МИНИСТЕРСТВО ОБРАЗОВАНИЯ
И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. Ф. СКОРИНЫ

Кафедра вычислительной математики и программирования

Лабораторный практикум и методические указания

по спецкурсу лекций «Оптимальное проектирование»
для студентов математического факультета

Гомель 1996

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ БЕЛАРУСЬ

Гомельский государственный университет
им. Ф.Скорины
кафедра вычислительной математики и программирования

Можаровский В.В.

Лабораторный практикум и методические указания по спецкурсу лекций "Оптимальное проектирование" для студентов математического факультета

Гомель 1996

Составитель Можаровский В.В.

Рекомендовано к печати методическим советом математического факультета Гомельского государственного университета имени Ф.Скорины.

Лабораторнрый практикум и методические указания по спецкурсу лекций "Оптимальное проектирование" предназначен для студентов математического факультета 01.01 - "Математика", "Прикладная математика", 22.04 - "ПО ВТ и АС".

Введение

Практикум необходим для практического применения методов вычислительной математики, теорий нелинейного программирования и оптимального проектирования в различных отраслях науки, техники и экономики. Он должен обеспечить студентам, специализирующимся в области прикладной математики, более глубокое изучение теоретического материала, а также способствовать приобретению практических навыков, дает возможность строить математические модели сложных устройств, процессов. При построении практических примеров основное внимание уделяется алгоритмам и программам, которые достаточно успешно используются в расчетах.

Каждая лабораторная работа состоит из теоретической части, алгоритмов, программ и заданий по расчету. При построении лабораторных работ использовался опыт создания алгоритмов оптимизации на основе нелинейного программирования [1,3]. Следует отметить, что программы, реализующие градиентные методы, включают одномерный поиск на основе кубической интерполяции. Поэтому для более детального ознакомления с программами нужно использовать работу [1].

В лабораторном практикуме все программы переведены на язык программирования "Паскаль", но не исключено использовать другие языки. Так, например, в работах [1,2] некоторые подобные программы представлены на языке "Бейсик".

Все лабораторные работы выполняются на ПЭВМ студентами математического факультета Гомельского госуниверситета.

Лабораторная работа 1
МЕТОД МИНИМИЗАЦИИ ФУНКЦИИ БЕЗ ОГРАНИЧЕНИЙ ПРЯМЫМ ПОИСКОМ ПО ХУКУ И ДЖИВСУ

Методы прямого поиска являются методами, в которых используются только значения функции и не учитываются их производные. Поиск по Хуку и Дживсу (Hooke, Jeeves) состоит из последовательности шагов исследующего поиска вокруг базисной точки, за которой в случае успеха следует поиск по образцу.

А.Исследующий поиск. При заданной величине шага исследующий поиск начинается в некоторой исходной точке. Необходимо выбрать начальную базисную точку b_1 и шаг длиной h_j для каждой переменной x_j.

1. Вычислить значение функции $f(b_1)$ в базисной точке b_1.
2. Каждая переменная по очереди изменяется прибавлением длины шага. Если значение целевой функции в пробной точке не превышает значения функции в исходной точке, то шаг поиска рассматривается как успешный. Таким образом, вычисляем значение функции $f(b_1+h_1e_1)$ где e_1 - единичный вектор в направлении оси x_1. Если это приводит к уменьшению значения функции, то b_1 заменяется на $b_1+h_1e_1$. В противном случае вычисляется значение функции $f(b_1-h_1e_1)$, и если ее значение уменьшилось, то b_1 заменяем на $b_1-h_1e_1$. Если ни один из проделанных шагов не приводит к уменьшению значения функции, то точка остается неизменной и рассматривается изменение в направлении оси x_2, т.е. находится значение функции $f(b_1+h_2e_2)$ и т.д.
   После перебора всех n координат исследующий поиск завершается. Полученную в результате точку b_2 называют базисной точкой.
3. Если $b_2=b_1$, то исследование повторяется вокруг той же базисной точки b_1, но с уменьшенной длиной шага.
4. Если $b_2 \neq b_1$, то проводится поиск по образцу.

Б.Поиск по образцу. Поиск по образцу заключается в реализации единственного шага из полученной базисной точки вдоль прямой, соединяющей эту точку с предыдущей базисной точкой.

1. Новую точку образца определяем по формуле
$$p_i=b_i+2(b_{i+1}-b_i),\ [p_1=b_1+2(b_2-b_1)].$$
2. Затем исследование следует продолжить вокруг точки $p_i(p_1)$.
3. Если наименьшее значение в шаге в п.2 меньше значения в базисной точке b_2, то имеем новую базисную точку $b_3(b_{i+2})$, после чего следует повторить шаг (В п.1). В противном случае не производить поиск по образцу из точки $b_2(b_{i+1})$, а продолжить исследования в точке $b_2(b_{i+1})$.

С:Завершить процесс, когда длина шага будет уменьшена до заданного малого значения.

АЛГОРИТМ МЕТОДА ПОИСКА ХУКА-ДЖИВСА

1. Определить:
   начальную точку $x_{(0)}$; шаг (приращение) $h_i(\nabla_i)$, $i=1,2,\ldots n$, коэффициент уменьшения шага $k>1$; параметр окончания поиска $\varepsilon>0$.
2. Провести исследующий поиск.
3. Исследующий поиск удачный?
   Да: перейти к шагу 5. Нет: продолжить.
4. Проверка на окончание поиска. Выполняется ли неравенство $\|\mathbf{h}\| < \varepsilon$?
   Да: прекратить поиск; текущая точка аппроксимирует точку оптимума x^*.
   Нет: уменьшить приращение $h_i = \frac{h_i}{k}$; перейти к шагу 2.
5. Провести поиск по образцу: $x_{p(i)}=x_{(i)}+(x_{(i)}-x_{(i-1)})$.
6. Провести исследующий поиск, используя $x_{p(i+1)}$ в качестве базовой точки; пусть $x_{(i+1)}$ - полученная в результате точка.
7. Выполняется ли неравенство $f(x_{(i+1)})<f(x_{(i)})$?
   Да: положить $x_{(i-1)}=x_{(i)}$, $x_{(i)}=x_{(i+1)}$. Перейти к шагу 5.
   Нет: перейти к шагу 4.

Пример. Поиск по методу Хука-Дживса.

Найти точку минимума функции

$$f(x) = 8x_1^2 + 4x_1x_2 + 5x_2^2,$$

используя начальную точку $x_{(0)}=|-4,-4|^T$. (Индекс "Т" означает транспонирование).

Решение. Зададим следующие величины:

$\Delta x(h)$ - векторная величина приращения $=|1,1|^T$,

k - коэффициент уменьшения шага $=2$,

ε - параметр окончания поиска $=10^{-4}$.

Итерации начинаются с исследующего поиска вокруг точки $x_{(0)}$, для которой $f(x_{(0)})=272$. Фиксируя x_2, дадим приращение переменной x_1:

$x_2=-4$;

$x_1=-4+1 \rightarrow f(-3,-4)=200 < f(x_{(0)}) \rightarrow$ успех.

Следовательно, фиксируем $x_1=3$ и дадим приращение переменной x_2:

$x_1=-3$;

$x_2=-4+1 \rightarrow f(-3,-3)=153 < 200 \rightarrow$ успех.

Таким образом, в результате исследующего поиска найдена точка $x_{(1)}=|-3,-3|^T$, $f(x_{(1)})=153$.

Поскольку исследующий поиск был удачным, переходим к поиску по образцу:

$$x_{p(2)}=x_{(1)}+(x_{(1)}-x_{(0)})=|-2,-2|^T,$$
$$f(x_{p(2)})=68.$$

Далее проводится исследующий поиск вокруг точки $x_{p(2)}$, который оказывается удачным. В результате получаем точку $x_2=|-1,-1|^T$, $f(x_{(2)})=17$.

Поскольку поиск по образцу успешный, и $x_{p(2)}$ становится новой базисной точкой при следующем проведении поиска по образцу. Итерации продолжаются, пока уменьшение величины шага не укажет на окончание поиска в окрестности точки минимума $x^* = |0,0|^T$.

Описанный выше алгоритм и блок-схема реализованы в виде программы на языке Паскаль.

Обозначения некоторых параметров: **n** - размерность пространства, в котором задана функция **f**, **x** - массив, в элементах которого размещаются значения координат начальной точки $x_{(0)}$, **z** - значение целевой функции **f**. **h** - длина шага, f_e - количество вычислений функции. **p** - точка, построенная при движении по образцу, **v** - текущая базисная

точка; **b** - предыдущая базисная точка; **k** -коэффициент уменьшения шага.

При **bs=1**, p_s**=0** - исследование в базисной точке;

при **ps=0**, p_s**=1** - исследование в точке образца.

В приведенной программе минимальная длина шага равна 10^{-8}, но она может быть изменена. Программа вычисляет значение минимизируемой функции.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Построить математическую модель задачи проектирования.
2. Сделать несколько итераций (или найти минимум) предлагаемой целевой функции по алгоритму, рассмотренному выше.
3. Используя приведенную программу вычисления минимума функции, составить программу для расчета предлагаемой функции. Выбрать начальную точку и начальный шаг.
4. Провести вычисления на персональных ЭВМ.

