вычисляет реакцию системы управления на единичное ступенчатое воздействие. Если целью исследования является

получение графика, то функция записывается в следующем виде *step(q, t)*,

где *q* – передаточная функция системы; *t* – время функционирования системы управления.

При этом график будет получен автоматически с указанием переменных по осям. Если же график необходим для иных целей с его сохранением, то функция записывается с указанием аргументов левой части, например, [*y*,*t*]=*step(q,t)*.

После этого для образования графика применяется функция *plot(t,y)*. При этом перед функцией *step(q,t)* необходимо указать диапазон изменения *t*, например, в таком виде: t=[0:0.1:3], т.е. от 0 до 3 с шагом 0,1.

Пример 9. Определить переходную характеристику системы управления, передаточная функция которой имеет вид:

$$Y(S) = \frac{5400}{2S^2 + 2.5S + 5402}$$

Решение:

```
>> n1=[5400];
>> m1=[2 2.5 5402];
>> q=tf(n1,m1)
>> t=[0:0.005:3];
>> [y,t]=step(q,t);
>> plot(t,y) ;grid on
```

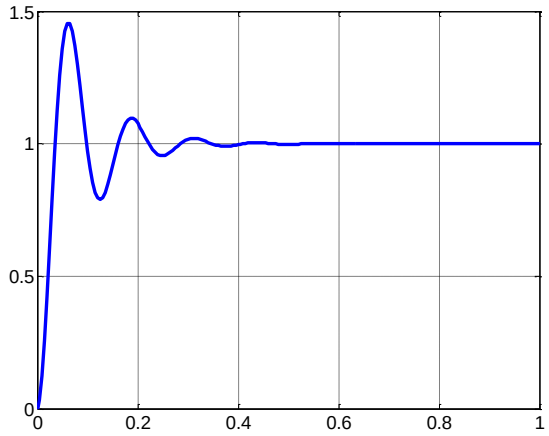


Рис.15. Переходная характеристика системы

3.5. Частотные характеристики системы

Амплитудно-частотные и фазо-частотные характеристики в системе *MATLAB* строятся с помощью функции *bode(sys)*, где *sys* – имя передаточной функции. Полученные частотные характеристики называются функциями Боде.

Пример 10. Необходимо построить частотные характеристики звена, передаточная функция имеет вид:

$$Y(S) = \frac{0.5S + 1}{S(2S + 1)}.$$

Решение:

```
>> n=[0.5 1];
>> m=[2 1 0];
>> sys=tf(n,m);
>> bode(sys)
```

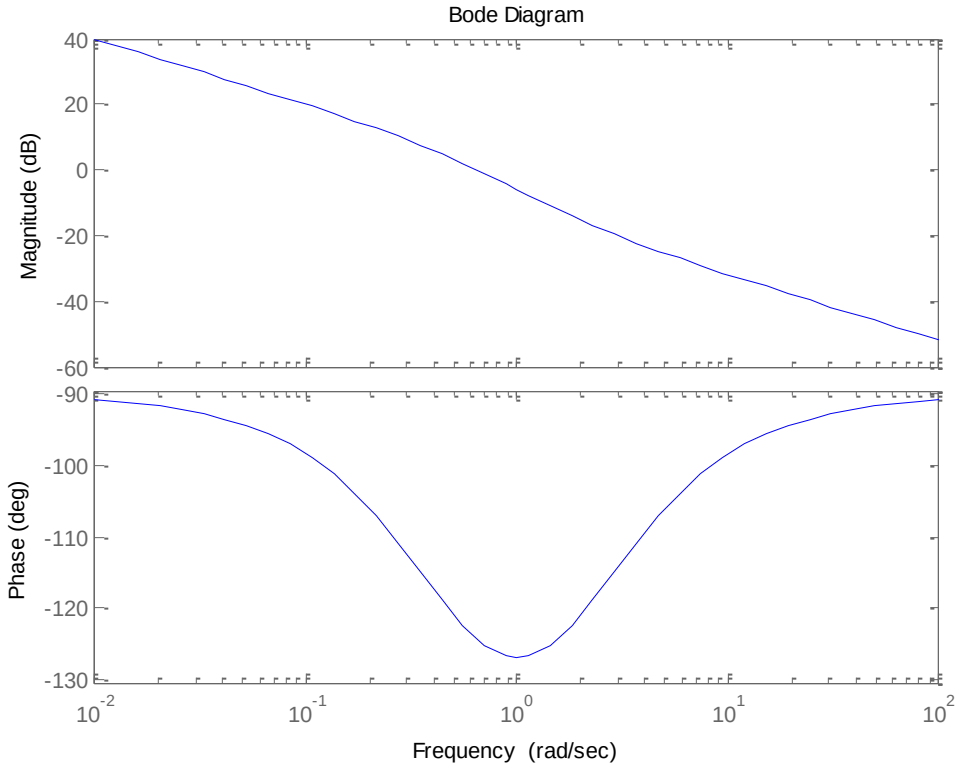


Рис.16. Частотные характеристики звена

При построении диаграммы Боде в области желаемых частот используется функция *logspace(a,b,n)*, где: *a* – начальное значение частоты; *b* – конечное значение частоты; *n* – число точек в диапазоне [*a;b*]. Функция *bode()* при этом записывается в следующем виде: *bode(sys,W)*.

Программа имеет вид:

```
>> N=[0.5 1];
>> M=[2 1 0];
>> sys=tf(N,M);
>> W=logspace(-1,3,200);
>> bode(sys,W)
```

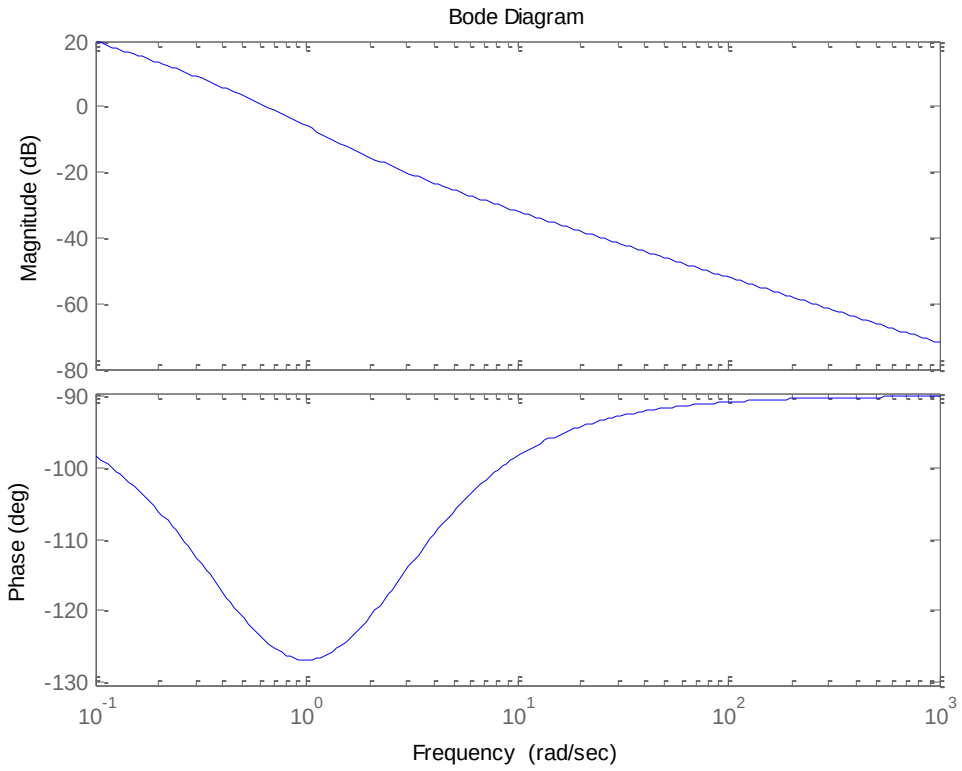


Рис.17. График функции Боде

3.6. Амплитудно-фазовая характеристика системы

Амплитудно-фазовой характеристикой называют диаграмму Найквиста. Она применяется для анализа устойчивости по критерию Найквиста. Реализуется в системе *MATLAB* с помощью функции *nyquist(sys)*, где *sys* – имя передаточной функции.

Пример 11. Необходимо построить диаграмму Найквиста звена, передаточная функция которой имеет вид:

$$Y(S) = \frac{0.5}{S^3 + 2S^2 + S + 0.5}$$

```
>> N=[0.5];
M=[1 2 1 0.5];
sys=tf(N,M);
nyquist(sys)
```

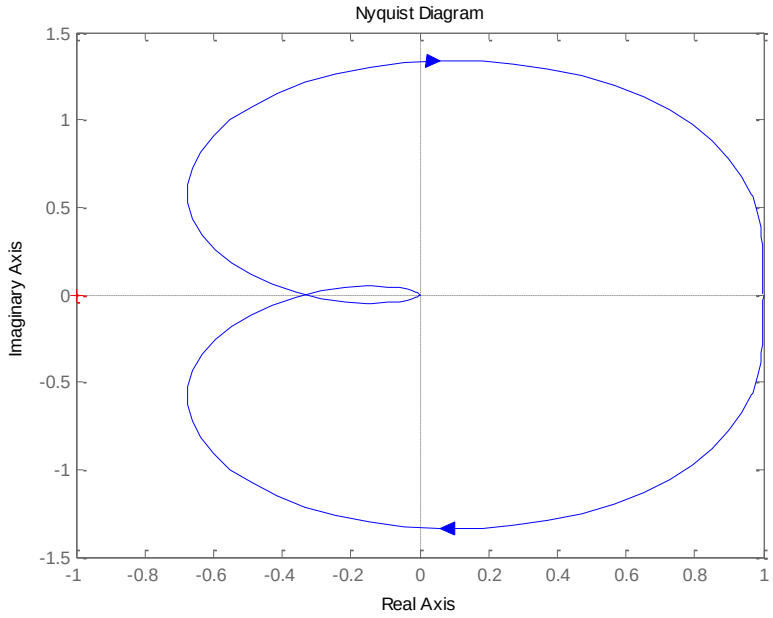


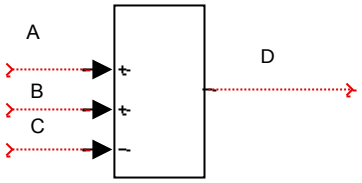
Рис.18. Диаграмма Найквиста построена при изменении частоты от 0 до ∞ (нижняя часть диаграммы) и от -∞ до 0 (верхняя часть диаграммы)

4. ВИЗУАЛЬНОЕ МОДЕЛИРОВАНИЕ СРЕДСТВАМИ *SIMULINK* В СИСТЕМЕ *MATLAB*

SIMULINK, сопутствующая *MATLAB* программа, - это интерактивная система для визуального моделирования. Она представляет собой среду, управляемую мышью, которая позволяет моделировать процесс путем перетаскивания блоков диаграмм из библиотеки *SIMULINK* на чистую страницу редактора. Блоки выстраиваются в соответствии с логикой решаемой задачи. Каждый блок, входящий в библиотеку *SIMULINK* , выполняет определенную математическую операцию: умножение, деление, сложение, вычитание, дифференцирование, интегрирование и т.д. Ниже приведены некоторые часто используемые блоки при решении задач химической технологии.

Для установки параметров в каждом блоке необходимо дважды щелкнуть мышью внутри блока, при этом появится окно установки параметров.

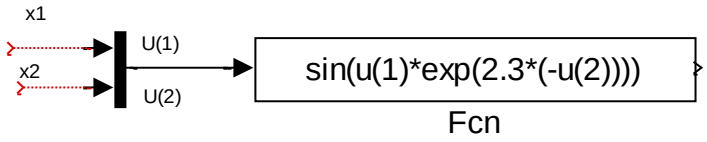
Например, установить алгебраическую сумму трех слагаемых $D=A+B-C$. В появившемся окне блока установить «++-» , в результате получится блок с установленными параметрами:



Требуется рассчитать значение функции двух переменных $x1$ и $x2$ в соответствии с уравнением:

$$f = \sin(x1\exp(2{,}3(-x2)))$$

Реализация решения этой задачи приведена ниже. При этом следует отметить, что блок *Fcn*, в котором может быть записано любое арифметическое выражение, в качестве входных переменных использует переменные *U(j)*, преобразованные блоком *Mux*. Индекс переменной *U(j)* определяется номером входа (сверху вниз) переменной *x(j)* в блок *Mux*.



Пример 1. Решить обыкновенное дифференциальное уравнение второго порядка вида:

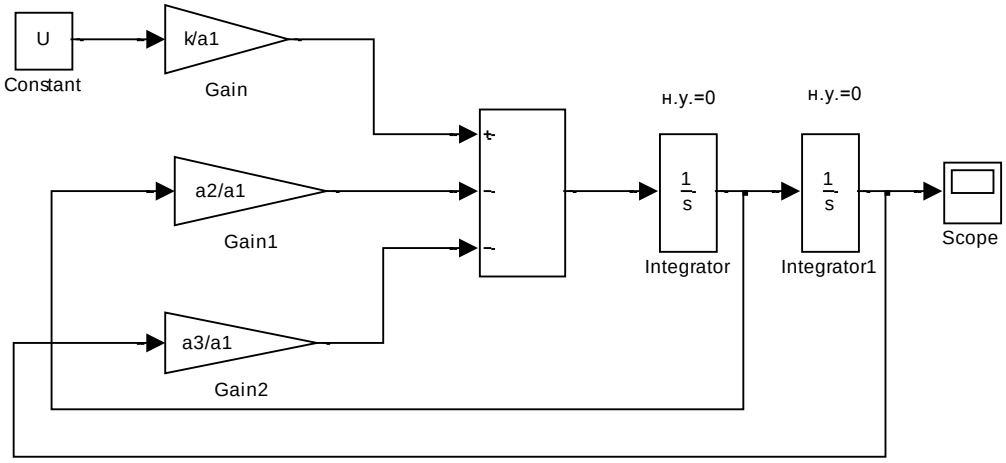
$$a1\frac{d^2F}{d\tau^2} + a2\frac{dF}{d\tau} + a3F = kU(\tau)$$

Задано $a1=3$, $a2=8$, $a3=1$, $к=1$, $U = 20$, н.у.: $\frac{dF}{d\tau} = F = 0$.

Для решения данного уравнения необходимо разрешить исходное уравнение относительно старшей производной:

$$\frac{d^2F}{d\tau^2} = -\frac{a2}{a1}\frac{dF}{d\tau} - \frac{a3}{a1}F + \frac{k}{a1}U(\tau)$$

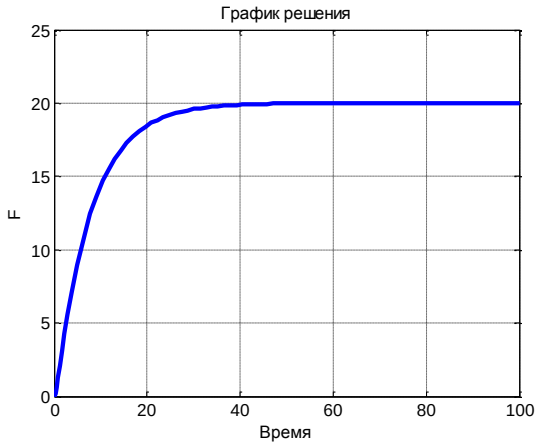
1. Для составления *Simulink*-программы нужны блоки: три блока умножения на постоянное число, блок суммирования, блок задания постоянного числа, два блока интегрирования, блок изображения решения. *Simulink*-программа имеет вид:



2. В рабочей области *MATLAB* задать:

U=20; *a*1=3; *a*2=8; *a*3=1; *k*=1, нажать на «*Enter*».

3. В верхнем меню *Simulink*-программы нажать кнопку пуска в виде черного треугольника. Результат решения в виде графика будет показан в блоке «*Scope*» и имеет вид:



Пример 2. Решить систему линейных алгебраических уравнений вида:

$$3x1 + 5x2 = 4$$
$$2x1 + 1{,}5x2 = 10$$

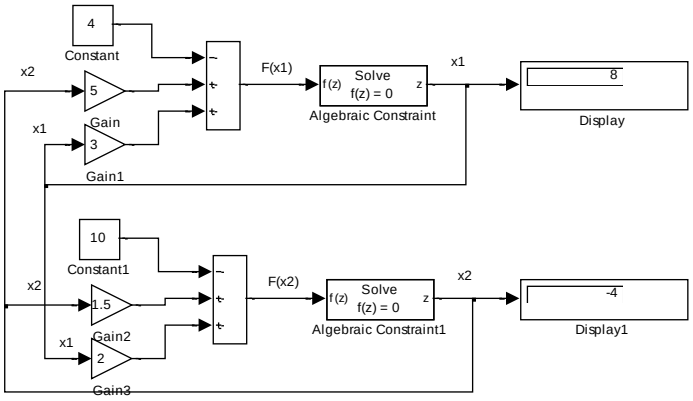
с использованием блока *solve*.

