Создать ПС с использованием технологий программирования: модульное программирование, структурное и объектно-ориентированное.

При использовании модульного программирования сделать описание метода разработки структуры программы (восходящая разработка, нисходящая, конструктивный подход, архитектурный подход), построить структуру программы в виде дерева.

При структурном программировании сделать схемы алгоритмов основной программы и всех используемых функций, с соответствующими описаниями.

При объектно-ориентированном подходе – диаграммы классов.

**Постановка задачи:**

Затраты топлива БПЛА при полете над сектором местности на фиксированной высоте определяются следующим соотношением:

F1(x) = 9X12 + 16X2 2 - 90X1- 128X2,

где X1, X2 – широта и долгота.

Коэффициент искажения качества снимков, производимых БПЛА при изменении траектории полета определяется следующим соотношением:

F2(x) = X12 + 2X1 X2  + 2X22 + X32 - X2X3 +X1 + 3X2 - X3

где X1 - скорость полета, X2 – угловая скорость в оси тангажа, X3 – высота полета.

Провести:

1. Найти методом Розенброка c дискретным шагом экстремумы функций F1(x) и F2(x) при различных значениях начальной точки;
2. Определение экстремума заданных функций при различных значениях точности

( ;

1. На каждой итерации метода предусмотреть вывод следующей информации:

* номера текущей итерации;
* координаты текущей точки;
* значение функции в текущей точке;
* на каждой итерации: номер переменной (j), допустимое направление dj, величину шага λj по направлению j, координату полученной точки (yj+1) и т.д.

Пример вывода результатов вычислений по методу Розенброка с дискретным шагом:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K | Xk  F(Xk) | j | yj  F(yj) | j | dj | yjjdj  F(yjjdj) |
| 1 | (0.00,3.00)  52.00 | 1    2  1  2  . . . . | (0.00, 3.00)  52.00  (0.10, 3.00)  47.84  (0.10, 3.00)  47.84  (0.30, 3.00)  40.84  . . . . . . | 0.10  0.10  0.20  - 0.05 | (1.0, 0.0)  (0.0, 1.0)  (1.0, 0.0)  (0.0, 1.0) | (0.10, 3.00)  47.84  (0.10, 3.10)  50.24  (0.30, 3.00)  40.84  (0.30, 2.95)  39.71 |
| 2 | . . . . . . |  |  |  |  |  |
| . . |  |  |  |  |  |  |
| N |  |  |  |  |  |  |

1. В конце вычислений выводить следующую информацию:

* оптимальное значение аргумента;
* оптимальное значение функции;
* количество итераций метода для достижения оптимальной точки.

1. Построить график и показать на нем:

* линии уровня функции F1(x);
* точки, получаемые на каждой итерации, начиная от начальной и заканчивая конечной, с изображением вектора направления от каждой предыдущей точки до последующей.

6. Сделать выводы по результатам исследований.

**Формулировка метода Розенброка с дискретным шагом:**

Задать начальные параметры:

> 0 – точность;

> 1 – коэффициент растяжения;

(-1, 0) – коэффициент сжатия,

– начальные длины шагов вдоль каждого направления;

– начальная точка.

Положить

Шаг 1.

Если , то шаг считается успешным и он осуществляется: . При этом .

Если же , то шаг считается неудачным: и .

Шаг 2.

Если , то положить и перейти на шаг 1.

Шаг 3.

Если , т.е. спуск хотя бы по одному из направлений оказался удачным, то и перейти на шаг 1.

Шаг 4.

Если , т.е. каждый из последних спусков по направлению был неудачным) и , то перейти на шаг 5.

Если же последнее условие не выполняется (т.е. ), то при (конец работы алгоритма), иначе , , перейти на шаг 1.