Пример. Найти минимум функции

$$f(x) = 20 + 0.3x_1 + 4x_2 - 0.3x_1^2 + 0.3x_2^2 + 0.4x_1x_2$$

по методу Хука-Дживса. В качестве начального приближения выберем точку **(1,1).**

Программа, реализующая решение задачи, может иметь вид

БЛОК-СХЕМА МЕТОДА ХУКА-ДЖИВСА

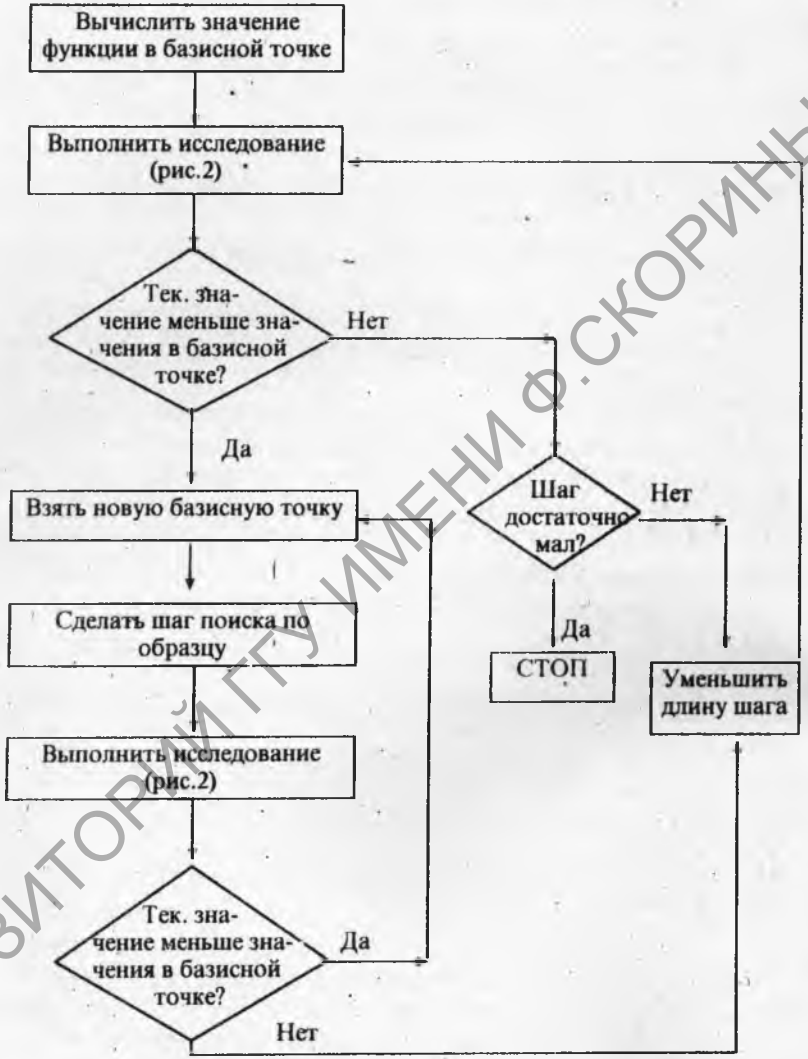


Рис.1

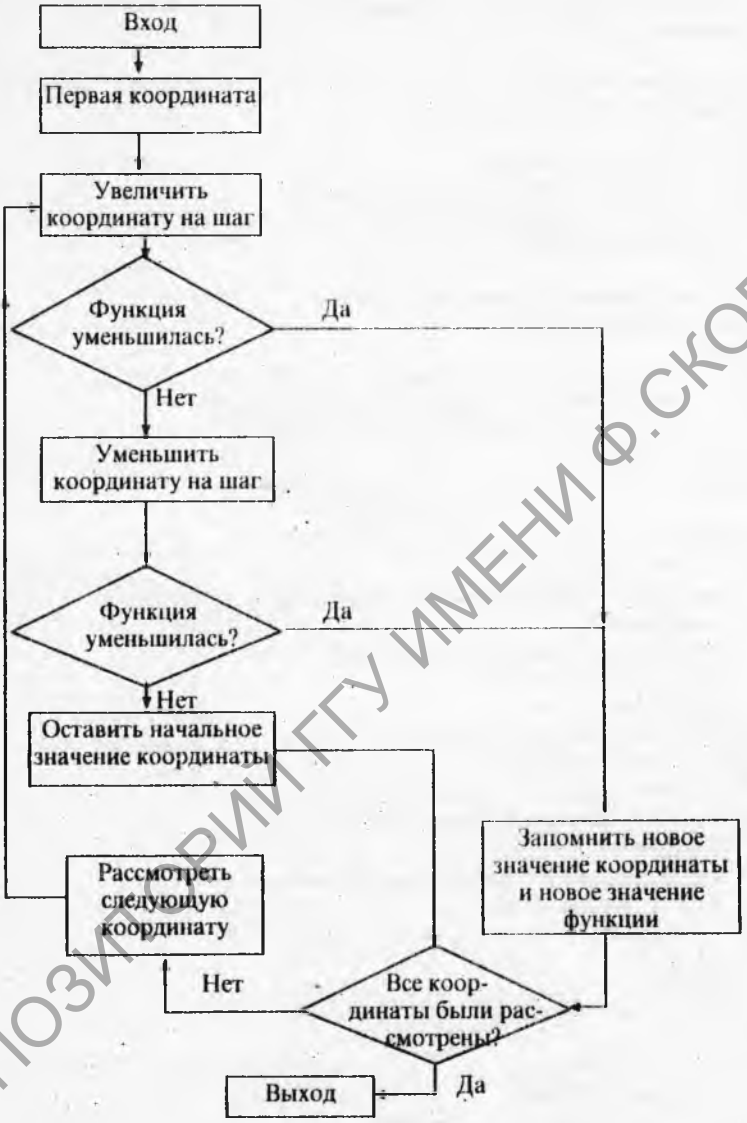


Рис.2

```
{ Метод минимизации функций без ограничений по Хуку-
Дживсу }

program khy_djv;
uses crt,printer;
var
  i,j,n,fe:integer;
  y,b,p,x:array[1..10] of real;
  z,ps,k,h,bs,fb,fi:real;
  pp,ppp:boolean;

{ Вычисление значения функции }
procedure prcd1330;
begin
  z:=0;
  z:=20+0.3*x[1]+4*x[2]+0.4*x[1]*x[2];
  z:=z+0.3*sqr(x[1])+0.3*sqr(x[2]);
  fe:=fe+1;
end;

begin
  clrscr;
  write(' Введите число переменных ⇒ ');
  readln(n);
  writeln(' Введите начальную точку ');
  for i:=1 to n do
    begin
      write('x',i,'⇒ ');
      readln(x[i]);
    end;
  write(' Введите длину шага ⇒ ');
  readln(h);
  writeln(lst,' Начальные значения ⇒ ');
  for i:=1 to n do
   writeln(lst,'x[',i,']=',x[i]:7:5);
  k:=h;
  fe:=0;
  for i:=1 to n do
    begin
      y[i]:=x[i];
      p[i]:=x[i];
      b[i]:=x[i];
    end;
  prcd1330;
```

```
fi:=z;
writeln(' Исследующий поиск z=',z:7:5);
for i:=1 to n do
  writeln('x[',i,']=',x[i]:7:5);
ps:=0;
bs:=1;
j:=1;
fb:=fi;
ppp:=true;
while ppp do
  begin
    x[j]:=y[j]+k;
    prcd1330;
    pp:=true;
    while pp do
      begin
        if z<fi then y[j]:=x[j]
          else
            begin
              x[j]:=y[j]-k;
              prcd1330;
              if z<fi then y[j]:=x[j]
                      else x[j]:=y[j];
            end;
        pp:=false;
      end;
    prcd1330;
    fi:=z;
    writeln(' Поиск по образцу z=',z:7:5);
    for i:=1 to n do
      writeln('x[',i,']=',x[i]:7:5);
    if j=n then
      if fi<fb-10E-8 then
        begin
          for i:=1 to n do
            begin
              p[i]:=2*y[i]-b[i];
              b[i]:=y[i];
              x[i]:=p[i];
              y[i]:=x[i];
            end;
          prcd1330;
          fb:=fi;
          ps:=1;
          bs:=0;
```

```
                fi:=z;
                writeln(' Замена базисной точки
z=',z:7:5);
                for i:=1 to n do
                  writeln('x[',i,']=',x[i]:7:5);
                j:=1;
              end
                else
                  if (ps=1) and (bs=0) then
                    begin
                      for i:=1 to n do
                        begin
                          p[i]:=b[i];
                          y[i]:=b[i];
                          x[i]:=b[i];
                        end;
                      prcd1330;
                      bs:=1;
                      ps:=0;
                      fi:=z;
                      fb:=z;
                      writeln(' Замена базисной точки
z=',z:7:5);
                      for i:=1 to n do
                        writeln('x[',i,']=',x[i]:7:5);
                      j:=1;
                    end
                      else
                        begin
                          k:=k/10;
                          writeln(' Уменьшить длину
шага ');
                          if k<10E-8 then
                            begin
                              writeln(lst,'Минимум найден
z=',z:7:5);
                              writeln(lst,'Точка с
координатами');
                              for i:=1 to n do
          writeln(lst,'x[',i,']=',x[i]:7:5);
                              writeln(lst,' Кол-во
вычислений функции =',fe);
                              ppp:=false;
                            end;
                          j:=1;
```

```
                    end
          else j:=j+1;
      end;
    readln;
  end.
```

Лабораторная работа 2
МЕТОД ГРАДИЕНТНОГО СПУСКА

В лабораторной работе для отыскания экстремума используется градиент целевой функции и строится последовательность векторов $\{x^{(k)}\}$, удовлетворяющих условию: $f(x^{(0)}) > f(x^{(1)}) > \ldots > f(x^{(n)})$. Методы построения таких последовательностей называются методами спуска. Элементы последовательности $\{x^{(k)}\}$ вычисляются согласно итерационной процедуре, реализуемой в соответствии с формулой

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p(x^{(k)}), \qquad (1)$$

где $x^{(k)}$ - текущее приближение к решению x^*; $\alpha^{(k)}$ - параметр, характеризующий длину шага; $p(x^{(k)}) \equiv p^{(k)}$ - направление поиска (спуска) в N - мерном пространстве переменных x_i, $i = \overline{1,N}$.

Способ определения $p(x)$ и α на каждой итерации связан с особенностями применяемого метода. Наиболее употребительны: метод наискорейшего спуска, т.е. метод Коши (Cauchy A.) и метод дробления шага.

А. Метод Коши. Выбирая антиградиент в качестве направления спуска, строим итерационный процесс

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}), \qquad (2)$$

где $\nabla f(x^{(k)}) = \text{grad } f(x)\big|_{x=x^{(k)}}$.

Величина $\alpha^{(k)}$ определяется из условия решения задачи минимизации $f(x^{(k+1)})$, вдоль направления $\nabla f(x^{(k)})$

$$f(x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)})) = \min_{\alpha \geq 0} (f(x^{(k)} - \alpha \nabla f(x^{(k)})),$$

т.е. на каждом шаге решается одномерная задача минимизации. При достаточно малой длине шага итерации выполняется неравенство

$$f(x^{(k+1)}) \leq f(x^{(k)}).$$

Б. Метод дробления шага. Процесс (2) с дроблением шага протекает следующим образом. Выбираем некоторое начальное значение $x^{(0)}$ из области расположения искомой точки минимума. Затем выбираем некоторое $\alpha^{(k)}=\alpha=\text{const}$ и на каждом шаге процесса (2) проверяем условие монотонности $f(x^{(k+1)}) < f(x^{(k)})$. Если это условие нарушается, то α дробим до тех пор, пока монотонность не восстановится. Для окончания счета можно использовать, например, условие $\|\nabla f(x^{(k+1)})\| < \varepsilon$, считая, что

$$x_{min} = x^{(k+1)},\ \|\nabla f(x^{(k+1)})\| = \sqrt{\sum \left(\partial f/\partial x_j\right)^2}\ .$$

Пример. Поиск по методу Коши (п.А)

Рассмотрим функцию

$$f(x) = 8x_1^2 + 4x_1x_2 + 5x_2^2$$

и по методу Коши найдем минимум функции или сделаем несколько итераций.

Решение.

1. Вычислим компоненты градиента

$$\frac{\partial f}{\partial x_1} = 16x_1 + 4x_2,\quad \frac{\partial f}{\partial x_2} = 10x_2 + 4x_1.$$

2. Зададим начальное приближение $x^{(0)}=[10,10]^T$, проверяем выполнение условия $\nabla f(x^{(0)}) \neq 0$ и затем с помощью формулы (2) построим новое приближение $x^{(1)}=x^{(0)}+\alpha^{(0)}\nabla f(x^{(0)})$ или

$$x_1^{(1)} = 10 - \alpha^{(0)}200,\ x_2^{(1)} = 10 - \alpha^{(0)}140.$$

3. Составляем задачу одномерной минимизации по $\alpha^{(0)}$, подставив значения $x_1^{(1)}$ и $x_2^{(1)}$ в целевую функцию $f(x)$. Находим стационарную точку из условия $f(\alpha^{(0)}) \rightarrow \min$, $\alpha^0=0.056$ (например, по условию $\frac{\partial f(\alpha^{(0)})}{\partial \alpha} = 0$).

4. Далее находим точку $x^{(2)}=x^{(1)}-\alpha^{(1)}\nabla f(x^{(1)})$ (повторяем все действия аналогично проведенным для точки $x^{(1)}$), проводя одномерный поиск (поиск вдоль прямой).

5. Работа алгоритма завершается, когда модуль градиента или модуль вектора Δx становится достаточно малым.

Блок-схема алгоритма приведена на рис.3

В. Программа расчета по методу дробления шага. Описанный алгоритм **п.Б** реализован в виде программы **gradsp**.

Входные параметры: **n** - размерность пространства, в котором задана функция **f**, **x** - массив, в элементах которого размещаются значения координат начальной точки $x^{(0)}$, **y** - значение целевой функции **f**; **p** - массив, содержащий компоненты градиента целевой функции; **al** - начальное значение параметра α; **eps** - значение величины ε, входящей в критерий окончания итерационного процесса. **F** - процедура вычисления функции **y=f(x)**, **F1** - процедура вычисления компонентов вектора **p=grad f($x^{(k)}$)**.

Выходные параметры: **X** - массив, содержащий координаты точки минимума x_{min}; **k** - число совершенных итераций; **y** - значение $f(x_{min})$.

Задание. Используя программу **gradsp**, минимизировать заданную целевую функцию.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Построить математическую модель задачи проектирования.
2. Сделать несколько итераций (или найти минимум) предлагаемой целевой функции по методу Коши (п.А).
3. Используя приведенную программу составить головную программу, содержащую обращение к **gradsp** и печать результатов.
4. Провести вычисления на ПЭВМ.

Пример. Найти минимум функции

$$f(x_1, x_2) = ax_1 + bx_2 + e^{cx_1^2 + dx_2^2},$$

где **a=1, b=2, c=3, d=4.**

В качестве начального приближения выберем $x^{(0)}=|0,0|^T$ и $\varepsilon=10^{-6}$.

Программа, реализующая решение задачи, может иметь вид

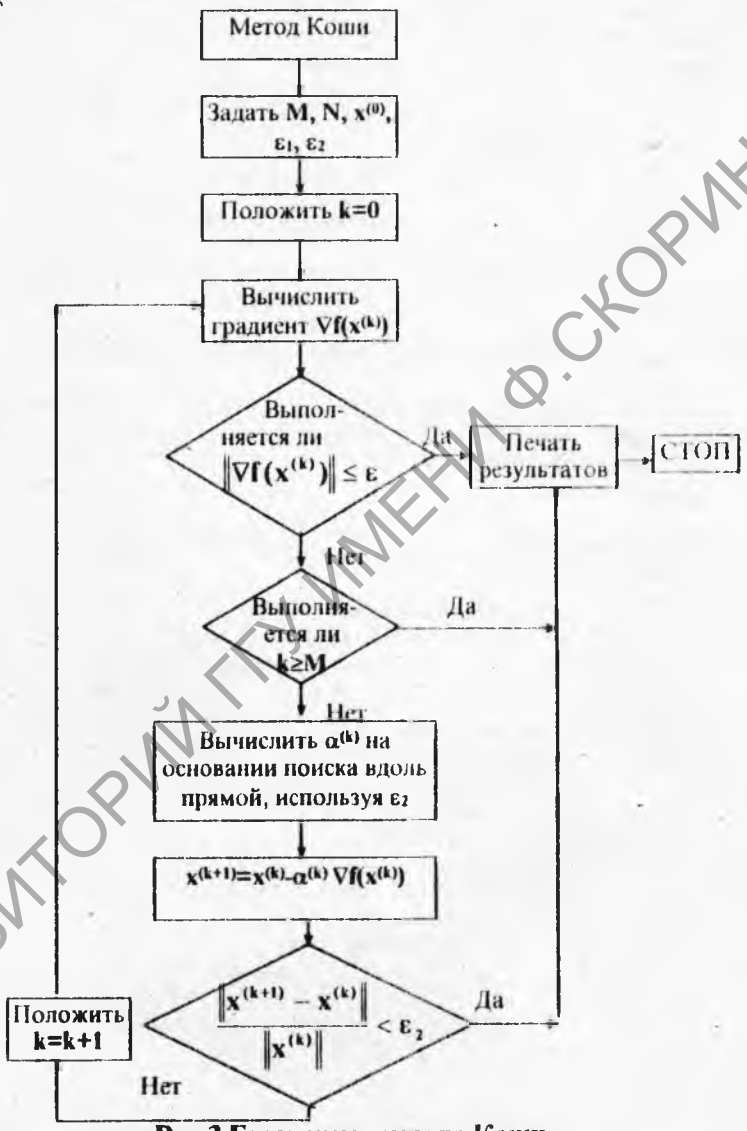


Рис.3 Блок-схема метода Коши

{ МЕТОД ГРАДИЕНТНОГО СПУСКА }

```
Program gradsp;
uses crt,printer;
Type  mas=array[1..2] of real;
Var
  yf,a,a1,eps,b,c,d,y,s,y1 : real;
  n,i,k : integer;
  x,p,yy,x1 : mas;

Procedure F(var x : mas; var y : real);
  begin
    y:=a*x[1]+b*x[2]+exp(c*x[1]*x[1]+d*x[2]*x[2]);
  end;

Procedure F1(var x,yy : mas; var y : real);
  begin
    yf:=exp(c*x[1]*x[1]+d*x[2]*x[2]);
    yy[1]:=a+2*c*x[1]*yf;
    yy[2]:=b+2*d*x[2]*yf;
  end;

Procedure GRAD;
 label 1,2;
  begin
     k:=0;
     F(x,y);
 1 : F1(x,p,y);
     s:=0;
     for i:=1 to n do
       s:=s+p[i]*p[i];
     if s>eps then
              begin
                k:=k+1;
              2: for i:=1 to n do
                 x1[i]:=x[i]-a1*p[i];
                 F(x1,y1);
                 if y1<=y then
                              begin
                                for i:=1 to n do
                                  x[i]:=x1[i];
                                y:=y1;
                                goto 1;
                              end
```

```
                                    else
                                     begin
                                      a1:=a1*0.5;
                                      goto 2;
                                     end;
                        end;
    end;

BEGIN
    ClrScr;
    n:=2;
    eps:=0.000001;
    x[1]:=0;
    x[2]:=0;
    a1:=0.15;
    writeln('Введите коэффициенты функции');
    write('a= ');
    readln(a);
    write('b= ');
    readln(b);
    write('c= ');
    readln(c);
    write('d= ');
    readln(d);
    GRAD;
    clrscr;
    write(lst,'a=',a:7:3,';   b=',b:7:3,';
c=',c:7:3,';   d=',d:7:3);
    writeln(lst);
    for i:=1 to n do
      writeln(lst,'x[',i,']= ',x[i]:7:3);
    writeln(lst,'f(x1,x2)= ',y:7:4);
    writeln(lst,'количество итераций ',k);
    readln;
END.
```

Лабораторная работа 3
МИНИМИЗАЦИЯ ФУНКЦИИ БЕЗ ОГРАНИЧЕНИЙ МЕТОДОМ СОПРЯЖЕННЫХ ГРАДИЕНТОВ - МЕТОДОМ ФЛЕТЧЕРА-РИВСА

Метод сопряженных градиентов обладает прекрасным свойством: положительно определенная квадратичная форма **n** переменных минимизируется методом сопряженных градиентов не более чем за **n** шагов. Для достижения точного минимума неквадратичных функций потребуется бесконечное число шагов. Но метод успешно применяется, так как в окрестности экстремума любая гладкая функция мало отличается от квадратичной.

Опишем алгоритм Флетчера-Ривса (Fletcher R.-Reeves C.M.).

1. Задав начальную точку $x^{(0)}$ и выбрав $\varepsilon>0$ определяем градиент в точке $x^{(0)}$; $d_0=-\nabla f(x_{(0)})$, если $\|\nabla f(x_0)\| \le \varepsilon$, то остановка.
2. На **k**-м шаге с помощью одномерного поиска решаем задачу минимизации функции $f(x_{(k)}+\lambda_k d_k)$ по λ_k, находим величину шага λ_k и точку $x_{(k+1)}=x_{(k)}+\lambda_k d_k$, если $k<n$, то переходим к шагу 3.
3. Вычисляем $f(x_{(k+1)})$ и градиент $\nabla f(x_{(k+1)})$.
4. Направление d_{k+1} находим по зависимости $d_{k+1}=-\nabla f(x_{(k+1)})+d_k\alpha_k$, где $\alpha_k = \dfrac{\Delta^T f(x_{(k+1)})\nabla f(x_{(k+1)})}{\Delta^T f(x_{(k)})\nabla f(x_{(k)})}$, T - обозначает транспонирование. После $n+1$-й итерации ($k=n$) процедура циклически повторяется с заменой $x_{(0)}$ на $x_{(n+1)}$.
5. Алгоритм заканчивается при $\|d_k\| < \varepsilon$.

Пример. Найти точку минимума функции

$$f(x) = 4x_1^2 + 3x_2^2 - 4x_1x_2 + x_1$$

методом Флетчера-Ривса, если $x^{(0)}=[0,0]^T$.

Решение.

1. $\nabla f(x)=|8x_1-4x_2+1,\ 6x_2-4x_1|^T$;
   $d_0=-f(x_{(0)})=-|1,0|^T$.
2. Поиск вдоль прямой, минимизация по λ_0:
   $x_{(1)} = x_{(0)} - \lambda_0 \nabla f(x_{(0)}) \rightarrow \lambda = \frac{1}{8}$,
   $x_{(1)}=|0,0|^T-\frac{1}{8}\ |1,0|^T=|-\frac{1}{8},0|$.
3. $k=1$
   $$\alpha_0 = \frac{|0,\frac{1}{2}||0,\frac{1}{2}|^T}{|1,0||1,0|^T} = \frac{1}{4};$$
   $d_1=-|0,\ \frac{1}{2}|^T-\frac{1}{4}|1,0|^T=|-\frac{1}{4},-\frac{1}{2}|^T$.
4. Поиск вдоль прямой:
   $x_{(2)}=x_{(1)}+\lambda_1 d_1 \rightarrow \lambda_1=\frac{1}{4}$;
   $x_{(2)}=|-\frac{1}{4},0|^T-\frac{1}{4}\ |\frac{1}{4},\frac{1}{2}|^T=|\frac{3}{16},\frac{1}{8}|^T$.
5. $\nabla f(x_{(2)})=|0,0|^T$.

Приведем блок-схему **(Рис.4)** и текст программы, реализующие идею метода сопряженных градиентов. Введем переобозначение градиента $\nabla f(x_{(i)})=g(x_{(i)})=g_i$ и направление наискорейшего спуска $d_i=-g_i$.

Описанный алгоритм реализован в виде программы **flrv.**

Входные параметры: **n** - размерность пространства, в котором задана функция **f**; процедуры вычислений значений функций и градиента.

Выходные параметры: **X** - массив из **n** чисел, содержащий x_{min}; **k** -количество итераций и значение f_{min}.

Задание. Используя программу **flrv** минимизировать заданную целевую функцию.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Построить математическую модель задачи проектирования.

2. Сделать несколько итераций (или найти минимум) предложенной целевой функции.
3. Используя приведенную программу составить программу вычисления функции.
4. Провести вычисления на ПЭВМ.

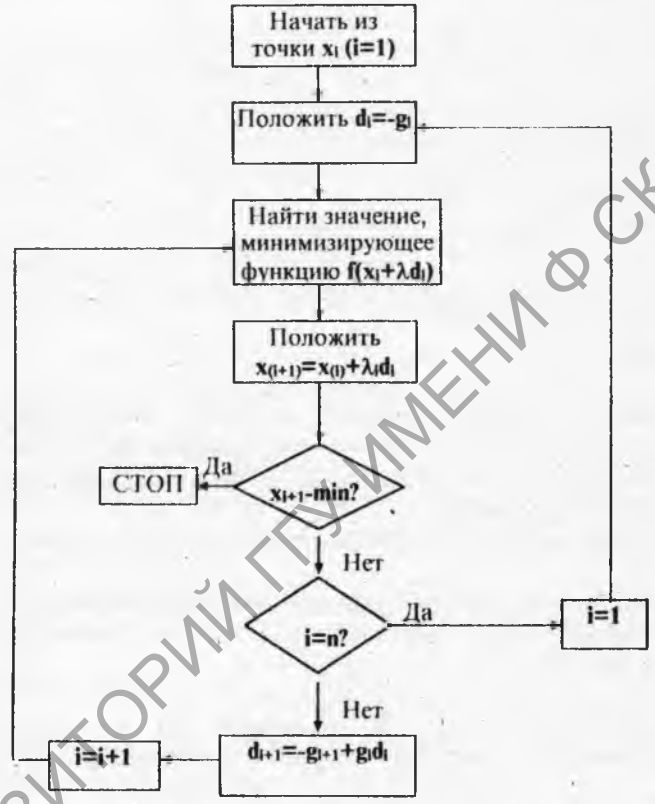


Рис.4

Пример. Найти минимум функции

$$f(x) = 20 + 0.3x_1 + 4x_2 + 0.3x_1^2 + 0.3x_2^2 + 0.4x_1x_2.$$

В качестве начального приближения выберем

$x_{1(0)}=0.25$; $x_{2(0)}=2.5$; $\varepsilon=0.1*10^{-6}$.

Программа, реализующая решение задачи, может иметь вид

```
{ Минимизация методом Флетчера-Ривса }

program flrv;
uses crt,printer;
var
  i,k,n,sv,tv,dv:integer;

ak,fp,z,g1,g0,gp,qx,hh,bb,fq,g2,gq,zz,ww,w,dd,fr,g3,gr
,gk:real;
  x,pr,d,g,q:array[1..10] of real;
  p,pp,ppp:boolean;

{ Вычисление значения функции }
procedure prcd1330;
begin
  z:=0;
  z:=20+0.3*x[1]+4*x[2]+0.4*x[1]*x[2];
  z:=z+0.3*sqr(x[1])+0.3*sqr(x[2]);
  tv:=tv+1;
end;

{ Вычисление градиента }
procedure prcd1380;
begin
  g0:=0;
  g[1]:=0.3+0.6*x[1]+0.4*x[2];
  g[2]:=4+0.6*x[2]+0.4*x[1];
  for i:=1 to n do
    g0:=g0+g[i]*g[i];
  g0:=sqr(g0);
end;

begin
    clrscr;
    write('Введите число переменных ==> ');
    readln(n);
    writeln('Введите начальную точку');
    for i:=1 to n do
      begin
        write('x',i,'==> ');
```

```
        readln(x[i]);
      end;
    sv:=1;
    tv:=0;
    dv:=0;
    { Промежуточный вывод }
    writeln(lst,'Начальные значения');
    for i:=1 to n do
      writeln(lst,'x[',i,']=',x[i]:6:3);
    writeln('Текущие значения');
    pp:=true;
    while pp do
      begin
        for i:=1 to n do
          begin
            pr[i]:=x[i];
            writeln('x',i,'=',x[i]:6:3);
          end;
        prcd1330;
        writeln('Итерация ',sv,' значение функции
=',z:7:4);
        fp:=z;
        prcd1380;
        g1:=g0;
        gk:=g0;
        { В качестве первого направления взять }
        {   направление наискорейшего спуска   }
        for i:=1 to n do
          d[i]:=-g[i];
        { Счетчик итераций }
        k:=1;
        ppp:=true;
        while ppp do
          begin
            p:=true;
            while p do
              begin
                gp:=0;
                for i:=1 to n do
                  gp:=gp+g[i]*d[i];
                {Определить начальный шаг и,если
необходимо, }
                {    изменить направление на
противоположное  }
                if gp>0 then
```

```
            begin
              qx:=abs(2*fp/gp);
              if qx>1 then qx:=1;
              for i:=1 to n do
                begin
                  x[i]:=pr[i]-qx*d[i];
                  pr[i]:=x[i];
                end;
              prcd1330;
              fp:=z;
              writeln('Нестабильность ?');
              prcd1380;
              g1:=g0;
            end
               else p:=false;
  end;
qx:=abs(2*fp/gp);
if qx>1 then qx:=1;
hh:=qx;
{ Найти след. точку }
p:=true;
while p do
  begin
    bb:=hh;
    for i:=1 to n do
      begin
        q[i]:=pr[i]+bb*d[i];
        x[i]:=q[i];
      end;
    prcd1330;
    fq:=z;
    prcd1380;
    g2:=g0;
    gq:=0;
    for i:=1 to n do
      gq:=gq+g[i]*d[i];
    {Выполнить интерполяцию,удвоить шаг и
перейти}
    {к точке Q,или удвоить шаг,чтобы      }
    {          'накрыть' минимум          }
    if (gq>10E-8) or (fq>fp) then p:=false
      else
        begin
          hh:=2*hh;
          for i:=1 to n do
```

```
            pr[i]:=q[i];
          fp:=fq;
          gp:=gq;
          g1:=g2;
        end;
  end;
p:=true;
while p do
  begin
    zz:=3*(fp-fq)/hh;
    zz:=zz+gp+gq;
    ww:=zz*zz-gp*gq;
    if ww<0 then ww:=0;
    w:=sqr(ww);
    dd:=hh*(1-(gq+w-zz)/(gq-gp+2*w));
    for i:=1 to n do
      x[i]:=pr[i]+dd*d[i];
    prcd1330;
    fr:=z;
    prcd1380;
    g3:=g0;
    { Найти градиент в новой точке }
    gr:=0;
    for i:=1 to n do
      gr:=gr+g[i]*d[i];
    if (z<=fp) and (z<=fq) then p:=false;
    if p then
      if gr>0 then
        begin
          hh:=dd;
          for i:=1 to n do
            q[i]:=x[i];
          fq:=z;
          gq:=gr;
        end
            else
        begin
          hh:=hh-dd;
          for i:=1 to n do
            pr[i]:=x[i];
          fp:=z;
          gp:=gr;
        end;
  end;
{ Проверка критерия завершения }
```

```
            if g3<10E-8 then
              begin
                writeln(lst,'Минимум найден');
                writeln(lst,'Кол-во итераций =',sv,'
значение минимума =',z:7:4);
                writeln(lst,'Точка с координатами');
                for i:=1 to n do
                  writeln(lst,'x',i,'=',x[i]:6:3);
                readln;
                halt;
              end;
            if k<>n then
              begin
                { Увеличить счетчик }
                k:=k+1;
                { Найти сопр. направление }
                ak:=g3*g3/(gk*gk);
                for i:=1 to n do
                  begin
                    d[i]:=-g[i]+ak*d[i];
                    pr[i]:=x[i];
                  end;
                writeln('Новое направление ',dv+1,'
поиска',dv);
                fp:=z;
                g1:=g0;
                gk:=g0;
                for i:=1 to n do
                  writeln('x',i,'=',x[i]:7:4);
                writeln('z=',z:7:4);
              end
                    else
              begin
                writeln('Рестарт');
                sv:=sv+1;
                dv:=dv+1;
                writeln('Итерация ',sv,' поиск ',dv);
                ppp:=false;
              end;
          end;
      end;
end.
```

Лабораторная работа 4
МИНИМИЗАЦИЯ ФУНКЦИИ БЕЗ ОГРАНИЧЕНИЙ МЕТОДОМ ДЭВИДОНА-ФЛЕТЧЕРА-ПАУЭЛЛА (ДФП)

Метод Дэвидона-Флетчера-Пауэлла (Davidon V. - Fletcher R. - Powell I.) основан на использовании итерационного соотношения

$$x_{(i+1)} = x_{(i)} - G^{-1}(x_{(i)})\nabla f(x_{(i)})$$

или

$$x_{(i+1)} = x_{(i)} - G^{-1}(x_{(i)})g(x_{(i)}), \quad (1)$$

или в более удобном виде

$$x_{(i+1)} = x_{(i)} - \lambda_i G^{-1}(x_{(i)})g(x_{(i)}),$$

где **G** - положительно полуопределенная матрица Гессе

$$\nabla^2 f(x_1) = G(x_1) = \begin{bmatrix} \frac{\partial^2 f(x_1)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(x_1)}{\partial x_1 \partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 f(x_1)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x_1)}{\partial x_n^2} \end{bmatrix}$$

λ_i - длина шага определяется одномерным поиском в направлении $G^{-1}(x_i)g(x_i)$;
$G^{-1}(x_i)$ - обратная матрица.