Исходная система уравнений записывается следующим образом:

$$F(x1) = 3x1 + 5x2 - 4$$
$$F(x2) = 2x1 + 1{,}5x2 - 10.$$

Для решения преобразованной системы из библиотеки *Simulink* необходимо перетащить следующие блоки: два блока задания постоянного числа, четыре блока умножения на постоянное число, два блока суммирования, два блока *solve*, два блока цифровых вольтметров.

Ниже приведена *Simulink*-программа:



После нажатия кнопки «пуск» найденные значения корней показывают цифровые вольтметры.

Пример 3. Решить систему нелинейных алгебраических уравнений вида:

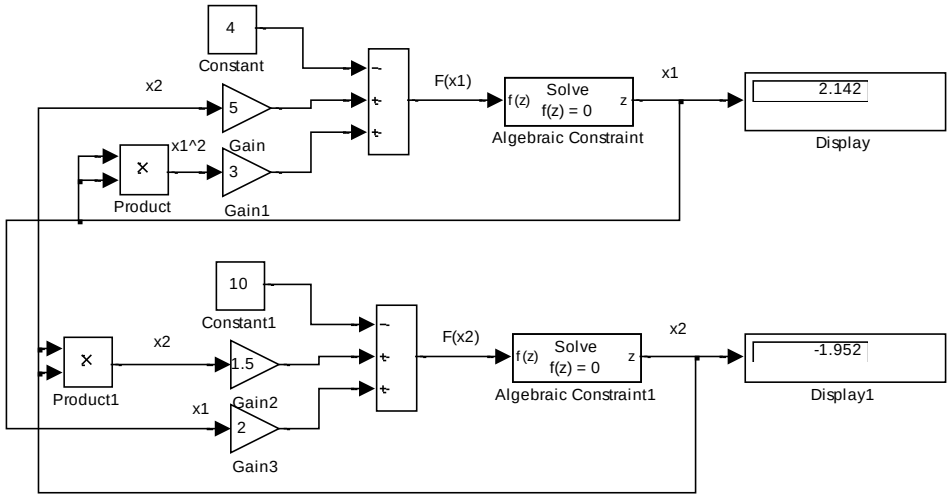
$$3x1^2 + 5x2 = 4$$
$$2x1 + 1{,}5x2^2 = 10$$

с использованием блока *solve*.

Исходная система уравнений записывается следующим образом:

$$F(x1) = 3x1^2 + 5x2 - 4$$
$$F(x2) = 2x1 + 1{,}5x2^2 - 10$$

Дополнительно к блокам примера 2 вводятся два блока умножения. Ниже приведена *Simulink*-программа:



5. ПРИМЕРЫ РЕШЕНИЯ ЗАДАЧ ХИМИЧЕСКОЙ ТЕХНОЛОГИИ В СИСТЕМЕ *MATLAB*

5.1. Аналитический способ определения оптимальных режимных параметров процесса крбидизации пенокарбидов титана

Для решения задачи оптимизации необходимо найти такие значения входных переменных из заданных областей их изменения, при которых критерий оптимальности принимает экстремальное значение с определенной точностью. При решении этой задачи в качестве критерия оптимальности используется функция в виде абсолютной разности между заданным значением выходной переменной f_z и рассчитанным значением f_r по заданному математическому описанию:

$$f_{opt}(x_1, x_2, x_3) = \left| f_z - f_r \right|,$$

где x_1, x_2, x_3 - безразмерные значения входных переменных .

В данном примере рассматривается задача поиска оптимальной скорости нагрева изделия ($x1$), находящегося в печи карбидизации, при которой максимально допустимый температурный градиент составляет 800 град/м при заданных геометрических размерах изделия: толщина стенки конуса 0,12 м, высота – 0,7 м, что соответствует в безразмерных координатах нулевому уровню ($x2 = 0$, $x3 = 0$). Зависимость градиента температур f_r от $x1$, $x2$, $x3$ имеет вид:

$$f_r = 765.18 + 193.48 \cdot x_1 + 256.70 \cdot x_2 + 154.93 \cdot x_3 + 56.46 \cdot x_1 \cdot x_2 + 30.36 \cdot x_1 \cdot x_3 + 85.06 \cdot x_2 \cdot x_3 + 36.97 \cdot x_1^2 + 33.74 \cdot x_2^2 - 159.83 \cdot x_3^2.$$

Для нахождения минимума критерия оптимальности f_{opt} используется детерминированный генетический алгоритм поиска глобального экстремума, который носит название «поиск по образцу». Он осуществляется с помощью *Matlab* – команды «*patternsearch*». Формат обращения к этой команде имеет следующий вид:

```
[km fopt1] = patternsearch(@myGRM,x0),
```

где *km* – вектор значений входных переменных, при которых критерий оптимальности *fopt*1 достигает минимального значения, *x*0 – вектор координат начальной точки поиска.

Под образцом понимается набор векторов, используемых алгоритмом для поиска наилучшей точки на каждой итерации.

На каждом шаге алгоритма поиска по образцу исследуется набор точек, называемых сеткой, для поиска точки, в которой значение целевой функции меньше по сравнению с ранее найденным значением. На каждом шаге алгоритма далее для всех точек сетки производится вычисление целевой функции. Если опция алгоритма *Complete poll* (Полный опрос) выключена (*off*), а это установка по умолчанию, то опрос указанных точек производится до тех пор, пока в какой-либо точке значение целевой функции не станет меньше, чем в текущей базовой точке. Если подобная точка находится, то опрос называется успешным и данная точка сетки принимается за новую базовую.

Если такая точка не находится, то опрос считается неудачным и базовая точка остается таковой и на следующей итерации алгоритма.

Алгоритм прямого поиска по образцу может быть описан следующим образом:

1. Задается некоторая начальная базовая точка, векторы образца и начальный шаг сетки, равный единице.
2. Определяются точки сетки.
3. Рассчитывается значение целевой функции в базовой точке.
4. Производится опрос точек сетки. В случае неудачного опроса переходим к пункту 6.
5. При успешном опросе определяется новая базовая точка, и шаг сетки увеличивается в два раза. Переходим к пункту 7.
6. Шаг сетки уменьшается в два раза. Проверяется выполнение условий останова алгоритма. В случае их невыполнения — переход к пункту 2.
7. Окончание работы алгоритма, выдача результатов поиска (найденные оптимальные значения входных переменных и минималь-

ная погрешность найденной целевой функции, которая определяется минимальным размером сетки).

Алгоритм завершает свою работу при выполнении одного из следующих правил (критериев) останова:

1) размер сетки меньше значения параметра - допуск сетки;

2) число итераций, выполненных алгоритмом, достигло величины, задаваемой параметром - максимальное число итераций;

3) общее число вычислений значений целевой функции достигло величины, определяемой параметром - максимальное число вычислений целевой функции;

4) расстояние между двумя базовыми точками, определенными по результатам двух последовательных успешных опросов, меньше, чем в заданном параметре - *X* допуск;

5) изменения в значении целевой функции для соседних базовых точек, полученных в результате двух последних успешных опросов, меньше, чем в заданном параметре - допуск функции.

Зная исходные геометрические размеры полого цилиндра и максимально допустимое значение градиента температуры с помощью алгоритма поиска по образцу находим оптимальную скорость нагрева и минимальное отклонение найденного значения градиента температуры от заданного. Программа поиска оптимальной скорости нагрева на основе алгоритма поиска по образцу приведена ниже и включает в себя два блока:

1. *Script_ patternsearch* . В этом блоке задаются координаты начальной точки поиска, формируется обращение к *patternsearch*, выводятся на печать полученные расчетные данные.

2. *M*-файл функция формирования критерия оптимальности *f*.

```
%Script_ patternsearch
x0=0.1; % начальная точка поиска
[km fopt1] = patternsearch(@myGRM,x0);%обращение к алгоритму
AM=(km(1)*0.007+0.022)*3600;
x1=km;x2=0;x3=0;
```

```
b=[25.9438 -5.4913 1.4308 0.9511 0.0588 0.0262 0.5287 1.4565 0.2117 -
0.9040];
yr=b(1)+b(2)*x1+b(3)*x2+b(4)*x3+b(5)*x1.*x2+b(6)*x1.*x3+b(7)*x2.*x
3+b(8)*x1.^2+b(9)*x2.^2+b(10)*x3.^2;
disp([km AM yr]);
 function f = myGRM(x);
b=[765.1806 193.4781 256.6975 154.9285 56.4625 30.3625 85.0625
36.9740 33.7427 -159.8288];
x3=0;
x2=0;
yz=800;
f=abs(yz-(b(1)+b(2)*x(1)+b(3)*x2+b(4)*x3+b(5)*x(1).*x2…
+b(6)*x(1).*x3+b(7)*x2*x3+b(8)*x(1).^2+b(9)*x2^2+b(10)*x3^2));
    disp([x f]);
```

Результат решения:

```
Optimization terminated: current mesh size 9.5367e-007 is less than
'TolMesh':        0.17        83.59        25.03
fopt1 = 2.1992e-005
```

По полученным результатам можно сказать, что при заданных значениях геометрических размеров изделия и максимального градиента температуры можно определить оптимальную скорость нагрева. В отличие от графо-аналитического способа этот метод позволяет уйти от построения номограмм при каждом новом соотношении входных параметров.

5.2. Нахождения оптимального состава композиции при получении вяло-упругих пенополиуретанов

Основными показателями качества вяло-упругих пенополиуретанов являются: $y1$ – отскок; $y2$ – время восстановления; $y3$ – коэффициент упругости; $y4$ – коэффициент механических потерь. Зависимость перечисленных показателей качества от начального состава ($x1$ – изоцианатный компонент (индекс); $x2$ – высокомолекулярный по-

лиэфир, масс.ч.; *x*3 – короткоцепной полиэфир, масс.ч.; *x*4 – плотность пены, кг/м^3) дается следующими уравнениями:

y1=(b(1)+b(2).x(1)+b(3).x(2)+b(4).x(3)+b(5).x(4)+b(6).x(1).x(2)+
b(7).x(1).x(3)+b(8).x(1).x(4)+b(9).x(2).x(3)+b(10).x(2).x(4)+
b(11).x(3).x(4)));

y2=(b1(1)+b1(2).x(1)+b1(3).x(2)+b1(4).x(3)+b1(5).x(4)+
b1(6).x(1).x(2)+b1(7).x(1).x(3)+b1(8).x(1).x(4)+b1(9).x(2).x(3)+
b1(10).x(2).x(4)+b1(11).x(3).x(4)));

y3=(b2(1)+b2(2). x(1)+b2(3).x(2)+b2(4).x(3)+b2(5).x(4)+
b2(6).x(1).x(2)+b2(7).x(1).x(3)+b2(8).x(1).x(4)+b2(9).x(2).x(3)+
b2(10).x(2).x(4)+b2(11).x(3).x(4)));

y4=(b3(1)+b3(2).x(1)+b3(3).x(2)+b3(4).x(3)+b3(5).x(4)+
b3(6).x(1).x(2)+b3(7).x(1).x(3)+b3(8).x(1).x(4)+b3(9).x(2).x(3)+
b3(10).x(2).x(4)+b3(11).*x(3).x(4))).

Коэффициенты в уравнениях регрессии приведены ниже:

b	*b*1	*b*2	*b*3
279.81	1568.00	-2213.00	15.50
0	0	0	0
309.59	3636.00	-5060.00	24.76
822.44	4682.00	-6788.00	45.10
-306.99	-16650.00	22519.00	-78.97
-3.78	16.00	-15.00	0.08
0	0	0	0
0	0	0	0
935.91	10932.00	-15236.00	74.73
0	0	0	0
-920.92	-49963.00	67561.00	-236.94

Для нахождения состава, обеспечивающего заданные значения показателей качества $y1_z$, $y2_z$, $y3_z$, $y4_z$, оптимизируемая функция будет иметь вид: *fopt = f1+f2+f3+f4,*

где *f1 = abs(y1-y1$_z$); f2 = abs(y2-y2$_z$); f3 = abs(y3-y3$_z$); f4 = abs(y4-y4$_z$).*

Найденный минимум функции *fopt* будет соответствовать составу композиции, при котором будут обеспечиваться заданные значения показателей качества.

В данном примере в качестве заданных значений показателей качества взяты следующие: $y1_z = 7$, $y2_z = 28$, $y3_z = 60$, $y4_z = 0{,}6$. Программа решения поставленной задачи приведена ниже:

```
%Script_poisk4
x0=[0 0 0 0];%координаты начальной точки поиска
[xopt fval]=fminsearch(@svoistva,x0);
disp('Найденные значения входных переменных x(i)');
disp(xopt);
X(1)=10*xopt(1)+80;
X(2)=7.5*xopt(2)+37.5;
X(3)=7.5*xopt(3)+22.5;
X(4)=16*xopt(4)+72;
disp('Найденные значения входных переменных x(i)в размерных величинах');
disp([X(1) X(2) X(3) X(4)]);
disp('Значение оптимизируемой функции');
disp(fval);
 function f=svoistva(x)
%отскок
b=[279.8063;0;309.5892;822.4432;-306.9893;-3.7820;0;0;935.9063;0;-920.9204];
%Время восстановления
b1=[1568;0;3636;4682;-16650;16;0;0;10932;0;-49963];
%Коэффициент упругости
b2=[-2213;0;-5060;-6788;22519;-15;0;0;-15236;0;67561];
%Коэффициент механических потерь
b3=[15.4979;0;24.7641;45.1033;-78.9680;0.0770;0;0;74.7319;0;-236.9402];
```

```
% Формирование ограничений на входные переменные x(i), i=1,4
for i=1:4;
   if x(1)>1;
      x(1)=1;end;
   if x(1)<-1;
      x(1)=-1;end;
   if x(2)>1;
      x(2)=1; end;
    if x(2)<-1;
      x(2)=-1; end;
    if x(3)>1;
      x(3)=1;  end;
    if x(3)<-1;
      x(3)=-1; end;
 if x(4)>1;
      x(4)=1; end;
   if x(4)<-1;
       x(4)=-1;end;
end;
f1=abs(7-(b(1)+b(2).*x(1)+b(3).*x(2)+b(4).*x(3)+b(5).*x(4)+
b(6).*x(1).*x(2)+b(7).*x(1).*x(3)+b(8).*x(1).*x(4)+b(9).*x(2).*x(3)+
b(10).*x(2).*x(4)+b(11).*x(3).*x(4)));
f2=abs(28-(b1(1)+b1(2).*x(1)+b1(3).*x(2)+b1(4).*x(3)+b1(5).*x(4)+
b1(6).*x(1).*x(2)+b1(7).*x(1).*x(3)+b1(8).*x(1).*x(4)+b1(9).*x(2).*x(3)
+b1(10).*x(2).*x(4)+b1(11).*x(3).*x(4)));
f3=abs(60-(b2(1)+b2(2).*x(1)+b2(3).*x(2)+b2(4).*x(3)+b2(5).*x(4)+
b2(6).*x(1).*x(2)+b2(7).*x(1).*x(3)+b2(8).*x(1).*x(4)+b2(9).*x(2).*x(3)
+b2(10).*x(2).*x(4)+b2(11).*x(3).*x(4)));
f4=abs(0.6-(b3(1)+b3(2).*x(1)+b3(3).*x(2)+b3(4).*x(3)+b3(5).*x(4)+
b3(6).*x(1).*x(2)+b3(7).*x(1).*x(3)+b3(8).*x(1).*x(4)+b3(9).*x(2).*x(3)
+b3(10).*x(2).*x(4)+b3(11).*x(3).*x(4)));
f=f1+f2+f3+f4;
disp([x,f]);
```

Результат работы программы:

Найденные значения входных переменных *x(i)*

0.33 -1.00 -0.04 -0.12

Найденные значения входных переменных *x(i)*в размерных величинах

83.34 30.00 22.21 70.02

Значение оптимизируемой функции

39.48

5.3. Обратные задачи химической технологии

Под обратными задачами понимают нахождение неизвестных параметров, входящих в математическое описание, по экспериментальным данным.

Постановка задачи: из заданной области изменения входных переменных найти такие их значения, при которых оптимизируемая функция *fopt* принимает минимальное значение с заданной степенью точности. Общий вид этой функции имеет вид:

$$fopt(x(i)) = \sum_{j=1}^{n} \left| (ye(j) - yr(j)) \right|,$$

где i – число искомых переменных, j – число экспериментальных точек, *ye(j)* и *yr(j)* – экспериментальные и расчетные значения выходной переменной.

Для нахождения минимума функции одной переменной может быть использована функция *MATLAB fminbnd()*. Для нахождения минимума функций многих переменных используется функция *MATLAB fminsearch()*.

Пример 1. Рассчитать константу скорости k химической реакции вида: $A \xrightarrow{k} B$ по экспериментальным данным изменения концентрации вещества A во время реакции.

Исходные данные:

Время, мин	0	1	2	3	4	5
Конц. комп. А, моль/л	1	0.65	0.4	0.29	0.14	0.118

Кинетика данной реакции описывается следующим дифференциальным уравнением:

$$\frac{dCa}{d\tau} = -kCa$$

Программа для решения данной задачи включает в себя три блока:

- *Script_mayfun1* – блок ввода входных данных, печать найденной константы скорости и графического сопоставления экспериментальных и расчетных данных;
- *mayfunf*1 – блок формирования оптимизируемой функции *fopt*;
- *fdr*1- блок решения дифференциального уравнения, описывающего кинетику реакции.

Текст *MATLAB* программы приведен ниже:

```
%Skript_mayfun1
ye=[1;0.65;0.4;0.29;0.14;0.118];
tspan=[0:1:5];y0=1;
k0=[0.18 10];
[km fopt1]=fminbnd('mayfunf1',k0,[],ye,tspan);
disp('Минимуи km=');disp(km);
disp('fopt1=');disp(fopt1);
[t,yr]=ode15s(@fdr1,tspan,y0,[],km);
disp('*****************************');
disp('Сравнение эксперим. и расчетн. данных');
disp('    ye      yr      ye-yr');
disp([ye yr ye-yr]);
disp('Сумма abs(ye-yr) = ');
disp(abs(sum(abs(ye-yr))));
plot(t,yr);grid on;
```

```
xlabel('Tau,c');ylabel('Изменение Ca');
hold on;
plot(t,ye,'*');
function f2=mayfunf1(k,ye,tspan);
%Поиск k
disp(k);
y0=1;
[t,yr]=ode15s(@fdr,tspan,y0,[],k);
dy=ye-yr;
f2=sum(abs(dy));
function df=fdr1(t,y,k);
df(1,1)=-k*y(1,1);
% end fdr
```

Результат решения: Минимуи *km*=0.43

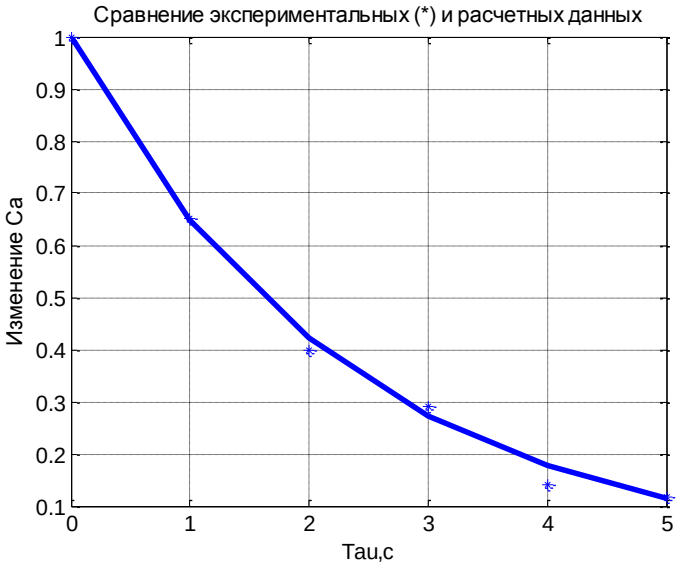
*fopt*1= 0.08

Сравнение эксперим. и расчетн. данных

ye	yr	ye-yr
1.00	1.00	0
0.65	0.65	0.00
0.40	0.42	-0.02
0.29	0.27	0.02
0.14	0.18	-0.04
0.12	0.12	0.00

Сумма *abs(ye-yr)* =

0.08



Пример 2. Нахождение констант последовательной химической реакции вида $A \xrightarrow{k1} B \xrightarrow{k2} C$ с использованием стохастического генетического алгоритма.

Кинетика данной реакции описывается системой дифференциальных уравнений вида:

$$\frac{dCa}{d\tau} = -k_1 Ca$$

$$\frac{dCb}{d\tau} = k_1 Ca - k_2 Cb$$

$$\frac{dCc}{d\tau} = k_2 Cb$$

Экспериментальные данные по изменению концентраций *Cb* в процессе реакции:

Время,с	0	1	2	3	4	5	6	7	8	9	10
Конц.*Cb*	0	0,36	0,44	0,41	0,33	0,26	0,19	0,14	0,10	0,07	0,04

Поиск глобального минимума оптимизируемой функции, имеющей вид: $fopt(k_1, k_2) = \sum_{j=1}^{n} \left| (ye(j) - yr(j)) \right|,$

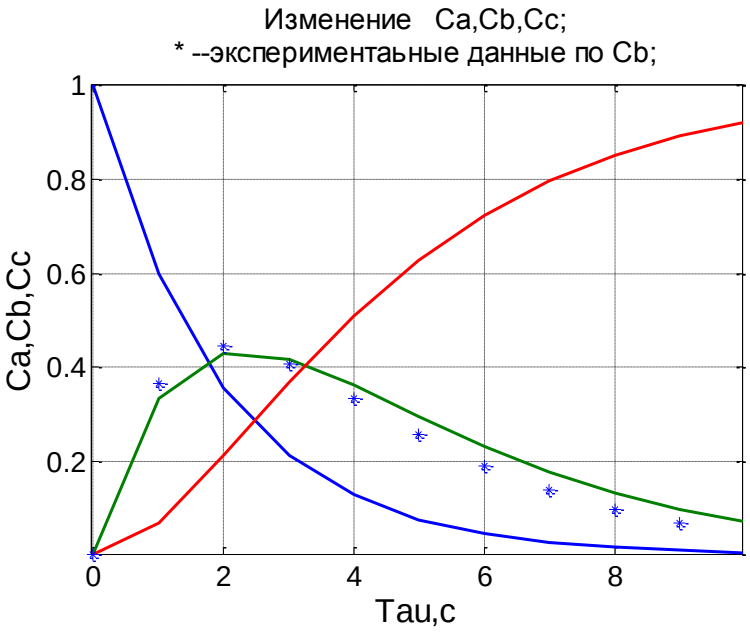
где *j* – число экспериментальных точек, *ye(j)* и *yr(j)* – экспериментальные и расчетные значения выходной переменной, осуществляется с помощью функции *MATLAB - ga()*

Формат обращения имеет вид: [*km,fval*] = *ga(@funf2,n),* где *km* – вектор найденных значений входных переменных, *fval* – оптимизируемая функция, *n* – число входных переменных, в нашем случае *n* = 2.

Текст программы :

```
%Skript_Gafunf2
% Нахождение Констант последовательной реакции A--->B-->C
% с использованием генетического алгоротма  ga
tspan=[0:1:10];
%Начальные условия y0  и экспериментальные данные ye
y0=[1;0;0];
```

```
ye=[0;0.3645;0.4444;0.4079;0.3338;0.2568;0.1903;0.1375;0.0976;0.0684;
0.04];
[km,fval] = ga(@funf2,2);
disp('Минимум km=');disp(km);
disp('fval=');disp(fval);
[t,yr]=ode15s(@fdr2,tspan,y0,[],km);
plot(t,yr);grid on;
xlabel('Tau,c');ylabel('Изменение Ca');
hold on;
plot(t,ye,'*');
srs=std(ye-yr(:,2));%Среднеквадратичная погрешность
disp('Среднеквадратичная погрешность=');disp(srs);
  function f2=funf2(k);
%Поиск k
ye=[0;0.3645;0.4444;0.4079;0.3338;0.2568;0.1903;0.1375;0.0976;0.0684;
0.4];
tspan=[0:1:10];y0=[1;0;0];
[t,yr]=ode15s(@fdr2,tspan,y0,[],k);
dy=ye-yr(:,2);
f2=sum(abs(dy));
disp([k f2])
function df=fdr2(t,y,k);
df=[-k(1)*y(1);k(1)*y(1)-k(2)*y(2);k(2)*y(2)];
% end fdr2
Минимум km=
        0.51         0.36
fval=
        0.12
Среднеквадратичная погрешность=
        0.02
```



5.4. Расчет оптимального режима термообработки при получении пенокарбидов титана

При карбидизации изделий в форме плоскопараллельной пластины толщина изделия *H* значительно меньше длины *A* и ширины *B*. В соответствии с этим можно считать, что нагрев образца, помещенного в печь карбидизации, идет со стороны больших поверхностей (симметричный двухсторонний нагрев).

Для расчета распределения температур по толщине изделия в этом случае можно использовать уравнение нестационарной теплопроводности вида:

$$\frac{\partial \dot{O}(h,\tau)}{\partial \tau} = a(ti,c,T)\frac{\partial^2 T(h,\tau)}{\partial h^2},$$

где *a(ti,c,T)* – температурная зависимость коэффициента температуропроводности; *h, τ* – текущие толщина и время.

Для решения данного уравнения необходимо задать:

- начальные условия $T(h,0) = T_{нач}$;
- граничные условия $T(0, \tau) = T(H, \tau) = temp^* \tau + T_{нач}$,

где *temp* – скорость изменения температуры в печи карбидизации; $T_{нач}$ – начальная температура.

Математическое описание собственно кинетики карбидизации изделия в различных сечениях образца будет описываться системой дифференциальных уравнений:

$$\frac{\partial y_1(h,\tau)}{\partial \tau} = -K(T) y_1^{n1(T)}(h,\tau) y_2^{n2(T)}(h,\tau)$$

$$\frac{\partial y_2(h,\tau)}{\partial \tau} = -K(T) y_1^{n1(T)}(h,\tau) y_2^{n2(T)}(h,\tau)$$

$$\frac{\partial y_3(h,\tau)}{\partial \tau} = n3 * K(T) y_1^{n1(T)}(h,\tau) y_2^{n2(T)}(h,\tau),$$

которая решается при заданных начальных условиях и где y_1 – число молей TiO_2; y_2 – число молей углерода; y_3 – число молей карбида титана.

Таким образом система представленных дифференциальных уравнений с заданными начальными и граничными условиями представляет собой математическое описание карбидизации изделий в виде плоской пластины.

Учитывая, что разогрев пластины представляет собой двухсторонний симметричный нагрев, данная задача может решаться на полутолщину изделия.

Конечностно-разностные уравнения математического описания процесса карбидизации будут иметь следующий вид:

$$\frac{dT_1}{d\tau} = \frac{a(ti,c,T_1)}{\Delta h^2}(T_p - 2T_1 + T_2)$$

$$\frac{dT_j}{d\tau} = \frac{a(ti,c,T_j)}{\Delta h^2}(T_{j-1} - 2T_j + T_{j+1}),\ j = 2, n-1$$

$$\frac{dT_n}{d\tau} = \frac{a(ti,c,T_n)}{\Delta h^2}(T_{n-1} + T_n),$$

где n – число элементарных слоев на полутолщине образца.

Система уравнений, описывающих кинетику процесса, при этом будет иметь вид:

$$\frac{dy_{1j}}{d\tau} = -K(T_j)y_{1j}^{n1(T)}y_{2j}^{n2(T)},$$

$$\frac{dy_{2j}}{d\tau} = -K(T_j)y_{1j}^{n1(T)}y_{2j}^{n2(T)}, \qquad j=1,n$$

$$\frac{dy_{3j}}{d\tau} = n3 * K(T_j)y_{1j}^{n1(T)}y_{2j}^{n2(T)},$$

Уравнения решались с начальными условиями:

$Tj(0) = 293\ K,\ j=1,n;$

$y_{1j}(0) = 2{,}85$ моль, $j=1,n;$

$y_{2j}(0) = 0{,}96$ моль, $j=1,n;$

$y_{3j}(0) = 0$ моль, $j=1,n.$

Температура в печи карбидизации изменялась по линейному закону: $T_p(\tau) = temp \cdot \tau + T_{нач}$ до температуры 1973 *К* и далее оставалась постоянной в течение всего времени выдержки.

К режимным параметрам процесса карбидизации изделий из пенокарбида титана можно отнести скорость нагрева изделия, начальный состав композиции, температуру карбидизации, полное время карбидизации и время выдержки изделия при постоянной температуре.

Задача оптимизации технологического режима процесса карбидизации может быть сформулирована следующим образом: для изделия в форме плоской пластины при различных толщинах найти такую скорость подъема температуры в печи карбидизации, при которой возникающий в изделии градиент температур не превшал бы предельно допустимого значения с заданной степенью точности при выполнении следующих условий ограничения: конечная температура в печи нагрева составляет 1973 *К*. При найденной скорости нагрева необходимо рассчитать время выдержки при конечной температуре и полное время процесса карбидизации.

Для решения поставленной задачи разработана *MATLAB*-программа.

Она включает в себя три *m*-файла: *Script*-файл (управляющая

часть программы), *mayskor_tic.m* (файл формирования критерия оптимальности) и *fdrskor.m* (файл формирования правых частей уравнений). *Script*-файл предназначен:

1.Для ввода входных данных: толщина пластины, содержание титана и углеродных микросфер, число слоев образца, начальная и конечная температуры карбидизации, заданный температурный градиент, начальные условия.

2.Для расчета максимального градиента температур при любом значении скорости нагрева.

3. Для расчета максимального градиента температур, полного времени карбидизации и времени выдержки образца при оптимальной скорости нагрева и для вывода расчетных данных.

Далее приводится программа поиска оптимальной скорости нагрева пластины:

```
%Skript_mayfunf1
h=input('Vvedite tolshiny plastini, m.');
h=h;%толщина пластины, м
n=10;%число слоев
ti=input('Vvedite soderganie titana, mass.');
c=input('Vvedite soderganie ugleroda, mass.');
x1=ti/48;%содержание титана
x2=c/12;%содержание углерода
b=[ 0.0209   -0.0249   -0.0030    0.0015    0.0076    0.0007];
alrr=b(1)+b(2)*x1+b(3)*x2+b(4)*x1.*x2+b(5)*x1.^2+b(6)*x2.^2;
a=alrr;%отношение теплопроводности к плотности
tnach=293;%начальная температура карбидизации
tkon=2000;%конечная температура карбидизации
tspan=[0:900:50*3600];
y0=[293;293;293;293;293;...% температура по слоям
   2.85;2.85;2.85;2.85;2.85;...% TiO2 по слоям
   0.96;0.96;0.96;0.96;0.96;...% C по слоям
   0;0;0;0;0;];% TiC по слоям
```

```
grtz=800;%заданный гардиент температур
[vm,fopt1]=fminbnd('mayskor_tic',50/3600,350/3600,[],tspan,a,
tnach,tkon,h,n,grtz);
%disp('Скорость нагрева');disp(vm);%скрость нагрева
%disp('fopt1=');disp(fopt1);
[t,yr]=ode15s(@fdrskor,tspan,y0,[],h,n,vm,tnach,tkon,a);
tsp=[0:900:50*3600];
[t1,yr]=ode15s(@fdrskor,tspan,y0,[],h,n,vm,tnach,tkon,a);
yrr=yr;
plot(t1/3600,yrr);
grid on;
xlabel('time, hours');
ylabel('gradient, degree/m');
hold on;
grtt=((yrr(:,1)-yrr(:,n/2))./(h/2));
subplot(2,1,1);
plot(t1/3600,grtt,'LineWidth',3);
grt=(yr(:,1)-yr(:,n/2))./(h/2);maxgrt=max(grt);
grtic=(yr(:,16)-yr(:,20))/(h/2);maxgrtic=max(grtic);
grid on;
xlabel('time,h');
ylabel('temperature gradient, degree/m');
subplot(2,1,2);
plot(t1/3600,grtic,'LineWidth',3);
grid on;
xlabel('time,h');
ylabel('TiC gradient, mole/m');
tn=((tkon-tnach)/vm)/3600;%время нагрева, ч
rr=vm*3600;
disp('Скорость нагрева, г/ч ');
disp(rr);
rq= maxgrt;
```

```
disp('Максимальный гардиент, г/м');
disp(rq);
rw=grtz;
disp('Заданный градиент, г/м');
disp(rw);
gr0=0;
grtict=((yr(:,16)-yr(:,20))/(h/2));
for i=1:200
   grt(i)=grtt(i);
   k=i;
   if grt(k)>gr0
      gr0=grt(k);
      kk=k; else   grtm=gr0;   end
end;
disp('Время достижения максимального температурного градиента,ч')
 disp([t1(kk)/3600]);
 grm0=grtm;
 gr0=0;
for y=1:200
   grt(y)=grtic(y);
   q=y;
   if grt(q)>gr0
      gr0=grt(q);
      qq=q;
         else
      grtm=gr0;
   end
end;
disp('Время достижения максимального градиента по пенокарбиду ти-
тана,ч')
 disp([t1(qq)/3600]);
 grm0=grtm;
```

```
for g=qq:200
gr(g)=grtic(g);
   mm=g;
   if gr(mm)<grtm
      grm0=gr(mm);
      kj=mm;
      %disp([grm0,kj]);
       end
   if grm0<0.05
      break
   end
   end;
tv=t1(mm)/3600-tn;%Время выдержки, ч
disp('Полное время карбидизации, м');
disp(t1(mm)/3600);
disp('Время нагрева, ч    Время выдержки, ч')
disp([tn,                tv]);
hold on; plot(t1(mm)/3600,grtic(mm),'o',...
              'MarkerEdgeColor','r',...
              'MarkerFaceColor','r',...
              'MarkerSize',7);
function f2=mayskor(v,tspan,a, tnach,tkon,h,n,grtz);
%Поиск скорости нагрева
%disp(v);
y0=[293;293;293;293;293;...% температура по слоям
   2.85;2.85;2.85;2.85;2.85;...% TiO2 по слоям
   0.96;0.96;0.96;0.96;0.96;...% C по слоям
   0;0;0;0;0;];% TiC по слоям
[t,yr]=ode15s(@fdrskor,tspan,y0,[],h,n,v,tnach,tkon,a);
grt=(yr(:,1)-yr(:,n/2))./(h/2);maxgrt=max(grt);
grtic=(yr(:,16)-yr(:,20))/(h/2);maxgrtic=max(grtic);
dy=grtz-maxgrt;
```

```
f2=sum(abs(dy));
function dy=fdrskor(t,y,h,n,v,tnach,tkon,a);
dh=h/n;
 m1=a./(((1.7047e-16.*y(1).^5-1.4279e-12.*y(1).^4+4.4927e-9.*y(1).^3-
6.6011e-6.*y(1).^2+4.7416e-3.*y(1)-5.0323e-
1)*1000*0.259+1010*0.741)*(dh^2));
m2=a./(((1.7047e-16.*y(2).^5-1.4279e-12.*y(2).^4+4.4927e-9.*y(2).^3-
6.6011e-6.*y(2).^2+4.7416e-3.*y(2)-5.0323e-
1)*1000*0.259+1010*0.741)*(dh^2));
m3=a./(((1.7047e-16.*y(3).^5-1.4279e-12.*y(3).^4+4.4927e-9.*y(3).^3-
6.6011e-6.*y(3).^2+4.7416e-3.*y(3)-5.0323e-
1)*1000*0.259+1010*0.741)*(dh^2));
m4=a./(((1.7047e-16.*y(4).^5-1.4279e-12.*y(4).^4+4.4927e-9.*y(4).^3-
6.6011e-6.*y(4).^2+4.7416e-3.*y(4)-5.0323e-
1)*1000*0.259+1010*0.741)*(dh^2));
m5=a./(((1.7047e-16.*y(5).^5-1.4279e-12.*y(5).^4+4.4927e-9.*y(5).^3-
6.6011e-6.*y(5).^2+4.7416e-3.*y(5)-5.0323e-
1)*1000*0.259+1010*0.741)*(dh^2));
tvn=v*t+tnach;
 if tvn>=tkon;tvn=tkon;
 end;
dy=[m1*(tvn-2*y(1)+y(2));...
   m2*(y(1)-2*y(2)+y(3));...
   m3*(y(2)-2*y(3)+y(4));...
   m4*(y(3)-2*y(4)+y(5));...
   m5*(y(4)-y(5));.....
% Po TiO2
(-exp(1.2001e-001)*exp(-5.98e3./y(1)))*y(6)^(0.0024.*y(1)-
0.9385)*y(11)^(-0.0071.*y(1)+20.9198);
(-exp(1.2001e-001)*exp(-5.98e3./y(2)))*y(7)^(0.0024.*y(2)-
0.9385)*y(12)^(-0.0071.*y(2)+20.9198);
(-exp(1.2001e-001)*exp(-5.98e3./y(3)))*y(8)^(0.0024.*y(3)-
```

```
0.9385)*y(13)^(-0.0071.*y(3)+20.9198);
(-exp(1.2001e-001)*exp(-5.98e3./y(4)))*y(9)^(0.0024.*y(4)-
0.9385)*y(14)^(-0.0071.*y(4)+20.9198);
(-exp(1.2001e-001)*exp(-5.98e3./y(5)))*y(10)^(0.0024.*y(5)-
0.9385)*y(15)^(-0.0071.*y(5)+20.9198);
 %po C
(-exp(1.2001e-001)*exp(-5.98e3./y(1)))*y(6)^(0.0024.*y(1)-
0.9385)*y(11)^(-0.0071.*y(1)+20.9198);
 (-exp(1.2001e-001)*exp(-5.98e3./y(2)))*y(7)^(0.0024.*y(2)-
0.9385)*y(12)^(-0.0071.*y(2)+20.9198);
(-exp(1.2001e-001)*exp(-5.98e3./y(3)))*y(8)^(0.0024.*y(3)-
0.9385)*y(13)^(-0.0071.*y(3)+20.9198);
(-exp(1.2001e-001)*exp(-5.98e3./y(4)))*y(8)^(0.0024.*y(4)-
0.9385)*y(14)^(-0.0071.*y(4)+20.9198);
(-exp(1.2001e-001)*exp(-5.98e3./y(5)))*y(10)^(0.0024.*y(5)-
0.9385)*y(15)^(-0.0071.*y(5)+20.9198);
%po TiC
1.3524.*(exp(1.2001e-001)*exp(-5.98e3./y(1)))*y(6)^(0.0024.*y(1)-
0.9385)*y(11)^(-0.0071.*y(1)+20.9198);
1.3524.*(exp(1.2001e-001)*exp(-5.98e3./y(2)))*y(7)^(0.0024.*y(2)-
0.9385)*y(12)^(-0.0071.*y(2)+20.9198);
1.3524.*(exp(1.2001e-001)*exp(-5.98e3./y(3)))*y(8)^(0.0024.*y(3)-
0.9385)*y(13)^(-0.0071.*y(3)+20.9198);
1.3524.*(exp(1.2001e-001)*exp(-5.98e3./y(4)))*y(9)^(0.0024.*y(4)-
0.9385)*y(14)^(-0.0071.*y(4)+20.9198);
1.3524.*(exp(1.2001e-001)*exp(-5.98e3./y(5)))*y(10)^(0.0024.*y(5)-
0.9385)*y(15)^(-0.0071.*y(5)+20.9198);];
```

С использованием разработанной программы проводились расчеты для пластины толщиной 0,05м, в составе исходной композиции которой – 82,25 масс.ч. титана, 7,5 масс.ч. углеродных микросфер и 10,25 масс.ч. фенолоформальдегидной смолы, при числе

слоев – 10, в интеравалее скростей 50 град/ч – 350 град/ч по данной программе получили следующие технологические параметры, предствленные в табл. 4.

Таблица 4.

Технологические параметры карбидизации пластины из пенокарбида титана

Скорость нагрева, град/ч	154.3935
Найденный максимальный гардиент, г/м	799.9958
Заданный градиент, град/м	800
Время достижения максимального температурного градиента,ч	10. 5000
Время достижения максимального градиента по пенокарбиду титана,ч	2.7500
Полное время карбидизации, ч	14,5000
Время нагрева, ч	11.0562
Время выдержки, ч	3.4438

Графики измения температурного градиента и гардиента по пенокарбиду титана приведены на рис. 19.

Из рис. 19 видно, что максимальный температурный градиент при скорости нагрева изделия не превышает заданный (800град/м).

5.5. Расчет процесса разогрева прессуемых изделий в переменном температурном поле

Прессформа конструктивно состоит из матрицы, пуансона, стола пресса. В зависимости от полученных изделий конструкции прессформ различны, при этом различны и способы нагрева (или охлажде-

ния). Например, при производстве изделий из термореактивных полимеров или стеклоизделий в прессформах происходит отверждение и они подвергаются принудительному обогреву. При производстве изделий из термопластичных полимеров для охлаждения впрыснутой из термопласта автомата пресскомпозиции необходимо охлаждать прессформу, поддерживая в ней определенную температуру. Для охлаждения или нагрева прессформ применяют или специальные нагревательные плиты или в самих прессформах устанавливают каналы в которых, в случае электрообогрева, устанавливают омические нагреватели, или в случае же обогрева жидким теплоносителем, в эти каналы подается теплоноситель (например, пар) при нагреве прессформы, или охлаждающая жидкость – при охлаждении.

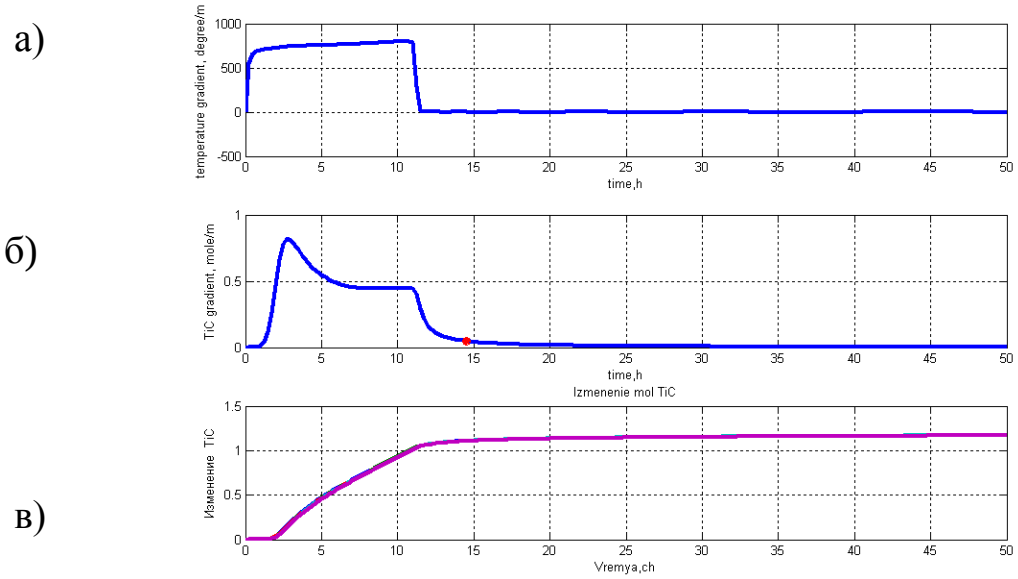


Рис.19. Изменение значения текущего градиента температуры(а), текущего значения градиента карбида титана(б) в процессе карбидизации, образование пенокарбида титана(в)

Рассмотрим прессформу с электрообогревом, когда электронагреватели установлены в теле матрицы и пуансона. В этом случае первоисточником тепла будет являться собственно электронагреватель, в котором при прохождении электрического тока выделяется джоулево тепло.

Мощность подведенной электроэнергии (Вт): $Q_{эл} = U \cdot I$, где U – напряжение, В; I – ток, А. Это тепло будет расходоваться на нагрев собственно массы матрицы и пуансона на нагрев пресуемого материала, потерь тепла прессформой в окружающую среду и в стол пресса. Следует отметить, что прессформа от стола пресса (плиты пресса со стороны матрицы и пуансона) разделена теплоизоляционной прокладкой.

На основании изложенного, блок-схема математического описания теплового режима рассматриваемой прессформы (матрица и плита стола пресса) будет иметь вид, приведенный на рис.20.

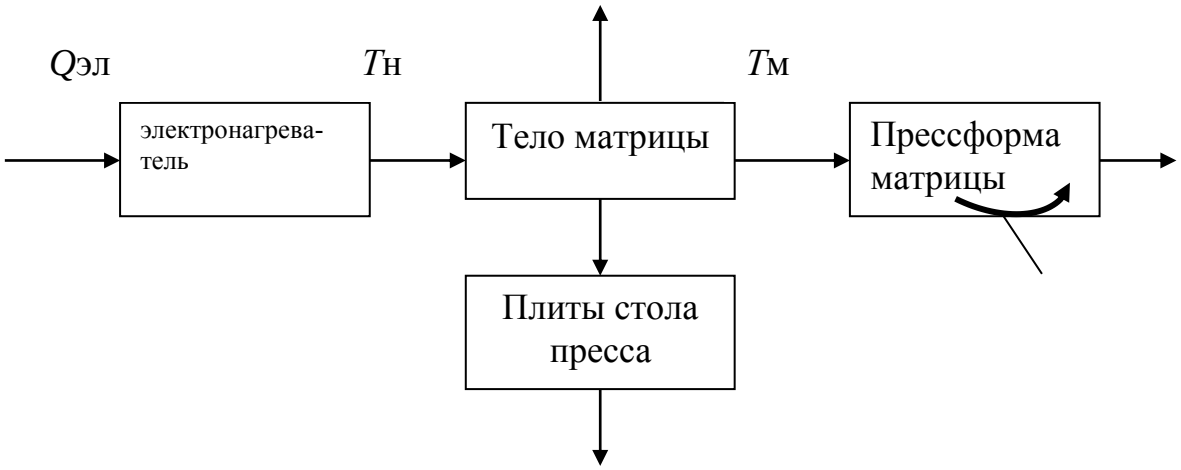


Рис. 20. Блок-схема нагрева матрицы пресс-формы

Тепловой режим матрицы и пуансона поддерживается программной автоматической системой регулирования по заданным законам. Объектом исследования является стадия прессования, которая характеризуется следующими параметрами: скоростью нагрева прессуемой композиции до заданной температуры и толщиной образца. Температура, до которой нагревают пресс-порошок, зависит от того, проводят ли прессование с последующей выдержкой или без нее. Практически установлено, что продолжительность выдержки заготовок при температуре прессования составляет 1-2 мин на 1 мм толщины заготовки.

Практика показывает, что неравномерный обогрев пресс-заготовок приводит к получению изделий с неодинаковой структурой, поэтому математическая модель должна позволить изучить распределение температур по толщине изделия и определить максимальный градиент, возникающий в изделии. Процесс нагрева пресс-композиции при заданных законах изменения температуры матрицы и пуансона описывается уравнением нестационарной теплопроводности вида

$$\frac{\partial T(h,\tau)}{\partial \tau} = a\frac{\partial^2 T(h,\tau)}{\partial h^2}$$

при заданных начальных и граничных условиях:

$$T(h,0) = T_{нач};\quad T(0,\tau) = T_{мат}(\tau);\qquad T(H,\tau) = T_{пуан}(\tau).$$

В уравнении обозначено: $h,\ \tau$ - текущие толщина изделия и время нагрева; $a = \lambda/(c\rho)$ - температуропроводность материала; λ, c, ρ - теплопроводность, теплоемкость и плотность прессуемого материала.

Дифференциальное уравнение в частных производных (50) может быть решено с использованием метода конечных разностей, который позволяет свести его к системе обыкновенных дифференциальных уравнений (задача Коши). Полученная система дифференциальных уравнений имеет вид:

$$\frac{dT_1}{d\tau} = \frac{a}{\Delta h^2}(T_{мат}(\tau) - 2T_1 + T_2)$$

$$\frac{dT_j}{d\tau} = \frac{a}{\Delta h^2}(T_{j-1} - 2T_j + T_{j+1})$$

$$\frac{dT_n}{d\tau} = \frac{a}{\Delta h^2}(T_{n-1} - 2T_n + T_{пуан}(\tau))$$

где $T_{мат}(\tau), T_{пуан}(\tau)$ – законы изменения температур матрицы и пуансона;

$\Delta h = \frac{H}{n}$ - элементарная толщина слоя, H – полная толщина изделия , n - число слоев.

Для решения уравнений математического описания нагрева прессуемого изделия разработана *Matlab*-программа. На основании этой программы можно рассчитать распределение температур в различных сечениях образца и максимальный градиент температур в процессе прессования. Расчетные данные приведены на рис. 21.

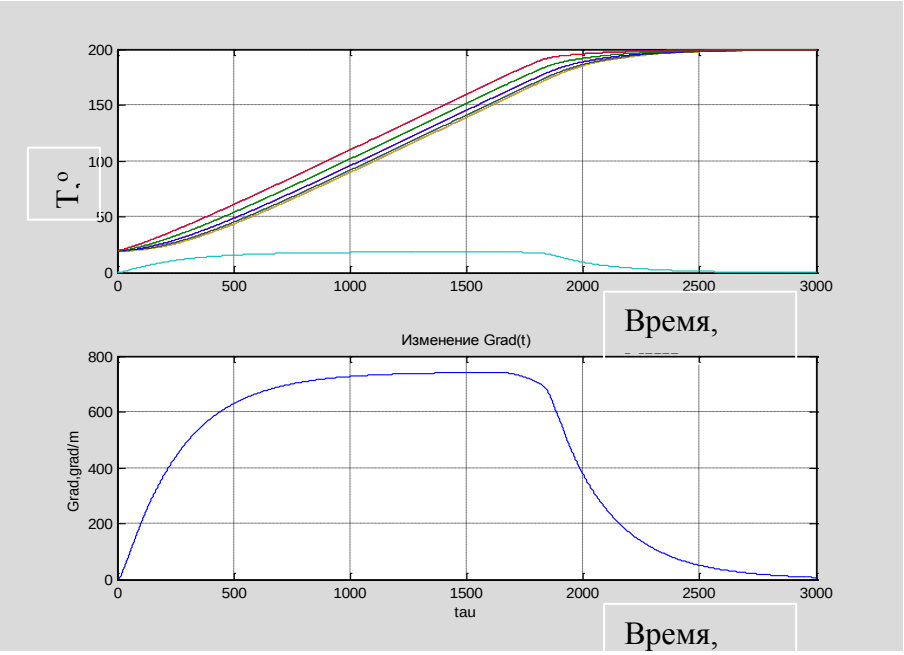


Рис. 21. Изменение температуры в различных сечениях образца и максимального градиента температуры в процессе прессования

Программа для расчета распределения температур по толщине прессуемого изделия и максимального градиента температуры в процессе нагрева

```
%Script-файл для press.m с передачей параметров p1, p2, l.
% в M-файл функцию
% В комплект программ для расчета распределения температур в
%прессуемом изделии входят:
% а) вызывающая программа - Sckipt_progpres.m;
% б) программа для решения систем дифференциальных уравнений
%     нагрева изделий в переменном температурном поле с именем
progpres.m
```

```
% kb, kn - скорость нагрева пуансона и матрицы, град/с
% l-толщина образца, м
% a-температуропроводность прессуемого материала,м^2/с
% n - число элементарных слоев
% (n-1) - число искомых температур
% tn - начальная температура прессуемого материала, град
% ddt - время вывода на печать
% ti - конечное время интегрирования
% y0 - начальные условия для решения дифференциальных уравнений
% tpres - температура прессования
n=8;l=0.05;kb=0.1;a=1e-06;
tpres=200;tfin=3000;kn=kb;tn=20;tspan=[0:tfin];
for i=1:n-1;y0(i)=tn; end;
[t,y]=ode15s(@progpres,tspan,y0,[],kb,kn,l,tpres,a);
dtt=[y(:,1)-y(:,n/2)]';
grad=dtt/(l/2);
subplot(2,1,1);
plot(t,y,t,dtt);grid on;
subplot(2,1,2);
plot(t,grad);grid on;
xlabel('tau');ylabel('Grad,grad/m ');
title('Изменение Grad(t)');
maxgr=max(grad);
disp('максимальный градиент =');disp(maxgr);
function dy=progpres(t,y,kb,kn,l,tpres,a);
m=length(y);
a=1e-6;dl=l/(m+1);b=a/dl^2;
      tpb=kb*t+20;
      if tpb>=tpres
   tpb=tpres;
  end;
tpn=tpb;
%disp([t tpb]);
```

```
dy(1,1)=b*tpb-2*b*y(1,1)+b*y(2,1);
for i=2:m-1
  dy(i,1)=b*y(i-1,1)-2*b*y(i,1)+b*y(i+1,1);
end;
dy(m,1)=b*y(m-1,1)-2*b*y(m,1)+b*tpn;
```

5.6. Математическое описание и расчет экструдера для режима нормальной эксплуатации (с учетом движения материала в зазоре цилиндра)

Необходимо составить математическое описание экструдера для режима нормальной эксплуатации, при котором в зазоре цилиндра движется перерабатываемый полимер.

Если перерабатываемый материал находится в зонах пластикации и дозирования, то структура потока материала подчиняется однопараметрической диффузионной модели; в зоне загрузки (первая технологическая зона) структура движения материала соответствует модели идеального вытеснения.

Составим математическое описание теплового режима для зоны пластикации в режиме нормальной эксплуатации, учитывая, что в этой зоне начинается выделение тепла в материале за счет пластической деформации.

При движении материала в данной тепловой зоне он нагревается не только за счет передачи тепла от цилиндра, но и за счет тепла, выделившегося за счет пластической деформации $Q_{\text{пл}}$. В соответствии с этим, для снятия избытка тепла, который может привести к перегреву материала, его частично удаляют за счет охлаждения шнека. Кроме того, следует отметить, что в режиме нормальной эксплуатации экструдера температура цилиндра в данной тепловой зоне поддерживается постоянной автоматической системой регулирования.

Следовательно, при составлении математического описания, можно считать, что температура цилиндра в процессе продвижения материала в рассматриваемой тепловой зоне остается постоянной, определяемой видом перерабатываемого материала.

При составлении математического описания для шнека данной зоны будем предполагать, что собственно масса шнека и объем охлаждающей жидкости в шнеке могут быть описаны тепловой моделью идеального смешения. Блок-схема математического описания теплового режима зоны представлена на рис. 22.

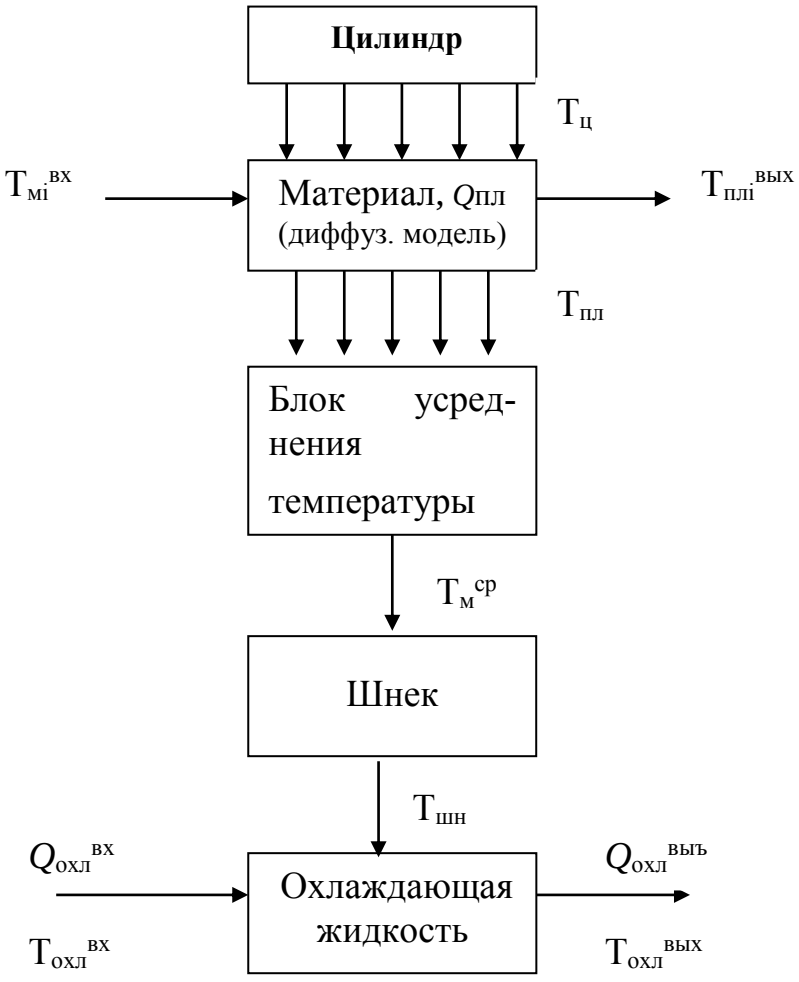


Рис.22. Блок-схема математического описания теплового режима зоны пластикации.

Для каждого теплового блока составим уравнения тепловых балансов в дифференциальной форме:

1) для цилиндра :

$T_{\text{ц}i}(l,\tau) = const$, i – номер тепловой зоны (i = 1, 2, 3);

2) для материала с учетом того, что структура потока материала описывается однопараметрической диффузионной моделью:

$$\frac{d}{d\tau}(\overline{\Delta S_i}\,\Delta l\,\overline{\rho_{\text{м}i}}\,\overline{c_{\text{м}i}}\,T_{\text{м}ij}) = u_{\text{пр}}\Delta S_i\,\overline{\rho_{\text{м}i}}\,\overline{c_{\text{м}i}}(T_{\text{м}ij-1} - T_{\text{м}ij}) +$$

$$+ u_{\text{пр}}\Delta S_i\,\overline{\rho_{\text{м}i}}\,\overline{c_{\text{м}i}}(T_{\text{м}ij+1} - T_{\text{м}ij}) + Q_{\text{пл}} + K_{\text{ц-м}}\Delta S_{\text{ц}ij}(T_{\text{ц}i} - T_{\text{м}ij}) -$$

$$- K_{\text{м-шн}}S_{\text{шн}}(\overline{T_{\text{м}i}} - T_{\text{шн}i}),$$

где $\overline{\Delta S_i}$ – средняя площадь поперечного сечения тепловой зоны; Δl – элементарная длина зоны; $Q_{\text{пл}}$ – тепло пластической деформации на длине Δl; $K_{\text{ц-м}}$ – коэффициент теплоотдачи от цилиндра к материалу; $\Delta S_{\text{ц}ij}$ – внутренняя теплоотдающая поверхность цилиндра на элементарной длине Δl; $T_{\text{ц}i}$ – температура цилиндра; $T_{\text{м}ij}$ – температура материала в j-ом сечении; j – индекс сечения тепловой зоны, j=1,n; n=L/Δl –число элементарных длин зоны.

3) для блока усреднения температур в зоне :

$$\overline{T_{\text{м}i}} = \frac{1}{n}\sum_{j=1}^{n} T_{\text{м}ij};$$

4) для шнека:
$$\frac{d}{d\tau}(M_{\text{шн}i}\,\overline{c_{\text{шн}}}\,T_{\text{шн}i}) = K_{\text{м-шн}}S_{\text{шн}}(\overline{T_{\text{м}i}} - T_{\text{шн}i}) - \alpha_{\text{шн-в}}S_{\text{шн}}(T_{\text{шн}i} - T_{\text{в}i}),$$

где $M_{\text{шн}i}\,\overline{c_{\text{шн}}}\,T_{\text{шн}i}$ – масса, теплоемкость, температура шнека в i-й тепловой зоне; $\alpha_{\text{шн-в}}$ – коэффициент теплоотдачи от шнека к охлаждающей жидкости; $T_{\text{в}i}$ – температура охлаждающей жидкости в i-й тепловой зоне;

5) для охлаждающей жидкости:

$$\frac{d}{d\tau}(M_{\text{в}}\,\overline{c_{\text{в}}}\,T_{\text{в}i}) = F_{\text{в}}\rho_{\text{в}}c_{\text{в}}T_{\text{в}}^{\text{вх}} + \alpha_{\text{шн-в}}S_{\text{шн}}(T_{\text{шн}i} - T_{\text{в}i}) - F_{\text{в}}\rho_{\text{в}}c_{\text{в}}T_{\text{в}i}.$$

Решение представленных уравнений с заданными начальными условиями позволит рассчитать нагрев экструдируемого материала, движущегося в зазоре цилиндра при заданной длине тепловой зоны.

Matlab-программа для расчета процесса нагрева экструдируемого материала в режиме нормальной эксплуатации

```
%File-function for dextrud
function dtm=dextrud(t,tm,u,l,rom,cm,d1,d2,kt,tbx,tc);
n=length(tm);
k1=u*n/l;
k2=kt*d1*l/((d1^2-d2^2)*rom*cm*n);
54
dtm(1,1)=k1*tbx-(k1+k2)*tm(1,1)+k2*tc;
for i=2:n
  dtm(i,1)=k1*tm(i-1)-(k1+k2)*tm(i,1)+k2*tc;
end;
%**************************************
%Skript-file for dextrud
u=0.001;n=10;rom=1200;cm=800;l=0.2;
d1=0.2;d2=0.17;kt=650*n/l;tbx=315;tc=415;
tfin=300;
tspan=[0:tfin];
for i=1:n
  tm0(i)=315;
  end;
[t,tm]=ode15s(@dextrud,tspan,tm0,[],u,l,rom,cm,d1,d2,kt,tbx,tc);
plot(t,tm);grid on;
xlabel('tau');ylabel('Tm ');
```

Пример расчета.

Исходные данные: средняя скорость движения материала u = 0.001м/с;

средняя плотность материала *rom* = 1200 кг/м^3; средняя теплоемкость материала *cm* = 800 Дж/(кг*град); длина тепловой зоны *l* = 0.2 м; входная температура материала в зону *tbx* = 315 K; наружный диаметр цилиндра *d*1 = 0.2 м; внутренний диаметр цилиндра *d*2=0.17 м; число элементарных зон *n*=10.

Примечание: при необходимости изменить исходные данные, их записывают в *Skript-file for dextrud.*

На рис. 23. представлены результаты расчета.

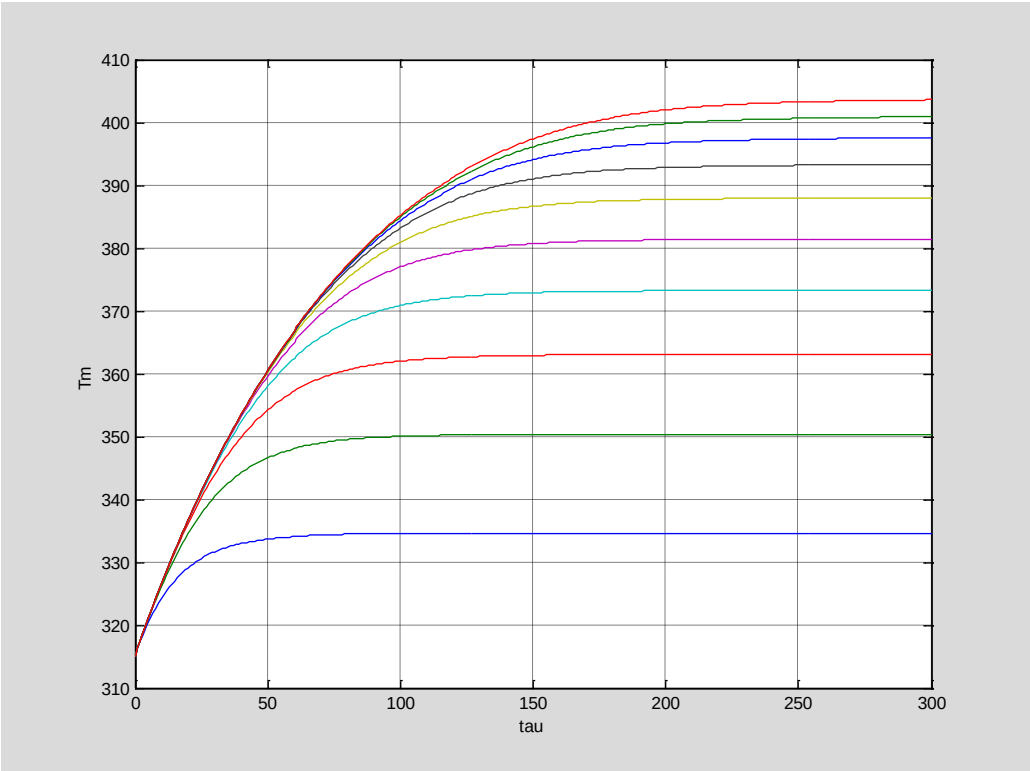


Рис. 23. Изменение температуры экструдируемого материала

5.7. Математическое моделирование процесса охлаждения листа в охладительной ванне

Расчет температурных полей при охлаждении листов термопласта или реактопласта, полученных экструзией или прессованием, на воздухе или в воде имеет большое значение, так как в процессе охлаждения возникает градиент температур по толщине листа, приводящий к развитию усадочных напряжений, в результате чего образуются микротрещины и, следовательно, уменьшается механическая прочность. В связи с этим возникает задача расчета такой температуры в ванне охлаждения и времени пребывания листа в ней, при которых возникающие внутренние напряжения не приводили к механическому повреждению изделия, т.е. градиент температур по толщине листа не должен превышать предельно допустимого значения.

Изменение температуры по толщине листа из термопласта в процессе двухстороннего симметричного охлаждения описывается уравнением нестационарной теплопроводности без внутренних источников тепла вида:

$$\frac{\partial \dot{O}(\tau, h)}{\partial \tau} = \frac{\lambda(T)}{c_p(T)\rho(T)} \frac{\partial^2 T(\tau, h)}{\partial h^2},$$

где $\lambda(T)$ – теплопроводность материала; $c_p(T), \rho(T)$ – теплоемкость и плотность материала; h, τ – текущие толщина листа и время.

При расчете температурных полей охлаждаемых изделий из композиционных материалов на основе термореактивных смол следует учитывать, что изменение температуры листа зависит не только от теплофизических характеристик материала (λ, c, ρ), но и от экзотермичности реакции отверждения. В этом случае температурное поле может быть рассчитано по уравнению:

$$\frac{\partial T(\tau, h)}{\partial \tau} = \frac{\lambda(T)}{c_p(T)\rho(T)} \frac{\partial^2 T(\tau, h)}{\partial h^2} + \frac{q_{\text{вн}}}{c_p(T)\rho(T)},$$

где $q_{\text{вн}}$ – интенсивность внутреннего источника тепла, определяемая скоростью протекания реакции охлаждения:

$$q_{âí} = \Delta Í \cdot \hat{E}(\dot{O}) \cdot (1 - y(\tau, h))^n,$$

где ΔH – тепловой эффект реакции отверждения; $K(T)$ – константа скорости реакции отверждения, которая подчиняется закону Аррениуса;

n – порядок реакции (в первом приближении он может быть принят равным единице); y – степень отверждения.

Кинетика отверждения описывается дифференциальным уравнением вида:

$$\frac{\partial y(\tau, h)}{\partial \tau} = \hat{E}(\dot{O}(\tau, h)) \cdot (1 - y(\tau, h))^n.$$

Таким образом, уравнения будут представлять математическое описание процесса охлаждения реактопласта, а уравнение – для термопласта и при заданных начальных и граничных условиях может быть использовано для расчета температурных полей в процессе охлаждения.

Для решения дифференциальных уравнений в частных производных нами использовался метод конечных разностей, приводящий эти уравнения к системе обыкновенных дифференциальных уравнений в форме Коши с заданными начальными условиями. Matlab-программа для расчета степени отверждения при охлаждении листа приведена ниже:

```
%Skript-file for vannaq1
%Входные данные:
dq=0.05;%Тепловой эффект
ts=[0,1000];%Параметры интегрирования
cp=750;ro=1200;k0=1.4;e=17000;h=0.02;al=0.25;
n=10;dh=h/n;
b=al/(ro*cp*dh^2);
b1=1/(ro*cp);
```

```
tvn=283;
y0=[493;493;493;493;493;0;0;0;0;0];
[t,y]=ode15s(@vannaq1,ts,y0,[],dq,cp,ro,k0,e,h,al,n);
subplot(2,1,1);
plot(t,y(:,1:n/2),t,y(:,n/2)-y(:,1),'.');grid on;
ylabel('Температура по слоям, К');
title('Изменение температуры в процессе охлаждения');
subplot(2,1,2);
plot(t,y(:,n/2+1:n),t,y(:,n),t,y(:,n)-y(:,n/2+1),'.');grid on;
xlabel('Время,с');ylabel('Степень отверждения');
title('Изменение степени отверждения в процессе охлаждения');
```

```
function dy=vannaq1(t,y,dq,cp,ro,k0,e,h,al,n);
%M-file функция для расчета степени отверждения
 % в процессе охлаждения в водяной ванне
% Расчет ведется на полутолщину листа
dh=h/n;
b=al/(ro*cp*dh^2);
b1=1/cp;
tvn=293;
dy=[b*(tvn-2*y(1)+y(2))+k0*exp(-e/(8.31*y(1)))*dq*(1-y(6))/b1;...
   b*(y(1)-2*y(2)+y(3))+k0*exp(-e/(8.31*y(2)))*dq*(1-y(7))/b1;...
   b*(y(2)-2*y(3)+y(4))+k0*exp(-e/(8.31*y(3)))*dq*(1-y(8))/b1;...
   b*(y(3)-2*y(4)+y(5))+k0*exp(-e/(8.31*y(4)))*dq*(1-y(9))/b1;...
   b*(y(4)-y(5))+k0*exp(-e/(8.31*y(5)))*dq*(1-y(10))/b1;...
   k0*exp(-e/(8.31*y(1)))*(1-y(6));...
   k0*exp(-e/(8.31*y(2)))*(1-y(7));...
   k0*exp(-e/(8.31*y(3)))*(1-y(8));...
   k0*exp(-e/(8.31*y(4)))*(1-y(9));...
   k0*exp(-e/(8.31*y(5)))*(1-y(10))];
```

Результаты расчета, проведенного по этой программе, приведены на рис. 24.

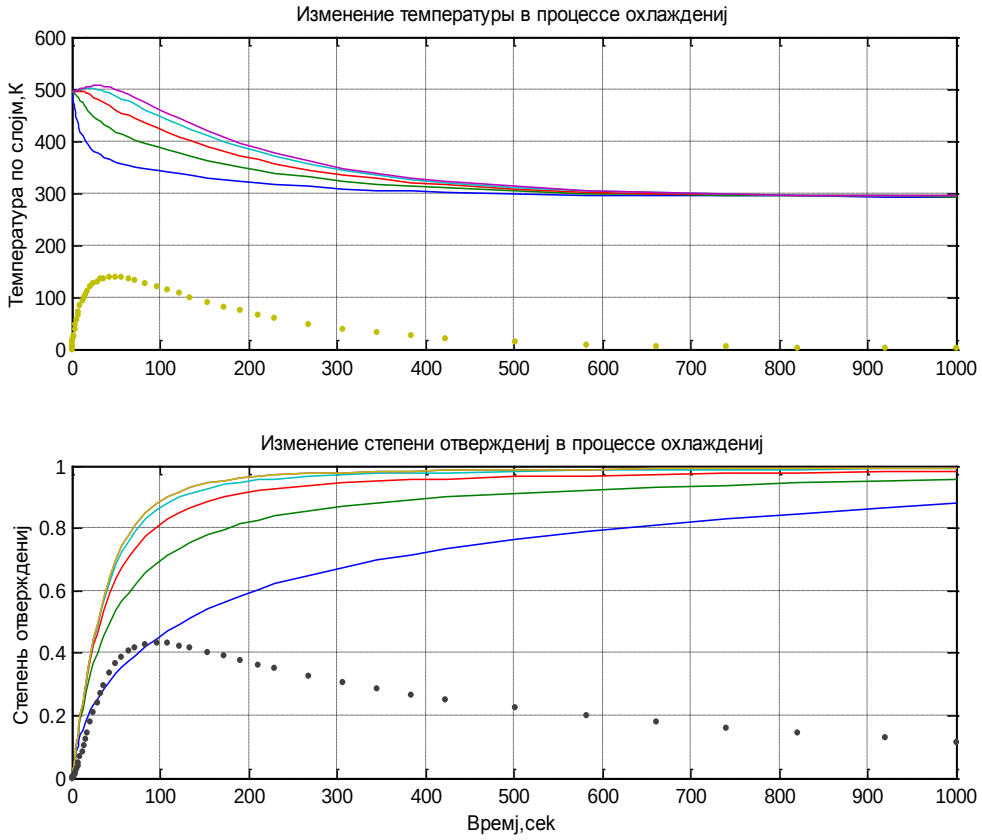


Рис. 24. Изменение температуры и степени отверждения полимерного листа в процессе охлаждения

5.8. Примеры расчета автоматических систем регулирования на заданные запасы устойчивости

В теории автоматического регулирования применяется ряд методов расчета параметров настроек регуляторов, обеспечивающих устойчивость автоматических систем регулирования и заданные значения показателей качества.

В данных примерах рассматривается широко распространенный частотный метод, теоретической основой которого является критерий устойчивости Найквиста. Этот метод позволяет:

- определить на основе анализа амплитудно-фазочастотной характеристики разомкнутой АСР устойчивость замкнутой АСР;

- рассчитать параметры настроек регуляторов, обеспечивающие заданные запасы устойчивости АСР.

Согласно критерию устойчивости Найквиста, для того чтобы замкнутая АСР была устойчива, необходимо и достаточно, чтобы годограф разомкнутой АСР при изменении частоты от нуля до бесконечности не охватывал на комплексной плоскости точку с координатами (-1, *i*0).

Если годограф разомкнутой АСР проходит через точку (-1,*i*0), замкнутая АСР находится на колебательной границе устойчивости (нейтральная АСР) и в системе при этом возникает колебательный переходный процесс с постоянной амплитудой колебания.

Если годограф разомкнутой АСР охватывает точку (-1, *i*0), замкнутая АСР неустойчива. Примеры годографов разомкнутых статических и астатических АСР приведены на рис. 25.

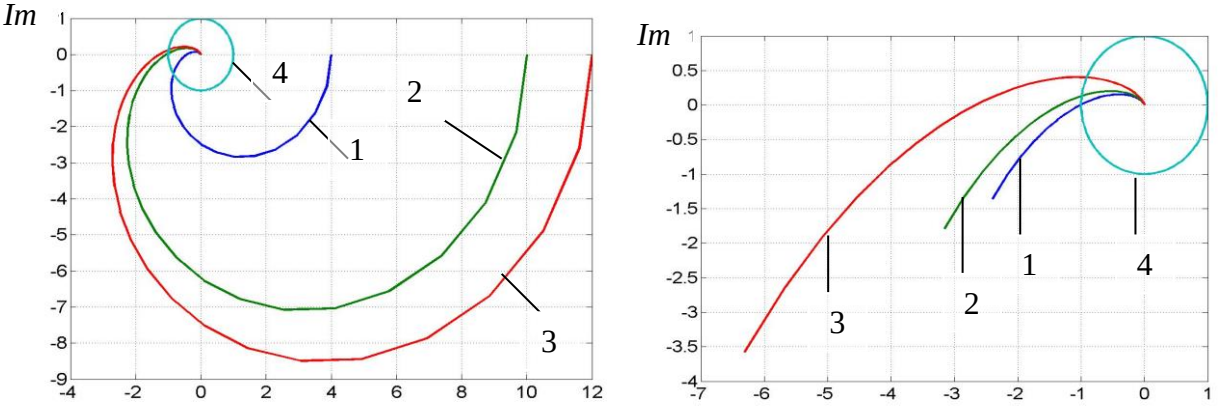


Рис.25. Годографы статических (а) и астатичеких (б) АСР: 1– устойчивая; 2 – нейтральная; 3 – неустойчивая; 4 – окружность единичного радиуса

Таким образом, чтобы замкнутая АСР находилась на колебательной границе устойчивости, необходимо выполнение следующего условия: $W_{об}(i\omega)W_{р}(i\omega) = -1$,

где $W_{об}(i\omega)$ – амплитудно-фазочастотная характеристика эквивалентного объекта управления; $W_{р}(i\omega)$ – амплитудно-фазочастотная характеристика автоматического регулятора.

Из представленного уравнения можно найти амплитудно-частотную (АЧХ) и фазочастотную (ФЧХ) характеристики разомкнутой АСР и записать условия, при которых замкнутая АСР будет находится на границе устойчивости.

Учитывая, что модуль частотной характеристики есть АЧХ, а аргумент частотной характеристики – ФЧХ, эти условия будут иметь вид:

$$\left|W_{\text{îá}}(i\omega)\right|\left|W_{\text{р}}(i\omega)\right| = 1 \quad ,$$

$$\arg W_{\text{об}}(i\omega) + \arg W_{\text{р}}(i\omega) = e^{-i\pi}.$$

Решая эти уравнения, можно найти значения параметров настроек того или иного регулятора, при которых замкнутая АСР будет находиться на границе устойчивости, т.е. годограф разомкнутой АСР будет проходить через точку с координатами (-1, *i*0).

Однако определение устойчивости АСР для оценки автоматических систем регулирования с точки зрения их практической пригодности недостаточно. Любая АСР должна обладать определенными значениями показателей качества регулирования. Качество процесса регулирования для стабилизирующих АСР обычно оценивают по переходному процессу в устойчивой АСР при нанесении на ее вход ступенчатого управляющего воздействия.

Основными показателями качества регулирования являются: время регулирования $\tau_{\text{р}}$; максимальное перерегулирование $\sigma_{\max}$ (или максимальная динамическая ошибка); статическая ошибка $\Delta_{\text{ст}}$; степень затухания ψ.

Перечисленные показатели качества регулирования находятся из графика переходного процесса в замкнутой АСР (рис. 26).

Временем регулирования $\tau_{\text{р}}$ называется время, в течение которого, начиная с момента приложения воздействия на систему, отклонение значений регулируемой переменной $\varphi(\tau)$ от ее установившегося

значения $\varphi_{уст}$ будет больше некоторого, наперед заданного значения ε. В практике принято, что по истечении времени регулирования отклонение регулируемой величины $\varphi(\tau)$ от установившегося значения $\varphi_{уст}$ должно быть $\varepsilon \leq$ 5 %. Время регулирования определяет быстродействие АСР.

Максимальным перерегулированием σ_{max} называют отношение максимального отклонения $\Delta\varphi_{max}$ регулируемой переменной $\varphi(\tau)$ относительно заданного значения $\varphi_{зад}$, выраженное в процентах:

$$\sigma_{max} = \frac{\varphi_{max} - \varphi_{зад}}{\varphi зад} 100 = \frac{A_1}{\varphi_{зад}} 100.$$

Статической ошибкой регулирования $\Delta_{ст}$ называют разность между заданным $\varphi_{зад}$ и установившемся $\varphi_{уст}$ значениями регулируемой переменной:

$$\Delta_{ст} = \varphi_{зад} - \varphi_{уст}.$$

Степенью затухания ψ называют отношение разности двух соседних амплитуд одного знака кривой переходного процесса к большей из них:

$$\psi = \frac{A_1 - A_3}{A_1} = 1 - \frac{A_3}{A_1}.$$

Если, например, по условиям технологии требуется, чтобы при колебательном переходном процессе амплитуда каждого последующего отклонения регулируемой переменной $\varphi(\tau)$ уменьшилась в K раз по отношению к амплитуде предыдущего отклонения ($K = \frac{A_1}{A_3}$), то степень затухания рассчитывается по уравнению: $\psi = 1 - 1/K$.

Если мы хотим получить переходный процесс таким, чтобы за время регулирования $\tau_р$ было совершено одно полуколебание с точностью $\pm$ 0,05 $\varphi_{уст}$ (K при этом будет равно 1/0,05 = 20), степень затухания ψ должна быть равна 0,95.

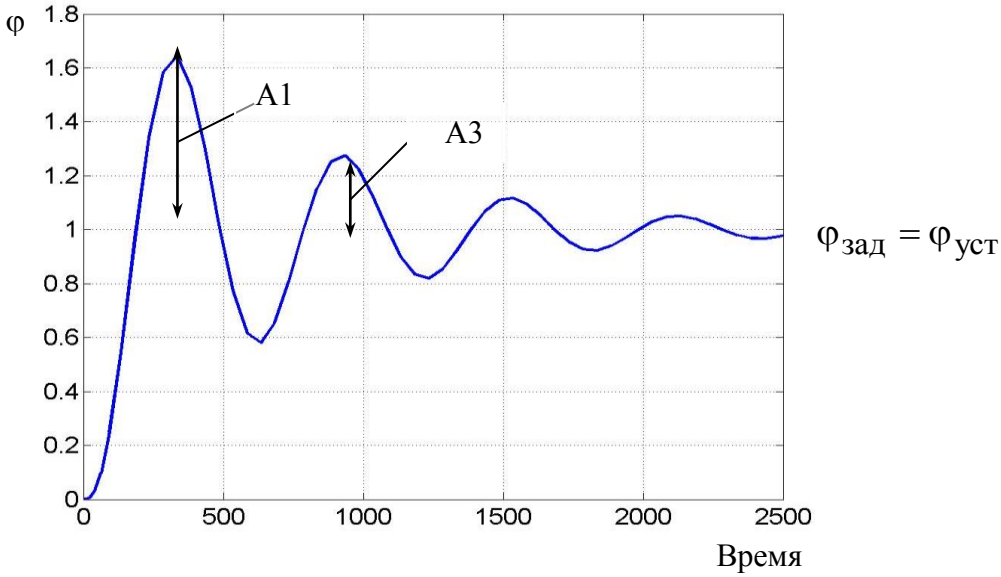


Рис. 26. К определению показателей качества регулирования

В инженерной практике принято, что рассчитанная АСР пригодна для технической эксплуатации, если степень затухания находится в пределах $0{,}75 \le \psi \le 0{,}9$.

Если рассчитанная амплитудно-фазовая характеристика не охватывает “опасную” точку (-1, *i*0), но при $\theta_{раз}(\omega) = -\pi$ ее амплитуда несущественно отличается от единицы, такая теоретически устойчивая АСР может оказаться практически неустойчивой, ибо могут иметь место неточности при составлении математического описания элементов АСР или во время работы системы могут возникнуть непредвиденные отклонения параметров элементов АСР, которые приведут к такому изменению вида амплитудно-фазочастотной характеристики, которое приведет к неустойчивости системы.

В соответствии с этим введено *понятие о запасе устойчивости* АСР. Численно запас устойчивости определяется двумя характеристиками: запасом устойчивости по модулю и запасом устойчивости по фазе.

Если при приближении вектора АФЧХ разомкнуой АСР при частоте $\omega = \omega_\pi$ (рис. 27) справа к точке с координатами (-1, *i*0) устой-

чивая АСР приближается к колебательной границе устойчивости, то, следовательно, степень устойчивости замкнутой АСР находится в прямой зависимости от степени удаления точки пересечения АФЧХ разомкнутой АСР с отрицательной вещественной полуосью до точки с координатами (-1, *i*0).

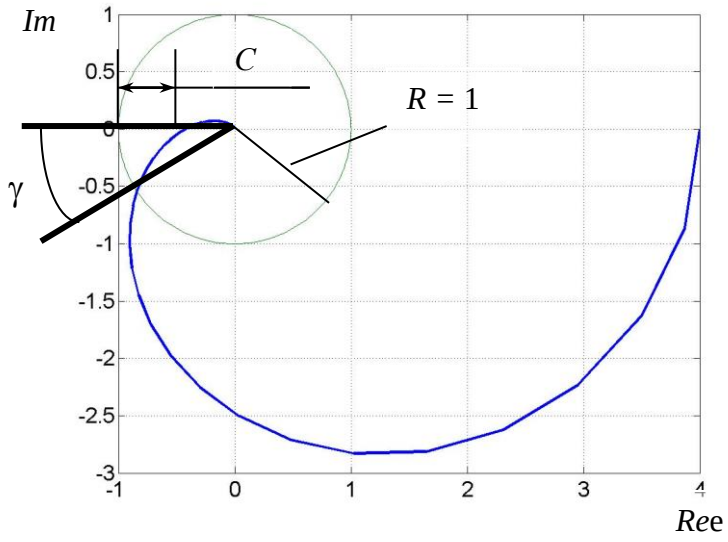


Рис. 27. Определение запаса устойчивости по модулю *C* и по фазе γ

Расстояние от точки пересечения АФЧХ разомкнутой АСР с отрицательной вещественной полуосью (см. рис. 3) до точки с координатами (-1, *i*0) называют *запасом устойчивости по модулю* C .

Угол γ, образованный вещественной отрицательной полуосью *Re* и лучем, проведенным из начала координат через точку пересечения АФЧХ разомкнутой АСР с окружностью единичного радиуса (*R* = 1), имеющий центр в начале координат называется *запасом устойчивости по фазе.*

Запас устойчивости по модулю C показывает, насколько может измениться модуль АФЧХ разомкнутой АСР для выхода замкнутой АСР на границу устойчивости при неизменных фазовых соотношениях.

Под *возмущающими воздействиями по модулю* понимаются воздействия, вызывающие увеличение коэффициента передачи разомкнутой

АСР без изменения фазы вектора АФЧХ на всех частотах. Под *возмущающими воздействиями по* фазе понимаются воздействия, вызывающие увеличение фазы векторов АФЧХ пропорционально их частоте без изменения их модуля.

Для обеспечения заданного запаса устойчивости замкнутой АСР по модулю C необходимо, чтобы годограф разомкнутой АСР пересекал вещественную отрицательную полуось *Re* на расстояние $C^{зад}$ от точки с координатами (-1, *i*0) справа от нее.

Таким образом, если известна амплитудно-фазочастотная характеристика разомкнутой АСР $W_{раз}(i\omega_\pi)$, то условие обеспечения заданного запаса устойчивости по модулю $C^{зад}$ запишется так:

$$\arg W_{раз}(i\omega_\pi) = -\pi \ ,$$

$$\left|W_{раз}(i\omega)\right| = 1 - C^{зад},$$

где $\arg W_{раз}(i\omega_\pi)$ – фазочастотная характеристика (ФЧХ) разомкнутой АСР; $\left|W_{раз}(i\omega)\right|$ – амплитудно-частотная характеристика разомкнутой АСР.

Условия обеспечения заданного запаса по фазе $\gamma^{зад}$ запишется в виде:

$$\arg W_{раз}(i\omega) = -\pi + \gamma^{зад} \ ,$$

$$\left|W_{раз}(i\omega)\right| = 1 \ .$$

Если требуется, чтобы замкнутая АСР имела заданные запасы устойчивости по модулю $C^{зад}$ и по фазе $\gamma^{зад}$ должны выполняться следующие условия:

$$\arg W_{раз}(i\omega) = -\pi + \gamma^{зад} \ ,$$

$$\left|W_{раз}(i\omega)\right| = 1 - C^{зад}$$

Таким образом, приведенные выше соотношения позволяют рассчитать параметры настроек выбранных автоматических регулято-

ров, которые обеспечат замкнутой АСР необходимые запасы устойчивости. Эти алгоритмы расчета реализованы в *Matlab*-программах, которые будут рассмотрены ниже.

Matlab-программы и примеры расчетов параметров настроек регуляторов на заданные запасы устойчивости по модулю

Расчет параметров настройки П-регулятора

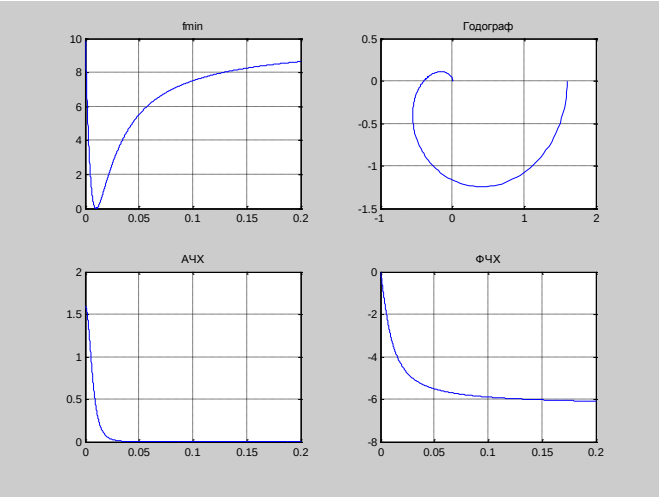
Расчет коэффициента усиления П-регулятора на заданный запас устойчивости по модулю cz = 0,6
на примере 4-х емкостного объекта управления с постоянными времени t0=[100 100 100 100];

M-файл функция имеет вид:

```
function f=krp(w,t0,cz);
f=(-sum(atan(t0'*w))+pi).^2;
%************************************
% Skpipt fyle для расчета П-регулятора на Cz
% Входные данные
t0=[100 100 100 100];cz=0.6;
%************************************************************
wfz=fminbnd(@krp,0,3,[],t0,cz);
kr=((prod(1+(t0'*wfz).^2).^0.5))*(1-cz);
ww=0:.0001:0.2;
ar=kr./((prod(1+(t0'*ww).^2).^0.5));
fr=-sum(atan(t0'*ww));
fmin=(-sum(atan(t0'*ww))+pi).^2;
re=ar.*cos(fr);im=ar.*sin(fr);
subplot(2,2,1);
plot(ww,fmin);grid on;title('fmin');
subplot(2,2,2);
plot(re,im);grid on;title('Годограф');
subplot(2,2,3);
plot(ww,ar);grid on;title('АЧХ')
subplot(2,2,4);
plot(ww,fr);grid on;title('ФЧХ');
disp('Коэффициент  П-регулятора kr=');disp(kr);
disp(' wfz=');disp(wfz);
```

Расчетные данные:

Коэффициент П-регулятора kr=1.5989
wfz=0.0100



Расчет параметров настройки ПИ-регулятора

Расчет коэффициента усиления ПИ-регулятора на заданный запас устойчивости по модулю cz = 0,6
на примере 4-х емкостного объекта управления с постоянными времени t0=[100 100 100 100];

M-файл функция имеет вид:

```
 function f=krpim(w,t0,cz,tiz);
f=(-sum(atan(t0'*w))-atan(1./(tiz*w))+pi).^2;
%**************************************
%Skript fyle для ПИ
% Входные данные
t0=[100 100 100 100];cz=0.6;tiz=100;
ww=0.001:.0001:0.05;
%*************************
wfz=fminbnd(@krpim,0.0001,0.05,[],t0,cz,tiz);
kr=(1-cz)*((prod(1+(t0'*wfz).^2).^0.5))./(1+(1./(wfz*tiz)).^2).^0.5;
```

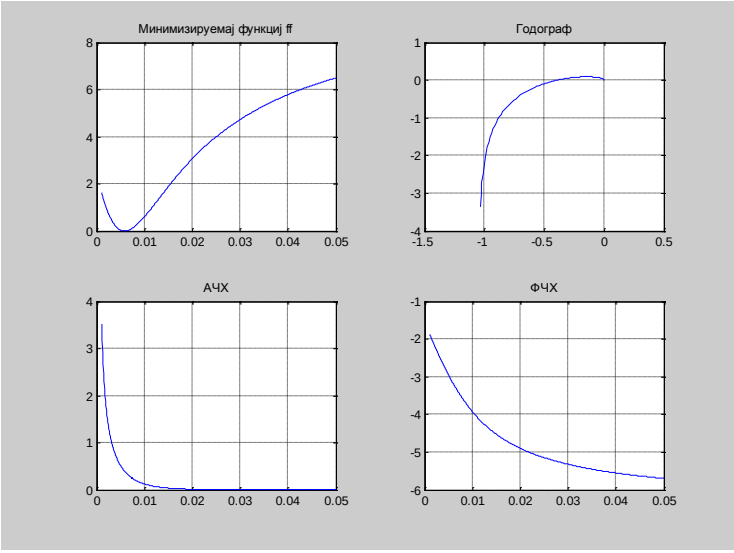
```
ff=(-sum(atan(t0'*ww))+atan(-1./(tiz*ww))+pi).^2;
subplot(2,2,1);
plot(ww,ff);grid on;title('Минимизируемая функция ff');
% Годограф разомкнутой АСР
ar=kr./((prod(1+(t0'*ww).^2).^0.5)).*(1+(-1./(ww*tiz)).^2).^0.5;
fr=-sum(atan(t0'*ww))+atan(-1./(tiz*ww));
re=ar.*cos(fr);im=ar.*sin(fr);
subplot(2,2,2);
plot(re,im);grid on;title('Годограф');
subplot(2,2,3);
plot(ww,ar);grid on;title('АЧХ')
subplot(2,2,4);
plot(ww,fr);grid on;title('ФЧХ');
disp('Коэффициент усиления ПИ-регулятора kr=');disp(kr);
disp('Время изодрома tiz=');disp(tiz);
disp('Частота в минимуме wfz=');disp(wfz);
```

Расчетные данные:

Коэффициент усиления ПИ-регулятора kr= 0.3573

Время изодрома tiz= 100

Частота в минимуме wfz= 0.0058



Расчет параметров настройки ПИД-регулятора

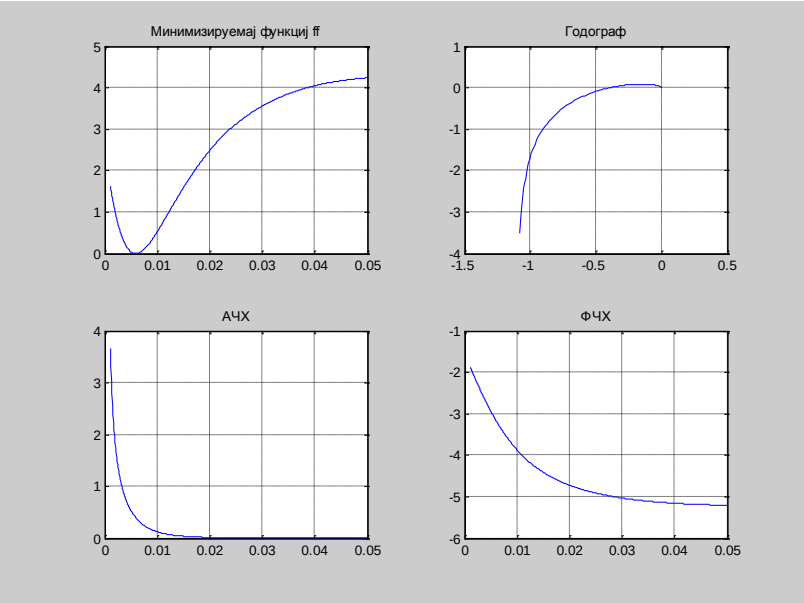
```
%Расчет коэффициента усиления ПИД-регулятора
%на заданный запас устойчивости по модулю cz = 0,6 на примере 4-х емкостного объекта управления
 с постоянными времени t0 =[100 100 100 100];
```

M-файл функция имеет вид:

```
function f=krpidm(w,t0,cz,tiz,td);
f=(-sum(atan(t0'*w))+atan(td*w-1./(tiz*w))+pi).^2;
%**********************************
%Skript fyle для ПИД
% Входные данные
t0=[100 100 100 100];cz=0.6;ww=0.001:.0001:0.05;tiz=100;td=10;
%************************************************
wfz=fminbnd(@krpidm,0.0001,0.05,[],t0,cz,tiz,td);
kr=(1-cz)*((prod(1+(t0'*wfz).^2).^0.5))./(1+(td*wfz-1./(wfz*tiz)).^2).^0.5;
ff=(-sum(atan(t0'*ww))+atan(td*ww-1./(tiz*ww))+pi).^2;
subplot(2,2,1);
plot(ww,ff);grid on;title('Минимизируемая функция ff');
% Годограф разомкнутой АСР
ar=kr./((prod(1+(t0'*ww).^2).^0.5)).*(1+(td*ww-1./(ww*tiz)).^2).^0.5;
fr=-sum(atan(t0'*ww))+atan(td*ww-1./(tiz*ww));
re=ar.*cos(fr);im=ar.*sin(fr);
subplot(2,2,2);
plot(re,im);grid on;title('Годограф');
subplot(2,2,3);
plot(ww,ar);grid on;title('АЧХ')
subplot(2,2,4);
plot(ww,fr);grid on;title('ФЧХ');
disp('Коэффициент усиления ПИД-регулятора kr=');disp(kr);
disp('Время изодрома tiz=');disp(tiz);
disp('Время предварения td=');disp(td);
disp('Частота в минимуме wfz=');disp(wfz);
```

Расчетные данные:

Коэффициент усиления ПИД-регулятора kr=0.3732
Время изодрома tiz=100
Время предварения td=10
Частота в минимуме wfz=0.0058



Расчет параметров настройки ПД-регулятора

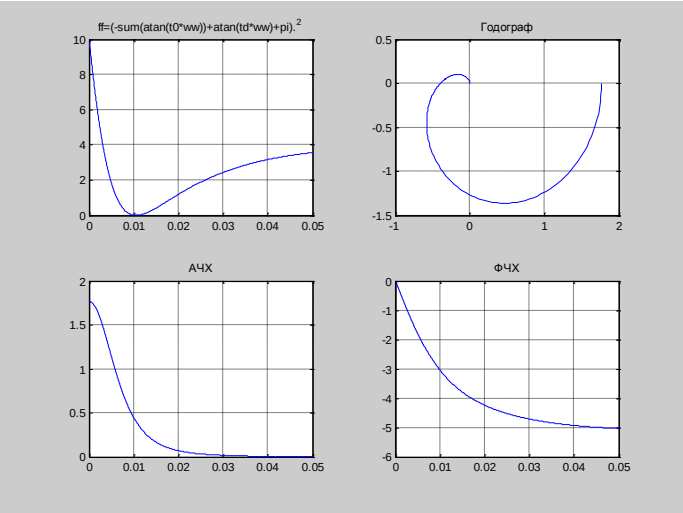
%Расчет коэффициента усиления ПД-регулятора
%на заданный запас устойчивости по модулю cz = 0,6 на примере 4-х емкостного объекта управления
с постоянными времени t0 =[100 100 100 100];

Matlab-программа
M-файл функция имеет вид:
function f=krpdm(w,t0,cz,td);

```
f=(-sum(atan(t0'*w))+atan(d*w)+pi).^2;%end
*****************************************
%Skript fyle для ПД
% Входные данные
t0=[100 100 100 100];cz=0.6;ww=0:.0001:0.05;td=10;
% *****************************************
wfz=fminbnd(@krpdm,0.0001,0.05,[],t0,cz,td);
kr=(1-cz)*((prod(1+(t0'*wfz).^2).^0.5))./(1+(td*wfz).^2).^0.5;
ff=(-sum(atan(t0'*ww))+atan(td*ww)+pi).^2;
subplot(2,2,1);
plot(ww,ff);grid on;title('ff=(-sum(atan(t0*ww))+atan(td*ww)+pi).^2');
%*************************************************
% Годограф разомкнутой АСР
ar=kr./((prod(1+(t0'*ww).^2).^0.5)).*(1+(td*ww).^2).^0.5;
fr=-sum(atan(t0'*ww))+atan(td*ww);
re=ar.*cos(fr);im=ar.*sin(fr);
subplot(2,2,2);
plot(re,im);grid on;title('Годограф');
subplot(2,2,3);
plot(ww,ar);grid on;title('АЧХ')
subplot(2,2,4);
plot(ww,fr);grid on;title('ФЧХ');
disp('Коэффициент усиления ПД-регулятора kr=');disp(kr);
disp('Время предварения td=');disp(td);
disp('Частота в минимуме wfz=');disp(wfz);
```

Расчетные данные:

Коэффициент усиления ПД-регулятора kr= 1.7677
Время предварения td=10
Частота в минимуме wfz= 0.0105



5.9. Планирование эксперимента и обработка экспериментальных данных

Обычно перед началом эксперимента нет сведений о воспроизводимости исследуемой системы. Тогда после постановки серий параллельных опытов в одной точке табл. 5 обычно в центре плана (опыты серий разнесены во времени, вычисляют следующие дисперсии серий и, если последние сильно отличаются друг от друга, делают заключение о невоспроизводимости результатов опытов. Для количественной оценки воспризводимости измерений используют критерий Кохрена G_{cox}, согласно которому рассчитывают дисперсии для каждой серии опытов по уравнению

$$S_k^2 = \frac{\sum_{i=1}^{n}\left(y_{kj} - \bar{y}_k\right)^2}{m-1},$$

где $к$ – число серий опытов, $к$=1, 2, …,n; m – число параллельных опытов в $к$-й серии; y_k - среднее значение выходной переменной в $к$-й

серии опытов; сумма дисперсий серий $\sum_{k=1}^{m} S_k^2$. Для проверки воспроизводимости опытов выбирают самую большую из дисперсий $S_{k\,\max}^2$ и делят ее на $\sum_{k=1}^{m} S_k^2$. полученное отношение называют критерием Кохрена

$$G_{cox} = \frac{S_{k\,\max}^2}{\sum_{k=1}^{m} S_k^2}.$$

Если $G_{cox}^{pac} \langle G_{cox}^{таб}\left(f_1, f_2\right)$ при выбранном уровне значимости α (обычно α=0,05) из известных степенях свободы $f_1 = n-1$ и $f_1 = m$, то опыты воспроизводимы, а дисперсии однородны. Однородность дисперсии S_k^2 означает следующее: если проведены однократные повторные наблюдения над величиной $y_к$ при некотором наборе значений входных переменных x_{1j}, x_{2j}, …, x_{lj} (l - число факторов), то полученная дисперсия не будет зависеть от математического ожидания, т.е. не будет зависеть от S_i^2, полученной при повторных наблюдениях для любого другого набора значений независимых переменных x_{1i}, x_{2i}, …, x_{li} (i, j).

Таблица 5

Данные для оценки воспроизводимости результатов

№ серии опытов	Число параллельных опытов				Данные для расчета критерия G_{kox}	
	1	2	…	m	y_s	S^2
1	y_{11}	y_{12}	…	y_{1m}	y_{s1}	S_1^2
2	y_{21}	y_{22}	…	y_{2m}	y_{s2}	S_2^2
n	y_{n1}	y_{n2}	…	y_{nm}	y_{sn}	S_n^2

Для обработки экспериментальных данных с целью определения однородности дисперсий могут быть использованы следующие функции *MATLAB*: *mean()* – расчет среднего значения по строкам, *std()* – расчет средне-квадратичного отклонения по строкам, *max()* – нахождение максимального значения дисперсий, которые рассчитываются через средне-квадратичные отклонения по уравнению: *dis = std()^2*, *sum()* – нахождение суммы дисперсий. Ниже приведен Script_файл обработки данных табл. 5:

```
a=[        ];%заданный массив данных в соответствии с табл.
ys=mean(a,2);%средние значения по строкам
dis=std(a,2).^2;%расчет построчных дисперсий
dismax=max(dis);%определение максимальной дисперсии
sumdis=sum(dis);%нахождение суммы построчных дисперсий
kox=dismax/sumdis; %расчет критерия Кохрена.
% Если дисперсии однородны, рассчитывается средняя дисперсия из
% построчных дисперсий, которая является дисперсией
% воспроизводимости результатов
disobch=sumdis/(n*m);% расчет общей дисперсии воспроизводимости
disp('Критерий Кохрена и дисперсия воспроизводимости');
disp([kox  disobch]);
```

Для проведения экспериментов применяется два подхода. Первый подход – пассивный эксперимент, когда ставится большая серия опытов с поочередным изменением каждой из переменных. Второй подход – активный эксперимент, когда план проведения эксперимента проводится по жесткой математической программе. Активный эксперимент является наиболее рациональным методом планирования.

В системе *MATLAB* для реализации плана активного эксперимента предусмотрена функция *cordexch()*, которая позволяет составить математический план эксперимента при условии, что входные переменные выражены в безразмерной форме и изменяются от -1 до +1. Кодирование производится по следующим соотношениям:

$$x_i = \frac{X_i - X_i^0}{h_i},$$

где x_i – кодированная i-ая входная переменная, X_i – размерная i-ая входная переменная, X_i^0 - среднее значение i-ой входной переменной, h_i - интервал изменения i-ой входной переменной относительно среднего значения.

Формат обращения к функции *cordexch()* имеет следующий вид:
[*x,xs*]=*cordexch(n,m,'model');*
где *x* – собственно план эксперимента, *xs* – расширенная матрица плана эксперимента для расчета коэффициентов регрессии, *n* – число входных переменных, *m* – число экспериментов, *model* – вид уравнения регрессии: '*interaction*' – неполное квадратное уравнение; '*quadratic*' – полное квадратное уравнение; '*purequadratic*' – квадратное уравнение, в котором отсутствуют парные взаимодействия.

Пример. Составить план проведения эксперимента при *n*=2, *m*=9 для получения уравнения регрессии вида

$$yr = b_0 x_0 + b_1 x_1 + b_2 x_2 + b_{12} x_1 x_2 + b_{11} x_1^2 + b_{22} x_2^2$$

В командной строке *MATLAB* набрать:
[x,xs]=cordexch(2,9,'quadratic');
Результатом работы этой функции получено:
Матрица собственно плана эксперимента x

```
x =
    -1.00    1.00
    -1.00   -1.00
    -1.00    0
     1.00    1.00
     1.00   -1.00
     1.00    0
     0       1.00
     0      -1.00
     0       0
```

Расширенная матрица планирования xs

```
xs =
  [  1.00    -1.00     1.00    -1.00     1.00     1.00
     1.00    -1.00    -1.00     1.00     1.00     1.00
     1.00    -1.00     0         0       1.00     0
     1.00     1.00     1.00     1.00     1.00     1.00
     1.00     1.00    -1.00    -1.00     1.00     1.00
     1.00     1.00     0         0       1.00     0
     1.00     0        1.00     0         0       1.00
     1.00     0       -1.00     0         0       1.00
     1.00     0        0        0         0       0    ];
```

В расширенной матрице планирования число столбцов равно числу коэффициентов в уравнении регрессии.

Пример обработки экспериментальных данных с целью получения уравнения регрессии в виде полного квадратного полинома в соответствии с представленным выше планом.

Реализовав этот план, получили значения выходной переменной:

```
ye = 1227.20  41.86  831.01  3528.12  427.21  2174.15  2293.90  150.77
     1418.81
```

MATLAB-программа для расчета коэффициентов регрессии и статистических данных и графической визуализации расчетов приведена ниже:

```
%Script-файл обработки данных активного эксперимента
k=6;n=9;
  x=[1.00    -1.00     1.00    -1.00     1.00     1.00
     1.00    -1.00    -1.00     1.00     1.00     1.00
     1.00    -1.00     0         0       1.00     0
     1.00     1.00     1.00     1.00     1.00     1.00
     1.00     1.00    -1.00    -1.00     1.00     1.00
     1.00     1.00     0         0       1.00     0
     1.00     0        1.00     0         0       1.00
```

```
        1.00          0       -1.00          0          0        1.00
        1.00          0          0           0          0          0];
[k,n]=size(x)
y=[1227.20;41.86;831.01;3528.12;427.21;2174.15;2293.90;150.77;1418.8
1];
b=regress(y,x);%расчет коэффициентов регрессии
yr=x*b; % расчетные значения выходной переменной по получен-
% ному уравнению регрессии
dy1=(y-yr).^2;
sad=sum(dy1)/(n-k);% расчет дисперсии адекватности
ys=mean(y);
dys=(y-ys).^2;
ssr=sum(dys)/(n-1); % дисперсия относительно среднего
fich=ssr/sad;%Критерий Фишера
disp('Коэффициенты регрессии b')
disp(b');
disp('Сравнение экспериментальных и расчетных данных ');
disp('    y       yr          y-yr');
disp([y yr y-yr]);
disp('');
disp('sad      ssr      Fich');
disp([sad ssr fich]);
[x1,x2]=meshgrid([-1:.1:1]);
yy=b(1)+b(2)*x1+b(3)*x2+b(4)*x1.*x2....
   +b(5)*(x1.^2-2/3)+b(6)*(x2.^2-2/3);
surf(x1,x2,yy);box;grid on;
xlabel('x1');
colorbar;
ylabel('x2');
zlabel('yr(x1,x2)');
title('Поверхность отклика функции yr(x1,x2)’);
figure;
```

```
c=contour(x1,x2,yy,10); %построение контурных линий
clabel(c);
grid on;xlabel('x1 ');ylabel('x2');
title('Линии равных уровней функции yr(x1,x2)');
```

Результаты работы программы:

Коэффициенты регрессии b

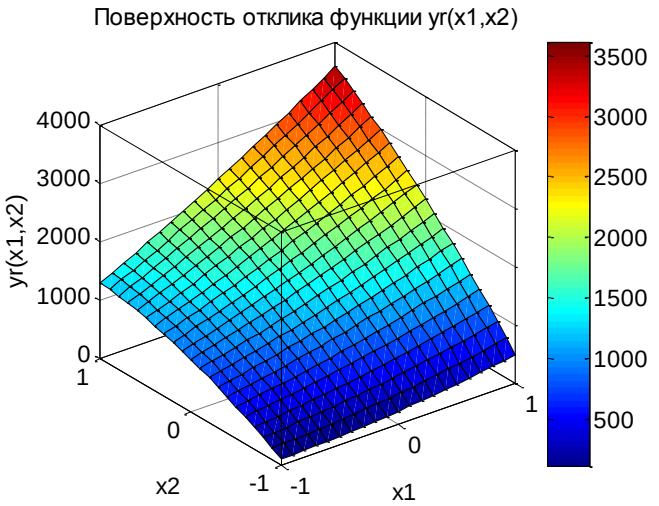
1418.81	671.57	1071.56	478.89	83.76	-196.48

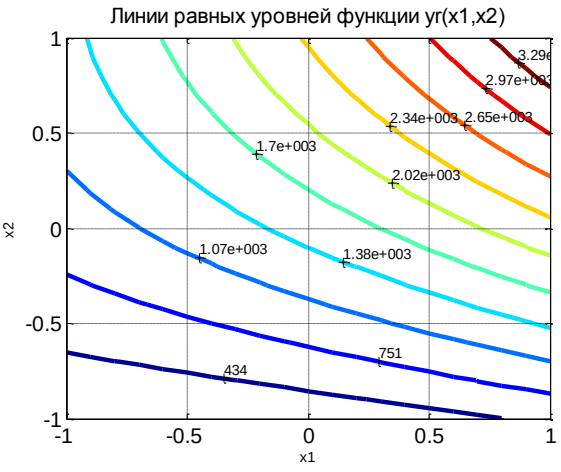
Сравнение экспериментальных и расчетные данные

y	yr	y-yr
1227.20	1227.20	-0.00
41.86	41.86	0.00
831.01	831.01	0.00
3528.12	3528.12	-0.00
427.21	427.21	-0.00
2174.15	2174.15	0.00
2293.90	2293.90	0.00
150.77	150.77	0.00
1418.81	1418.81	-0.00

sad	ssr	Fich
0.00	1325513.26	95436954537.06

Графическая визуализация расчетных данных





ЗАКЛЮЧЕНИЕ

В пособии достаточно подробно излагается материал по вопросам программирования в одной из современных систем компьютерных технологий *MATLAB*, которого достаточно при решении задач химической технологии. Рассмотрены следующие вопросы: алгоритмы решения линейных и нелинейных алгебраических уравнений, дифференциальных уравнений в обыкновенных и частных производных; графическая визуализация экспериментальных и расчетных данных; задачи оптимизации, обратные задачи химической технологии.

Изложенный материал сопровождается многочисленными примерами из области химической технологии.

КОНТРОЛЬНЫЕ ВОПРОСЫ

Раздел 1.

1. При каких условиях используется функция построения графиков *plot(x,y)?*
2. Какие команды *MATLAB* используются при построении нескольких графиков в одном окне при различных размерностях аргумента?
3. Какая функция *MATLAB* позволяет сохранить раздельно все графики?
4. Каким образом осуществляется полиномиальная интерполяция экспериментальных данных в графическом окне команды *plot*()?
5. Какой командой осуществляется формирование пространственной сетки координат для построения объемных графиков?
6. Какими командами строятся графики контурных линий с указанием значения расчетной функции на каждой контурной линии?

7. Какими командами можно построить объемные фигуры в трехмерном пространстве координат?

Раздел 2.

1. Как решается в системе *MATLAB* система линейных алгебраических уравнений?
2. Как решается в системе *MATLAB* система нелинейных алгебраических уравнений?
3. Какими командами находится минимум и максимум функции одной переменной? Формат обращения к функции.
4. Какими командами находится минимум и максимум функции многих переменных? Формат обращения к функции.
5. Какой командой определяется размер массива данных?
6. Как сформировать вектор-строку и вектор-столбец?
7. Как сформировать двухмерных массив данных?
8. Какие решатели системы *MATLAB* используются для решения дифференциальных уравнений?
9. Алгоритм решения дифференциальных уравнений в частных производных?

Раздел 3.

1. Как сформировать передаточную функцию элемента в системе *MATLAB*?
2. Как сформировать передаточную функцию в системе *MATLAB* последовательно соединенных элементов?
3. Как сформировать передаточную функцию в системе *MATLAB* параллельно соединенных элементов?
4. Как сформировать передаточную функцию в системе *MATLAB* замкнутой системе элементов с отрицательной обратной связью?
5. С помощью какой команды рассчитывается и строится график переходного процесса по передаточной функции?

6. Какие команды системы *MATLAB* используются для расчета и построения графиков логарифмических частотных характеристик и диаграммы Найквиста (годографа)?

Раздел 4.

1. Что понимается под визуальным моделированием?
2. Алгоритмы решения систем линейных и нелинейных алгебраических уравнений средствами *SIMULINK* системы *MATLAB.*
3. Для решения каких уравнений используется блок *Transfer Fcn* ?
4. Для чего предназначен блок *Fcn* и с какими сопутствующими блоками он используется?
5. Какие блоки *SIMULINK* используются для визуализации расчетных и экспериментальных данных?

Раздел 5.

1. Как формируется в общем случае целевая функция задач оптимизации?
2. Как формируется в общем случае оптимизируемая функция при решении обратных задач химической технологии?
3. Какими командами системы *MATLAB* находится экстремум оптимизируемой функции?
4. Какая типовая гидродинамическая модель использована при составлении математического описания теплового режима процесса нагрева изделий в переменном температурном поле?
5. Какой метод используется при решении дифференциальных уравнений в частных производных?
6. Что значит привести уравнение в частных производных к задаче Коши?
7. Какой вид передачи тепла от одного элементарного слоя к другому используется в математическом описании?
8. Что такое технологическая и тепловая зоны экструдера?
9. Что такое рабочая точка экструдера?

10. Какой гидродинамической моделью описывается движение экструдируемого материала в режиме нормальной эксплуатации?
11. За счет чего передается тепло от наружной стенки экструдера в окружающую среду и в каком уравнении это учитывается?
12. Что описывает уравнение нестационарной теплопроводности?
13. Чем отличается пассивный эксперимент от активного эксперимента?
14. Какой командой системы *MATLAB* формируется план активного эксперимента. Формат обращения к этой команде.
15. Какой командой системы *MATLAB* рассчитываются коэффициенты уравнения регрессии?
16. Что является критерием адекватности полученного математического описания?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Дьяконов В.П., Круглов В.В. MATLAB* 6.5 SP/7/7 SP1/7 SP2+SIMULINK Инструменты искусственного интеллекта и биоинформатики. Серия «Библиотека профессионала». – М.: СОЛОН-ПРЕСС, 2006. – 456 с. – ISBN 5-98003-255-X.
2. *Половко А.М., Бутусов П.Н. MATLAB* для студента. – Спб.: БХВ-Петербург, 2005. – 320 с. – ISBN 5-94157-595-5.
3. Математическое моделирование процессов переработки пластмасс: учеб. пособие / Н.Н.Барабанов, В.Т.Земскова; Владим. гос. ун-т. – Владимир : Изд-во Владим. гос. ун-та, 2007. – 64 с. – ISBN 5-89368-743-4.

ОГЛАВЛЕНИЕ

Учебное издание

БАРАБАНОВ Николай Николаевич
ЗЕМСКОВА Валентина Тимофеевна

РАСЧЕТЫ ХИМИКО-ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ
В СИСТЕМЕ *MATLAB*

Учебное пособие

Подписано в печать
Формат 60x84/16. Усл.печ. л. Тираж
Заказ
Издательство
Владимирского государственного университета.
600000, Владимир, ул. Горького, 87.