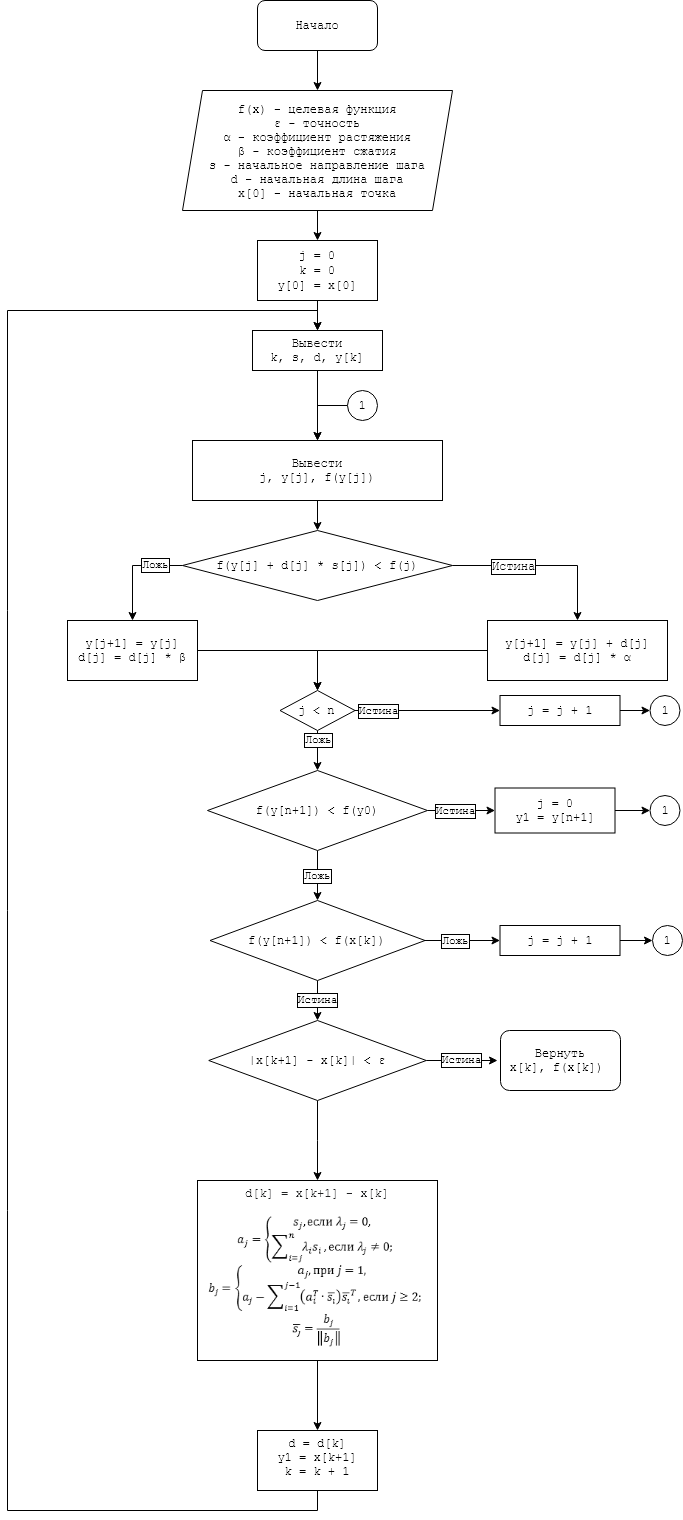
Шаг 5.

Положить . Если , то (конец работы алгоритма), иначе вычислить из соотношения и построить новые направления , на основе следующих соотношений:

Шаг 6.

Положить: для . Перейти на шаг 1.

Блок-схема программы:



Зададим начальные параметры:

ε = 0.01

α= 3.0

β = -0.5

s =

delta = (0.5, 0.5)

Результаты итераций для функции F1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K | Xk  F(Xk) | j | yj  F(yj) | j | dj | yjjdj  F(yjjdj) |
| 1 | (2.50,2.50)  -388.75 | 0    1 | (2.50, 2.50)  -388.75  (3.00, 2.50)  -409.00 | 0.5  0.5 | (1.0, 0.0)  (0.0, 1.0) | (3.00, 2.50)  -409.00  (3.00, 3.00)  -429.00 |
| 2 | (3.00, 2.50)  -429.0 | 0  1  0  1 | (3.00, 2.50)  -429.0  (3.35, 3.35)  -449.92  (3.35, 3.35)  449.92  (4.41, 4.41)  -475.17 | 0.5  0.5  1.5  -0.25 | (0.70, 0.70)  (-0.70, 0.70)  (0.70, 0.70)  (-0.70, 0.70) | (3.35, 3.35)  -449.92  (3.00, 3.71)  -448.91  (4.41, 4.41)  -475.17  (4.59, 4.24)  -478.59 |
| 3 | (4.59, 4.24)  -478.59 | 0  1  0  1 | (4.59, 4.24)  -478.59  (4.59, 4.24)  -478.59  (4.59, 4.24)  -478.59  (4.59, 4.24)  -478.59 | 2.09  1.74  -1.05  -0.87 | (0.09, 0.99)  (-0.99, 0.09)  (0.09, 0.99)  (-0.99, 0.09) | (4.78, 6.32)  -394.49  (2.86, 4.40)  -478.59  (4.49, 3.19)  -478.59  (5.56, 4.16)  -478.59 |
| 4 | . . . . . . |  |  |  |  |  |
| . . |  |  |  |  |  |  |
| 7 | (4.99, 4.00)  -480.99 | 1  2 | (4.99, 4.00)  -480.99  (5.00, 4.00)  -481.00 | -0.02  -0.01 | (-0.90, -0.43)  (0.43, -0.90) | (5.00, 4.00)  -481.00  (5.00, 4.00)  -481.00 |

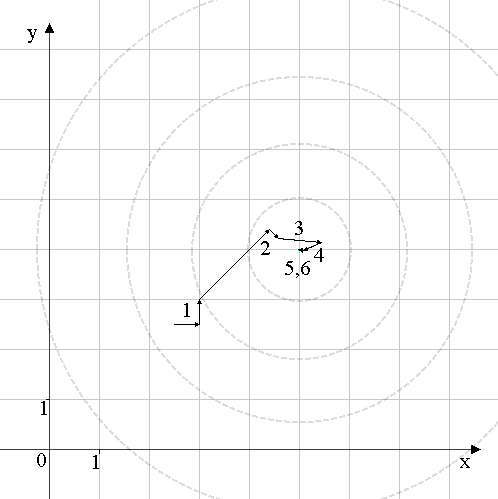


Рисунок 1 – График шагов итераций (цифрами указаны номера итераций).