Интересно отметить, что по сравнению с простым методом наискорейшего спуска, направлением спуска в данном случае будет не $g(x_i)$, а $G^{-1}(x_i)g(x_i)$, где учитываются и вторые производные. Методом ДФП можно получить наилучший результат, производя поиск на i-м этапе в направлении поиска $-H_i g(x_i)$, где H_i - положительно определенная симметричная матрица, которая в конечном счете становится равной $G^{-1}(x^*)$ и определяется рекуррентной зависимостью:

$$H_k = H_{k-1} + H_k^c.$$

Итерационную процедуру вычисления можно представить в виде:

1. На i-м шаге имеется точка x_i и матрица H_i.

2. Берем направление поиска $d_i=-H_ig_i$.
3. Находим λ_i из условия минимума вдоль прямой $x_i+\lambda d_i$.
4. Вводим обозначение: $v_i=\lambda_i d_i$.
5. Переходим к итерации $x_{(i+1)}=x_{(i)}+v_i$.
6. Находим значение функции $f(x_{(i+1)})$ и $g(x_{(i+1)})$, если величина $|g_i|$ и $|v_i|$ достаточно малы, то процедура завершается. В противном случае продолжить.
7. Положить $u_i=g_{i+1}-g_i$.
8. Обновить матрицу H следующим образом

$$H_{i+1} = H_i + H_i^c = H_i + A_i + B_i,$$

где

$$A_i = V_i V_i^T / (V_i^T U_i), \quad B = -\frac{H_i u_i u_i^T H_i}{u_i^T H_i u_i}.$$

В качестве H_0 выбираем единичную матрицу $H_0=J$.

9. Увеличить i на единицу $i=i+1$ и возвратиться к п.2.

Пример. Найти методом ДФП минимум функции

$$f(x) = 4x_1^2 + 3x_2 - 4x_1x_2 + x_1,$$

выбрав за начальную точку $x_{(0)}=|0,0|^T$.

Решение.

1. Находим $d_0=-\nabla f(x_{(0)})=-|1,0|$.
2. Поиск вдоль прямой (одномерный поиск см. лабораторную работу 2) приводит к точке $x_1=|-\frac{1}{8},0|^T$.

3. $i=1$, $H_1=H_0+H_0^c$, $H_0 = J = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$,

$$H_0^c = A_0 + B_0, \quad A_0 = \frac{V_0 V_0^T}{V_0^T u_0}, \quad B_0 = \frac{H_0 u_0 u_0^T H_0}{u_0^T H_0 u_0},$$

$$V_0 = x_{(1)} - x_{(0)} = |\tfrac{1}{8},0|^T - |0,0| = |-\tfrac{1}{8},0|;$$

$$u_0 = g_1 - g_0 = \nabla f(x_{(1)}) - \nabla f(x_{(0)}) = |0, \tfrac{1}{2}|^T - |1,0|^T = |-1, \tfrac{1}{2}|^T;$$

$$\mathbf{H}_0^c = \frac{[-1/8, 0]^T[-1/8, 0]}{[-1/8, 0][-1, 1/2]} - \frac{\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}[-1, 1/2]^T[-1, 1/2]\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}}{[-1, 1/2]\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}[-1, 1/2]^T};$$

$$\mathbf{H}_0^c = \begin{pmatrix}1/8 & 0\\ 0 & 0\end{pmatrix} - \begin{pmatrix}1/5 & -2/5\\ -2/5 & 1/5\end{pmatrix} = \begin{pmatrix}-27/40 & 2/5\\ 2/5 & -1/5\end{pmatrix};$$

$$\mathbf{H}_1 = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} + \begin{pmatrix}-27/40 & 2/5\\ 2/5 & -1/5\end{pmatrix} = \begin{pmatrix}13/40 & 2/5\\ 2/5 & 4/5\end{pmatrix};$$

Найдем $\mathbf{d}_1 = -\begin{pmatrix}0.325 & 2/5\\ 2/5 & 4/5\end{pmatrix}\begin{pmatrix}0\\ 1/2\end{pmatrix} = -(1/5 \quad 2/5)^T$.

4. Поиск вдоль прямой $\mathbf{x}_{(2)} = \mathbf{x}_{(1)} + \lambda_1\mathbf{d}_1 \rightarrow \lambda_1 = 5/16$.

Затем процесс продолжается и в пределе матрица $\mathbf{G}^{-1}$ стремится к матрице **H**.

Действительно, при **i=2** после вычислений имеем

$$\mathbf{H}_2 = \begin{pmatrix}0.325 & 0.4\\ 0.4 & 0.8\end{pmatrix} + \begin{pmatrix}1/16 & 1/8\\ 1/8 & 1/4\end{pmatrix}\begin{pmatrix}0.2 & 0.4\\ 0.4 & 0.8\end{pmatrix} = \begin{pmatrix}3/16 & 1/8\\ 1/8 & 1/4\end{pmatrix}.$$

Если найдем гессиан, то имеем

$$\mathbf{G} = \begin{pmatrix}8 & -4\\ -4 & 6\end{pmatrix} \longrightarrow \mathbf{G}^{-1} = \begin{pmatrix}3/16 & 1/8\\ 1/8 & 1/4\end{pmatrix}.$$

Таким образом, $\mathbf{H}_2 = \mathbf{G}^{-1}(\mathbf{x}^*)$.

Приведем на **рис.5** блок-схему и текст программы, реализующие идею метода ДФП.

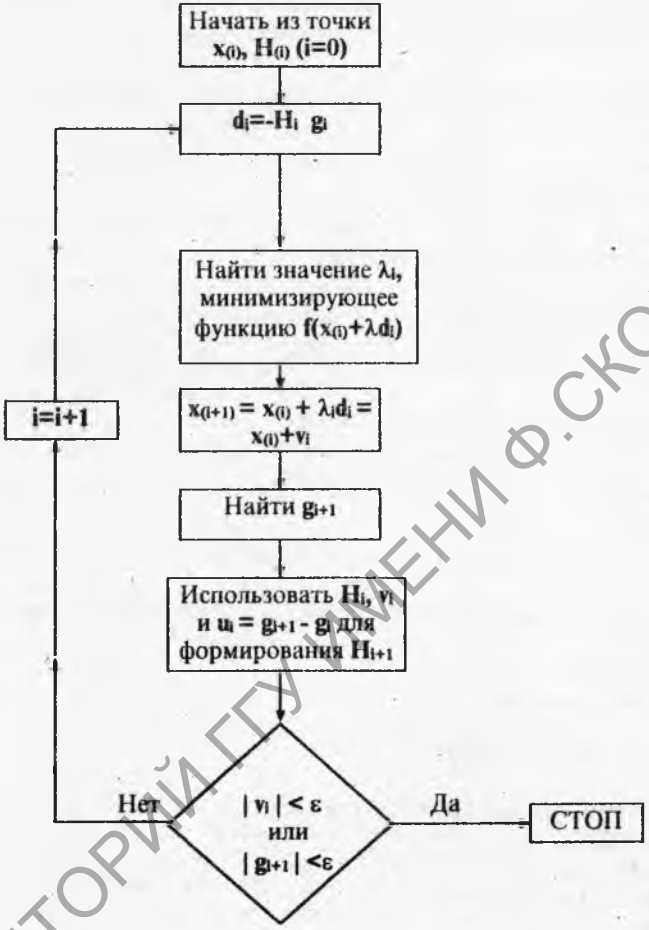


Рис. 4

Описанный алгоритм реализован в виде программы dfp.

Входные параметры: n - размерности пространства, в котором задана функция f; процедуры вычисления значений функций и градиента.

Выходные параметры: **x** - массив из **n** чисел, содержащий x_{min}, **k** - количество итераций; значение f_{min}.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Построить математическую модель задачи проектирования.
2. Сделать несколько итераций (или найти минимум) полученной целевой функции.
3. Используя приведенную программу составить программу вычисления функции.
4. Провести вычисления на ПЭВМ.

Пример. Найти минимум функции

$$f(x) = 20 + 0.3x_1 + 4x_2 + 0.3x_1^2 + 0.3x_2^2 + 0.4x_1x_2.$$

В качестве начального приближения выберем $x_{1(0)}$=0.25; $x_{2(0)}$=2.5.

Программа, реализующая решение задачи, может иметь вид

```
{Минимизация методом ДПФ Дэвидона-Флэтчера-Пауэлла) }

program dfp;
uses crt,printer;
var
  i,j,n,cc,tt:integer;

fp,z,g1,g0,gp,qx,hh,bb,fq,g2,gq,zz,ww,w,dd,fr,g3,gr,kk
,dk,wk:real;
  h:array[1..10,1..10] of real;
  x,pr,y,u,d,g,q,v,m:array[1..10] of real;
  p,pp,ppp:boolean;

{ Вычисление значения функции }
procedure prcd1330;
begin
  z:=0;
  z:=20+0.3*x[1]+4*x[2]+0.4*x[1]*x[2];
  z:=z+0.3*x[1]*x[1]+0.3*x[2]*x[2];
  tt:=tt+1;
end;

{ Вычисление значения градиента }
```

```
procedure prcd1380;
begin
  g0:=0;
  g[1]:=0.3+0.6*x[1]+0.4*x[2];
  g[2]:=4+0.6*x[2]+0.4*x[1];
  for i:=1 to n do
    g0:=g0+g[i]*g[i];
  g0:=sqr(g0);
end;

begin
  clrscr;
  write('Введите число переменных ==> ');
  readln(n);
  { Задаем H единичной матрицей }
  cc:=0;
  for i:=1 to n do
    begin
      for j:=1 to n do
        h[i,j]:=0;
      h[i,i]:=1;
    end;
  tt:=0;
  writeln('Введите начальную точку');
  for i:=1 to n do
    begin
      write('x',i,'==> ');
      readln(x[i]);
    end;
  clrscr;
  { Промежуточный вывод }
  writeln(lst,'Начальные значения');
  for i:=1 to n do
   writeln(lst,'x[',i,']=',x[i]:6:3);
  writeln('Текущие значения');
  pp:=true;
  while pp do
    begin
      for i:=1 to n do
        begin
          pr[i]:=x[i];
          y[i]:=x[i];
          writeln('x',i,'=',x[i]:6:3);
        end;
      prcd1330;
```

```
      writeln('Итерация ',cc,' значение функции
=',z:7:4);
      fp:=z;
      prcd1380;
      g1:=g0;
      { Градиент запомнить в U и выбрать нач.
направление D }
      for i:=1 to n do
        begin
          u[i]:=g[i];
          d[i]:=0;
          for j:=1 to n do
            d[i]:=d[i]-h[i,j]*g[j];
        end;
      p:=true;
      while p do
        begin
          gp:=0;
          for i:=1 to n do
            gp:=gp+g[i]*d[i];
          { Найти нач. шаг и,если необходимо,изменить
}
          {   направление спуска на противоположное
}
          if gp>=0 then
            begin
              qx:=abs(2*fp/gp);
              if qx>1 then qx:=1;
              for i:=1 to n do
                begin
                  x[i]:=pr[i]-qx*d[i];
                  pr[i]:=x[i];
                end;
              prcd1330;
              fp:=z;
              writeln('Нестабильность ?');
              clrscr;
              prcd1380;
              g1:=g0;
            end
                  else p:=false;
        end;
      qx:=abs(2*fp/gp);
      if qx>1 then qx:=1;
      hh:=qx;
```

```
        { Найти след. точку Q }
        p:=true;
        while p do
          begin
            bb:=hh;
            for i:=1 to n do
              begin
                q[i]:=pr[i]+bb*d[i];
                x[i]:=q[i];
              end;
            prcd1330;
            fq:=z;
            prcd1380;
            g2:=g0;
            gq:=0;
            for i:=1 to n do
              gq:=gq+g[i]*d[i];
            { Выполнить кубич. интерполяцию или удвоить
}
            {       шаг, чтобы 'накрыть' минимум
}
            if (gq<=0) or (fq<=fp) then hh:=2*hh
                                   else p:=false;
          end;
        p:=true;ppp:=true;
        while p do
          begin
            zz:=3*(fp-fq)/hh;
            zz:=zz+gp+gq;
            ww:=zz*zz-gp*gq;
            if ww<0 then ww:=0;
            w:=sqr(ww);
            dd:=hh*(1-(gq+w-zz)/(gq-gp+2*w));
            for i:=1 to n do
              x[i]:=pr[i]+dd*d[i];
            prcd1330;
            fr:=z;
            prcd1380;
            g3:=g0;
            { Найти градиент в новой точке }
            gr:=0;
            for i:=1 to n do
              gr:=gr+g[i]*d[i];
            if (z>fp) and (z>fq) then
               if gr<=0 then
```

```
          begin
            hh:=hh-dd;
            for i:=1 to n do
              pr[i]:=x[i];
            fp:=z;
            gp:=gr;
            g1:=g0;
          end
                else p:=false
      else
        begin
          p:=false;
          ppp:=p;
        end;
  end;
while ppp do
  begin
    hh:=dd;
    for i:=1 to n do
      q[i]:=x[i];
  end;
{ Обновить матрицу H }
kk:=0;
wk:=0;
dk:=0;
for i:=1 to n do
  begin
    u[i]:=g[i]-u[i];
    v[i]:=x[i]-y[i];
  end;
for i:=1 to n do
  begin
    m[i]:=0;
    for j:=1 to n do
      m[i]:=m[i]+h[i,j]*u[j];
    kk:=kk+m[i]*u[i];
    wk:=wk+v[i]*u[i];
    dk:=dk+v[i]*v[i];
  end;
if (kk<>0) or (wk<>0) then
  for i:=1 to n do
    for j:=1 to n do
      h[i,j]:=h[i,j]-m[i]*m[j]/kk+v[i]*v[j]/wk;
cc:=cc+1;
{ Проверка критерия завершения }
```

```
      if (sqrt(dk)<0.0000001) or (g3<0.0000001) then
pp:=false;
    { Начать новую итерацию поиска }
    end;
  writeln(lst,'Минимум найден');
  writeln(lst,'Кол-во итераций =',cc,' значение
минимума=' , z:7:4);
  writeln(lst,'Точка с координатами');
  for i:=1 to n do
    writeln(lst,'x',i,'=',x[i]:6:3);
  readln;
end.
```

Лабораторная работа 5
МЕТОД МИНИМИЗАЦИИ ФУНКЦИИ С ОГРАНИЧЕНИЯМИ С ИСПОЛЬЗОВАНИЕМ ШТРАФНЫХ ФУНКЦИЙ

Суть метода штрафных функций составляет преобразование задачи минимизации функции $z=f(x)$ с ограничениями, наложенными на x, в задачу поиска минимума без ограничений функции $z=f(x)+p(x)$, где $p(x)$ - штрафная функция, $x \in R^n$.

Задача минимизации имеет вид

Найти минимум функции $z=f(x)$, $x \in R^n$ при ограничениях $C_j(x)>0$, $j=\overline{1,m}$. Приняв функцию $p(x)=r\sum_{j=1}^{m}\frac{1}{c_j(x)}$, где $r \geq 0$, запишем

$$z=\varphi(x,r)=f(x)+r\sum_{j=1}^{m}\frac{1}{c_j(x)}. \qquad (1)$$

Имеются различные виды представления штрафов (квадратичный штраф, логарифмический и т.д.), но в этой работе будем рассматривать штраф, задаваемый обратной функцией (1).

Таким образом, решаем задачу минимизации (1) с ограничениями $C_j(x) \geq 0$ для последовательности значений r, стремящейся к нулю, исследуем стационарные точки функции $\varphi(x,r)$ при различных значениях r, которые сходятся к x_{min} при $r \to 0$.

В качестве метода, реализующего минимизацию функции, используем метод SUMT Фиакко и Маккормик **(Sequential unconstrained minimisation technique).**

Алгоритм метода штрафных функций состоит в следующем:

1. Задать значения: ε_1, ε_2, ε_3, x_0, r_0, где ε_1 - параметр окончания одномерного поиска, ε_2 - параметр окончания процедуры безусловной минимизации; ε_3 - параметр окончания работы

алгоритма; r_0 - начальный вектор штрафных функций; x_0 - начальное приближение.

2. Построить штрафную функцию $\varphi(x,r)=f(x)+\phi(r,g(x))$ для задачи $f(x)\to\min$, $g_j(x)\ge 0$, $j=\overline{1,J}$.
3. Найти точку $x_{(k+1)}$, которая доставляет минимум функции $\varphi(x_{(k+1)})$ и найти r_k. В качестве начальной точки использовать $x_{(k)}$, а в качестве параметра окончания ε_2.
4. Проверить выполнение условия $|F(x_{(k+1)}, r_{k+1}) - F(x_{(k)}, r_{k-1}|\le\varepsilon_3$. В случае, если оно выполняется, положить $x_{(k+1)}=x'$ и закончить процесс. В противном случае п.5.
5. Положить $z_{k+1}=r_k+\Delta r_k$ и перейти к п.2.

Пример. Рассмотрим следующую задачу: минимизировать функцию

$$f(x_1, x_2) = 3x_1^2 + 4x_1x_2 + 5x_2^2$$

при ограничениях $x_1\ge 0$, $x_2\ge 0$, $x_1+x_2\ge 4$.

Решение. По аналогии с уравнением (1) составим функцию

$$\varphi(x, r) = 3x_1^2 + 4x_1x_2 + 5x_2^2 + r\left(\frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{x_1 + x_2 - 4}\right).$$

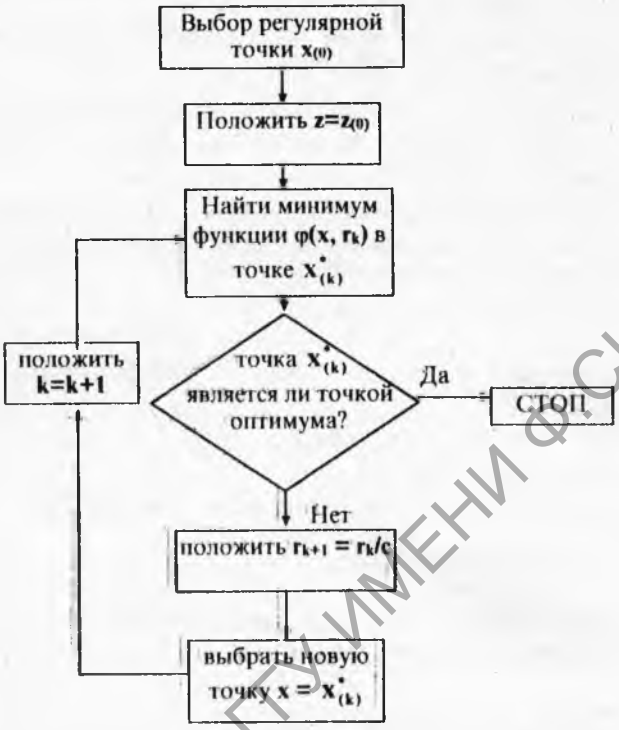
Используя необходимое условие минимума, находим стационарные точки из уравнений

$$\frac{\partial\varphi}{\partial x_1} = 6x_1 + 4x_2 + r\left(-\frac{1}{x_1^2} - \frac{1}{(x_1 + x_2 - 4)^2}\right) = 0;$$

$$\frac{\partial\varphi}{\partial x_2} = 4x_1 + 10x_2 + r\left(-\frac{1}{x_2^2} - \frac{1}{(x_1 + x_2 - 4)^2}\right) = 0.$$

При $r\to 0$ имеем $x_1=3x_2$; точки минимума $x_1=3$; $x_2=1$, $f_{min} = f(x_1=3, x_2=1) = 44$.

Приведем блок-схему метода **SUMT:**



Предполагая, что в начале процедуры имеется допустимая точка, необходимо выбрать начальное значение **r** по формуле

$$r = \frac{-\nabla f(x)^T \nabla P(x)}{\nabla P(x)^T \nabla P(x)}.$$

Затем уменьшаем **r**: $r_{k+1} = \frac{r_k}{c}$,

где, например, **c=10**. Для минимизации функции $\varphi(x, r_{k+1})$ используется метод ДФП при специальном выборе шага, таком, чтобы в процессе одномерного поиска не выйти за

область ограничений (принимаем $\lambda = \frac{\lambda}{\alpha}$, где, например, α=1.05).

Функция $\varphi(x,r)$ минимизируется до тех пор, пока два последовательных значения F_1 и F_2 не станут такими, что $|(F_1-F_2)/F_1|<\varepsilon$.

Критерий завершения осуществляет остановку программы при $r_k<\varepsilon$.

Описанный алгоритм реализован в виде программы.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Построить математическую модель задачи проектирования.
2. Привести задачу оптимизации с ограничениями к задаче на безусловный минимум используя штрафные функции.
3. Используя приведенную программу, составить программу вычисления полученной функции.
4. Провести вычисления на ПЭВМ.

Пример. Найти минимум функции

$$f(x) = 3x_1^2 + 4x_1x_2 + 5x_2^2$$

при ограничениях $x_1 \geq 0$, $x_2 \geq 0$, $x_1+x_2 \geq 4$.

В качестве начального приближения выберем $x_{1(0)}=2$; $x_{2(0)}=2$.

Программа, реализующая решение задачи, может иметь вид

```
{Метод минимизации функций с ограничениями с
использованием}
{          штрафных функций. }

program shtraf;
uses crt,printer;
var
  i,ii,j,jj,mm,n,cc,s:integer;

z1,z2,fp,r,z,g1,g0,gp,qx,hh,bb,fq,g2,gq,zz,ww,w,dd,fr,
gr,kk,wk:real;
  ka,kb,kc,t,ff,l:real;
  h:array[1..10,1..10] of real;
```

```
  x,pr,y,u,d,g,q,v,m,c,cg,ir:array[1..10] of real;
  p,pp,ppp:boolean;

{ Вычисление значения функции }
procedure prcd1330;
begin
  z1:=3*sqr(x[1])+4*x[1]*x[2]+5*sqr(x[2]);
  z2:=0;
  for jj:=1 to mm do
    z2:=z2+1/c[jj];
  z:=z1+r*z2;
end;

{ Вычисление значения градиента }
procedure prcd1380;
begin
  g[1]:=6*x[1]+4*x[2];
  cg[1]:=-1/sqr(x[1])-1/sqr(x[1]+x[2]-4);
  g[1]:=g[1]+r*cg[1];
  g[2]:=4*x[1]+10*x[2];
  cg[2]:=-1/sqr(x[2])-1/sqr(x[1]+x[2]-4);
  g[2]:=g[2]+r*cg[2];
  g0:=0;
  for jj:=1 to n do
    g0:=g0+g[jj]*g[jj];
  g0:=sqr(g0);
end;

procedure prcd8000;
  begin
    if ii=1 then   c[1]:=x[1];
    if ii=2 then   c[2]:=x[2];
    if ii=3 then   c[3]:=x[1]+x[2]-4;
  end;


begin
  clrscr;
  write('Введите число переменных ==> ');
  readln(n);
  write('ВВедите кол-во ограничений ==> ');
  readln(mm);
  writeln('Введите начальную точку');
  for i:=1 to n do
    begin
```

```
    write('x',i,'==> ');
    readln(x[i]);
  end;
clrscr;
{ Промежуточный вывод }
writeln(lst,'Начальные значения');
for i:=1 to n do
  writeln(lst,'x[',i,']=',x[i]:6:3);
{ Проверка вып. ограничений }
{ если ir[j]=0 - выполнено, ir[j]=1 - нарушено }
s:=0;
for ii:=1 to mm do
  begin
    prcd8000;
    if c[ii]<0 then
      begin
        s:=s+1;
        ir[ii]:=1;
      end;
  end;
if s>0 then
  begin
    writeln('Точка не является допустимой');
    readln;
    halt;
  end;
t:=0;
hh:=0;
r:=0;
cc:=0;
prcd1380;
for i:=1 to n do
  begin
    t:=t-g[i]*cg[i];
    hh:=hh+cg[i]*cg[i];
  end;
r:=t/hh;
if r<0 then r:=1;
ppp:=true;
while ppp do
  begin
    writeln('r=',r:7:5);
    { Задаем H единичной матрицей }
    for i:=1 to n do
      begin
```

```
          for j:=1 to n do
            h[i,j]:=0;
          h[i,i]:=1;
        end;
      pp:=true;
      while pp do
        begin
          writeln('Текущие значения');
          for i:=1 to n do
            begin
              pr[i]:=x[i];
              y[i]:=x[i];
              writeln('x',i,'=',x[i]:6:3);
            end;
          for ii:=1 to mm do
            prcd8000;
          prcd1330;
          writeln('Итерация ',cc,' значение функции
=',z:7:4);
          fp:=z;
          prcd1380;
          g1:=g0;
          ff:=z;
          p:=true;
          while p do
            begin
             {Градиент запомнить в U и выбрать
нач.направление D}
              for i:=1 to n do
                begin
                  u[i]:=g[i];
                  d[i]:=0;
                  for j:=1 to n do
                    d[i]:=d[i]-h[i,j]*g[j];
                end;
              gp:=0;
              for i:=1 to n do
                gp:=gp+g[i]*d[i];
              { Найти нач. шаг и,если
необходимо,изменить }
              {   направление спуска на
противоположное   }
              if gp>0 then writeln('Функция
возрастает');
              l:=2;
```

```
                for i:=1 to n do
                  x[i]:=pr[i]+l*d[i];
                s:=1;
                while s>0 do
                  begin
                    s:=0;
                    for ii:=1 to mm do
                      begin
                        p:=true;
                        while p do
                          begin
                            ir[ii]:=0;
                            prcd8000;
                            if c[ii]>=0 then p:=false
                              else
                                begin
                                  ir[ii]:=1;
                                  s:=s+1;
                                  l:=l/1.05;
                                  for i:=1 to n do
                                    x[i]:=pr[i]+l*d[i];
                                end;
                          end;
                      end;
                end;
                { Найти след. точку Q }
                hh:=1;
                for i:=1 to n do
                  begin
                    q[i]:=pr[i]+hh*d[i];
                    x[i]:=q[i];
                  end;
                for ii:=1 to mm do
                  prcd8000;
                prcd1330;
                fq:=z;
                prcd1380;
                g2:=g0;
                gq:=0;
                for i:=1 to n do
                  gq:=gq+g[i]*d[i];
                { Выполнить кубич. интерполяцию или
удвоить }
                {      шаг, чтобы 'накрыть' минимум
}
```

```
      if (gq>=0) and (fq>=fp) then p:=false
        else
          begin
            for i:=1 to n do
              begin
                for j:=1 to n do
                  h[i,j]:=h[i,j]-d[i]*d[j];
                pr[i]:=q[i];
                x[i]:=pr[i];
                y[i]:=x[i];
              end;
            ff:=z;
            fp:=z;
            g1:=g0;
          end;
    end;
  p:=true;
  while p do
    begin
      zz:=3*(fp-fq)/hh;
      zz:=zz+gp+gq;
      ww:=zz*zz-gp*gq;
      if ww<0 then ww:=0;
      w:=sqr(ww);
      dd:=hh*(1-(gq+w-zz)/(gq-gp+2*w));
      for i:=1 to n do
        x[i]:=pr[i]+dd*d[i];
      for ii:=1 to mm do
        prcd8000;
      prcd1330;
      fr:=z;
      prcd1380;
      { Найти градиент в новой точке }
      gr:=0;
      for i:=1 to n do
        gr:=gr+g[i]*d[i];
      if (z<=fp) and (z<=fq) then p:=false
        else
          if gr>0 then
            begin
              hh:=dd;
              for i:=1 to n do
                q[i]:=x[i];
              fq:=z;
              gq:=gr;
```

```
                    g2:=g0;
                  end
                    else
                      begin
                        hh:=hh-dd;
                        for i:=1 to n do
                          pr[i]:=x[i];
                        fp:=z;
                        gp:=gr;
                        g1:=g0;
                      end;
            end;
            { Обновить матрицу H }
            kk:=0;
            wk:=0;
            for i:=1 to n do
              begin
                u[i]:=g[i]-u[i];
                v[i]:=x[i]-y[i];
              end;
            for i:=1 to n do
              begin
                m[i]:=0;
                for j:=1 to n do
                  m[i]:=m[i]+h[i,j]*u[j];
                kk:=kk+m[i]*u[i];
                wk:=wk+v[i]*u[i];
              end;
            if (kk<>0) or (wk<>0) then
              for i:=1 to n do
                for j:=1 to n do
                  h[i,j]:=h[i,j]-
m[i]*m[j]/kk+v[i]*v[j]/wk;
            cc:=cc+1;
            { Проверка критерия завершения }
            if abs((ff-z)/ff)<0.00001 then pp:=false
                                      else ff:=z;
          end;
        if r*z2<0.00001 then ppp:=false
                        else r:=r/10;
        { Начать новую итерацию поиска }
      end;
    writeln(lst,'Минимум найден');
    writeln(lst,'Кол-во итераций =',cc,' значение
минимума=', z:7:4);
```

```
  writeln(lst,'Точка с координатами');
  for i:=1 to n do
    writeln(lst,'x',i,'=',x[i]:6:3);
  readln;
end.
```

ЗАДАЧИ И УПРАЖНЕНИЯ

1. Вал, передающий крутящий момент, проектируется для одного из устройств аэрокосмической системы **(Рис.7).** Его длинна 10см, максимальный крутящий момент **T=50н·м.** Требуется выбрать такую конструкцию вала, при которой он будет обладать необходимой прочностью при наименьшем возможном весе. Используется материал, обладающий высокой прочностью (плотность ρ=**437кг/м^3**, модуль упругости **E=121ГПа**). По практическим соображениям, максимальное значение радиуса **r** не должно превышать **2см**, а толщина стенки вала **t** должна быть не менее **1мм.**

Указание. Расчет на прочность вала произвести по сдвиговым напряжениям с ограничением $\frac{T}{2\pi r^2 t} \le \frac{S}{N}$, где **N** - коэффиц. запаса прочности, **N = 1.5, S=0.4 ГПа.**

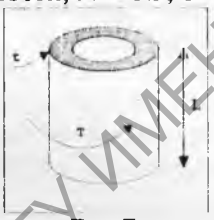


Рис.7.

2. Емкость отстойника для жидких отходов **(Рис.8)** должна составлять **40000л.** Изготавливается отстойник из железобетона толщиной **10см**. Определить геометрические параметры отстойника, при которых на его изготовление пойдет минимальное количество бетона.

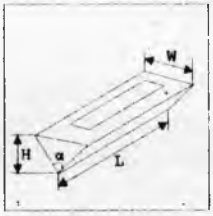


Рис.8.

3. Решить задачу для случая, когда отстойник имеет крышу.

4. Изготовитель контейнеров проектирует открытый контейнер из листового материала, раскрой которого показан на **рис.9.** Заготовка вырезается, сгибается и сваривается. Каковы должны быть размеры контейнера наибольшего объема, если площадь его дна не должна превышать **1м²** и их один из линейных размеров **a, в, с** не должен быть больше другого более чем в **3** раза.

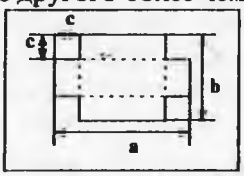


Рис.9.

5. Для хранения авиационного горючего требуется спроектировать бак нового типа **(Рис.10.)**. Бак имеет вид цилиндра с коническими передними и задними днищами. Объем бака должен составить **1м³**. С целью достижения минимального веса и стоимости, требуется изготовить бак из минимального количества материала. Найти значения L_1, L_2, и **D**.

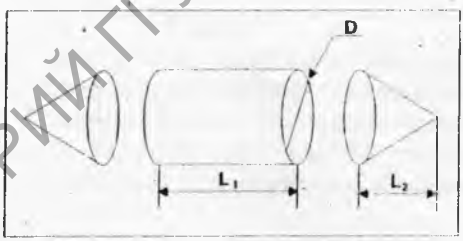


Рис.10.

6. Решить задачу **5** в случае, если бак имеет вид цилиндра с усеченными коническими днищами **(Рис.11.)** при **d=kD**, где **0<k<1**.

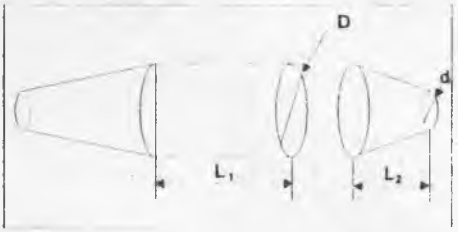


Рис.11.

6. Найти значения параметров h, d, φ (Рис.12.) таким образом, чтобы бункер имел заданный объем (10м²), а его стоимость была минимальная. Основание сделано из деревянных плит стоимостью **C1 = 1 долл/м²**, а остальная часть из металла стоимостью **C2 = 1,5 долл/м²**.

Рис.12.

8. Считая, что соотношения между давлением, молярным объемом и температурой реальных газов описывается полуэмперически $P = \frac{RT}{V-b} - \frac{a}{T^{1/2}V(V+b)}$ (для идеальных газов **PV=RT**). Необходимо, применяя результаты экспериментальных исследований **(см.табл.)**, определить **a** и **b**.

Указание. Для нахождения параметров использовать метод наименьших квадратов $L(a,b) = \sum_{i=1}^{N}\left[P_i - f(x_i, a, b)\right]^2 \to \min$.

Таблица:

№ эксперимента	P, атм.	V, см³/г·моль	T, k
1	33	500	273
2	43	500	323
3	45	600	373
4	26	700	273
5	37	600	323
6	39	700	373
7	38	400	373
8	63.6	400	373

Здесь введены обозначения: **P** - давление **(атм.)**, **V** - молярный объем **(см³/г·моль)**; **T** - температура **(К)**. **R** - универсальная газовая постоянная **(82.06 атм·см³/г·моль)**, **f(x,a,b)** - теоретические значения.

9. Найдите наилучшие оценки параметров B_0, B_1, C модели $\hat{y} = B_0 + B_1 e^{-Cx}$ путем минимизации $\sum(\hat{y}_i - y_{эксп})^2$, при заданных значениях $y_{эксп}$ и x.

$y_{эксп,i}$	x
51.6	0.4
53.4	1.4
20.0	5.4
-4.2	19.5
-3.0	48.2
-4.8	95.9

10. Определить коэффициенты уравнения $y = ax_1^{\beta_1} x_2^{\beta_2}$ путем минимизации суммы квадратов разностей между экспериментальными и предсказанными значениями.

$y_{эксп,i}$	x_1	x_2
46.5	2.0	36.0
591	6.0	8.0
1285	9.0	3.0
36.8	2.5	6.25
241	4.5	7.84

1075	9.5	1.44
1024	8.0	4.0
151	4.0	7.0
80	3.0	9.0
485	7.0	2.0
632	6.5	5.0

11. Статистический анализ данных, апроксимированных экспоненциальными функциями $y = \frac{K_1}{K_1 - K_2}\left(e^{-K_2 t} - e^{-K_1 t}\right)$.

Минимизируйте сумму квадратов отклонений

$F = \sum_{i=1}^{n}\left(\hat{y} - y_{np}\right)_i^2 \rightarrow \min$ при

$\hat{y}$	0.263	0.455	0.548
t	0.5	1	1.5

12. Повторите задачу 11 для модели $y = \frac{K_1 x_1}{4 + K_1 x_2 + K_3 x_3}$

и данных

$\hat{y}$	x_1	x_2
0.126	1	1
0.219	2	1
0.076	1	2
0.126	2	2
0.186	0.1	0

13. Предприниматель может производить товар **A** с затратами в **20** долларов за килограмм и товар **B** с затратами в **10** долларов за килограмм. Фирма может продавать **1000000/x²y** кг товара **A** в день и **2000000/xy²** товара **B** в день, - **x** продажная цена **A** в долларах за **кг**, а **y** - продажная цена **B** в долларах за **кг**. Определить максимальную прибыль, если:
1. **A** и **B** продаются по одной и той же цене. Чему равны x и y?
2. **A** и **B** продаются по разным ценам. Чему равны x и y?

14. Для перевозки газа спроектировать замкнутый цилиндрический баллон **(Рис.13)**, минимального веса,

вмещающий **V=60м³** газа при давлении **P=3,5·10⁶ Н/м²**, выбрав в качестве переменных проектирования средний радиус **R** и толщину стенки **t**. Напряжение в радиальном направлении баллона **σ** не должно превышать **10⁸ Н/м²** , а окружная деформация **ε** не превышает **ε=10⁻³ м**. Напряжение и деформация могут быть записаны в виде:

$$\sigma = \frac{PR}{t}, \quad \varepsilon = PR(2 - \nu)/(2Et),$$

где **E** - модуль Юнга **(E=2.1·10¹¹ Н/м²)**, **ν** -коэффициент Пуассона **ν=0.3**.

Длина баллона **L=8м**. Материал имеет плотность **ρ**; **ρg=7.7·10⁴ Н/м³**, где **g** -ускорение свободного падения.

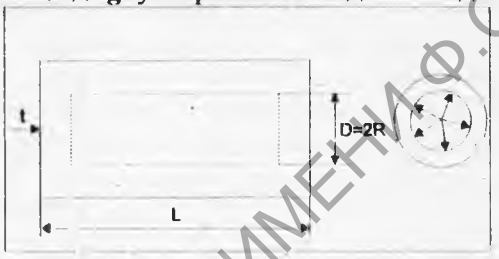


Рис.13.

15. Контейнер для воды для космонавтов сделан из сферы и конуса, основа которого равна радиусу сферы **(Рис.14)**. Если радиус **r=183см** и площадь поверхности не более **13716см²**, найти размеры x_1 и x_2 такие, чтобы объем контейнера был максимальный.

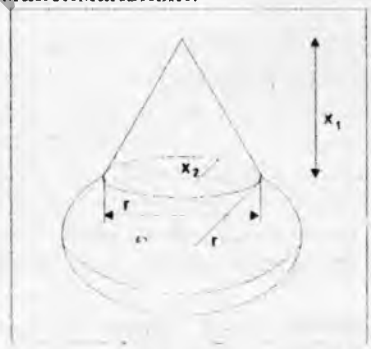


Рис.14.

16. Необходимо оптимально спроектировать трехстержневую ферму **(Рис.15)** так, чтобы ферма была бы легкой и удовлетворяла ограничениям на напряжение $\sigma_i < [\sigma_i]$, $i=1,2,3$, т.е. выбрать площадь поперечных сечений стержней x_1, x_2, x_3 так, чтобы вес был минимальным.
Принять **P=909кг**, допустимое напряжение $[\sigma_i] = 140$ кг/см^2; $\theta = 30^\circ, 45^\circ, 60^\circ$. Решить задачу, если $x_1=x_2$.

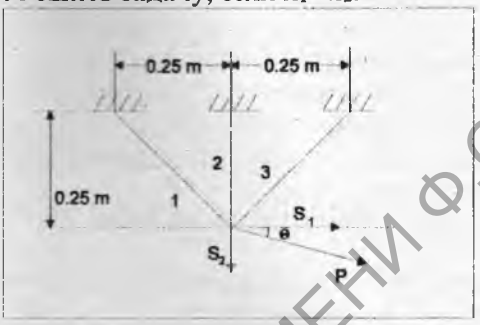


Рис.15.

17. Из круглого бревна диаметром $d = 1/\sqrt{c}$ необходимо выпилить балку прямоугольного сечения так, чтобы величина bh^2, характерезующая ее сопротивление изгибу, была максимальной (**b** и **h** - размеры поперечного сечения балки).

18. Из куска жести площадью **S=2a** необходимо изготовить в форме параллелепипеда коробку с максимальным объемом **V**. Чему равны стороны? Если **a=400см^2**.

19. Найти размер цилиндрической емкости заданного объема **V=1**, которая имела бы минимальную поверхность.

Литература

1. Банди Б. Методы оптимизации. - М: Радио и связь. 1988. 127с.
2. Дьяконов В.П. Справочник по алгоритмам на языке Бейсик для персональных ЭВМ. - М: Наука. 1987. 240с.
3. Химмельблау Д. Прикладное нелинейное программирование. - М: Мир. 1975. 537с.
4. Реклейтис Г., Рейвиндран А., Рэгсдел К. Оптимизация в технике. Т 1. - М: Мир. 1986. 347с.
5. Хог Э., Арора Я. Прикладное оптимальное проектирование. - М: Мир. 1983. 471с.
6. Шуп Т. Решение инженерных задач на ЭВМ. - М. Мир. 1982. 235с.
7. Плис А.И. и др. Лабораторный практикум по курсу высшей математики. - М: Высшая школа. 1979. 150с.

Содержание

Лабораторный практикум и методические указания по спецкурсу лекций "Оптимальное проектирование" для студентов математического факультета.

Составитель :

Можаровский Валентин Васильевич.

Подписано к печати 17.09.96.

формат 60x84 1/16. Бумага писчая №1. Печать офсетная.

Усл. п. л. 3,49.

Тираж экз. 120. Заказ № 2027.

Отпечатано на ротапринте БелГУТа, 246022,

г.Гомель, ул. Кирова, 34.