Псевдокод:

START

INPUT ε

INPUT delta

INPUT α

INPUT β

INPUT x1

y := x1

n := 2

k := 1

j := 0

s := [[1, 0], [0, 1]]

WHILE k < 10

PRINT k

PRINT y

value := function(y)

PRINT value

WHILE j < n

y\_with\_step := y + delta[j] + s[j]

PRINT y\_with\_step

stepvalue = function(y\_with\_step)

IF step\_value < value THEN

delta[j] := delta[j] \* a

y := y\_with\_step

last\_successful\_point := y\_with\_step

ELSE

delta[j] := delta[j] \* b

ENDIF

ENDIF

IF y = y\_with\_step

y := last\_successful\_point

j := 0

CONTINUE

ELSEIF |funtction(y) – funtction(y\_with\_step)| < ε

BREAK

ENDIF

FOR j = 0, ..., n

delta[j] := last\_successful\_point[j] – y[j]

ENDFOR

A := []

FOR j = 0, ..., n

FOR i = j, ..., n

A = A + delta[i]\*s[i]

ENDFOR

ENDFOR

s[0] = A[0] / ||A[0]||

B := []

FOR j = 0, ..., n

B\_sum := []

FOR i = 0, ..., j

B\_sum := B\_sum + A[j]T \* s[i] \* s[i]T

ENDFOR

B := A[j] – B\_sumT

S[j]= B / ||B||

ENDFOR

j := 0

ENDWHILE

PRINT k

PRINT y

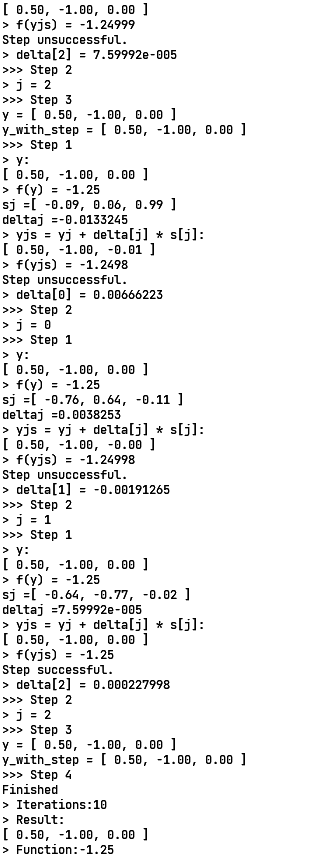
PRINT function(y)

END

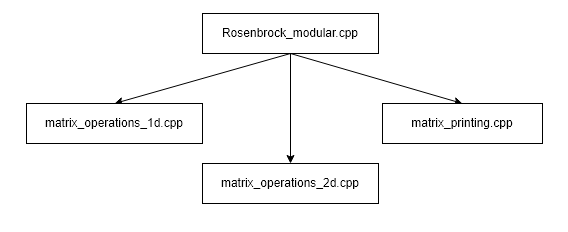
Пример выполнения программы, написанной с применением процедурного подхода, для первой функции. Исходные значения параметров:

|  |  |
| --- | --- |
| Начало выполнения программы: | Завершение выполнения программы: |

Пример выполнения программы, написанной с применением процедурного подхода, для первой функции. Исходные значения параметров:

****

Древовидная схема для программы, написанной с применением модульного подхода:



При применении этого подхода, алгоритм программы было принято разбить на 4 составляющие:

1. Модуль “Rosenbock\_modular” содержащий функцию с основным блоком кода выполняющим итерации алгоритма оптимизации. Он содержит 3 зависимости к модулям, содержащим операции для работы с матрицами.
2. Модуль ”matrix\_operations\_2d” содержащий функции для работы с одномерными массивами: суммирование, разность, проверка на эквивалентность, умножение на скаляр, нахождение нормы, перевод в двумерную матрицу.
3. Модуль “matrix\_operations\_1d” содержащий функции для выполнения операций с двумерными матрицами: суммирование, разность, транспонирование и умножение.
4. Модуль “matrix\_printing” содержащий функции для выведения одномерных и двумерных массивов в консоль.

UML диаграмма для программы, написанной с применением объектно-

ориентированного подхода:

