**Программирование в САПР**

Лабораторная работа №5

***Тема занятия:***

*Численные методы. Многомерная минимизация*

***Аннотация:***

Изучение численных методов минимизации функции нескольких переменных

***План занятия:***

1. Подготовка к выполнению работы, в том числе:
   1. повторение ранее изученного материала по методам одномерной минимизации
   2. изучение теоретических основ численных методов многомерной минимизации
   3. повторение ранее изученного материала по архитектуре MFC DialogBased Application;
2. Создание диалогового приложения на языке C++ для нахождения функции нескольких переменных в соответствии с индивидуальным заданием.
3. Защита лабораторной работы.

К сожалению, рассмотренная нами ранее задача одномерной минимизации на практике зачастую является лишь подзадачей в более сложных программных расчетах. Характерным примером такого случая может служить задача *аппроксимации* экпериментально полученных данных какой-либо аналитической зависимостью.

*Аппроксима́ция* (от лат. proxima — ближайшая) или приближе́ние — научный метод, состоящий в замене одних объектов другими, в каком-то смысле близкими к исходным, но более простыми.

Например, на специализированном измерительном приборе мы контроллируем точность изготовления шеек ступечатого вала. Результатом работы прибора является облако точек с координатами X,Y в некоторой общей для них двумерной системе координат. Нашей задачей является оценить диаметр шейки вала и ее отклонение от круглости по полученным данным.

**Первый этап**. Нам необходимо подобрать вид аналитической зависимости (функции), которой мы будем аппроксимировать экспериментальные данные. Очевидно, что поперечное сечение шейки вала конструктивно должно представлять из себя окружность. Но ввиду наличия погрешностей изготовления и измерения исходные данные искажаются и мы получаем некотрую более сложную, возможно случайную зависимость. Однако для целей апраксимации окружность подойдет наилучшим образом, поскольку именно она заложена в конструктив детали.

**Второй этап.** Выбор начального приближения. Для старта любого итерационного процесса нужна отправная точка и чем она ближе к правильному решению тем более скоротечен будет процесс. Выбранная в первом этапе для аппроксимации окружность описывается тремя параметрами: Xc, Yc, R – координаты центра и радиус соответственно. Нам нужно опреелить с каких значений этих параметров мы начнем поиск. *Предлагаю Вам сделать это самостоятельно и предоставить аргументированный письменный ответ вместе с решением задачи.*

**Третий этап. Выбор критерия оптимальности.** Очевидно, что если мы нанесем исходные данные на диаграмму и попробуем вручную «нарисовать» по ним окружность, то визуально мы можем получить вполне приемлемый для такого метода результат. Но если эту процедуру выполнит еще несколько человек, результат у каждого будет разным. Так какой же из них верен? Кто прав?

Нужно сформулировать критерий правильности (оптимальности) результата. Как Вы наверное уже догадались такой критерий был уже давно сформирован. Сделал это еще Гаусс, *разработавший метод наименьших квадротов (МНК).* Суть его состоит в том, что оптимальной для аппроксимации признается такая зависимость сумма квадратов отклонений которой в исследуемых точках от исходных данных минимальна.

**Четвертый этап. Метод покоординатного спуска.** Мы только что поняли что нам предстоит минимизировать некоторую функцию (суммы квадратов отклонений). Делать это с функцией одной переменной мы умеем (хвала предыдущей л/р). А что делать если мы имеем дело с функцией нескольких переменных – Xc, Yc, R? Нам на помощь придет *метод покоординатного спуска*. В этом случае мы разбиваем задачу на несколько (по числу переменных) одномерных мнимизаций, «замораживая» при этом остальные переменные. То есть когда мы минимизируем функцию по Xc, Yc и R становятся константами. Нашли минимум по Xc морозим его, размораживаем Yc и ищем минимум по нему. Нашли минимум по Yc морозим его вновь и переходим к разморозке R. После нахождения минимума по R процесс опять переходит на Xc и так до победного конца. Вопрос: когда остановиться? Критерием остановки процесса будем считать такое состояние при котором модуль вектора разности текущего и предыдущего решений будет меньше заданной погрешности вычислений.

Метод покоординатно спуска в случае двух переменных можно проиллюстрировать следующим примером: Вы спускаетесь на дно котлована, но имеете право передвигаться поочередно только по направлениям север-юг и восток-запад. Графически это выглядело бы так:

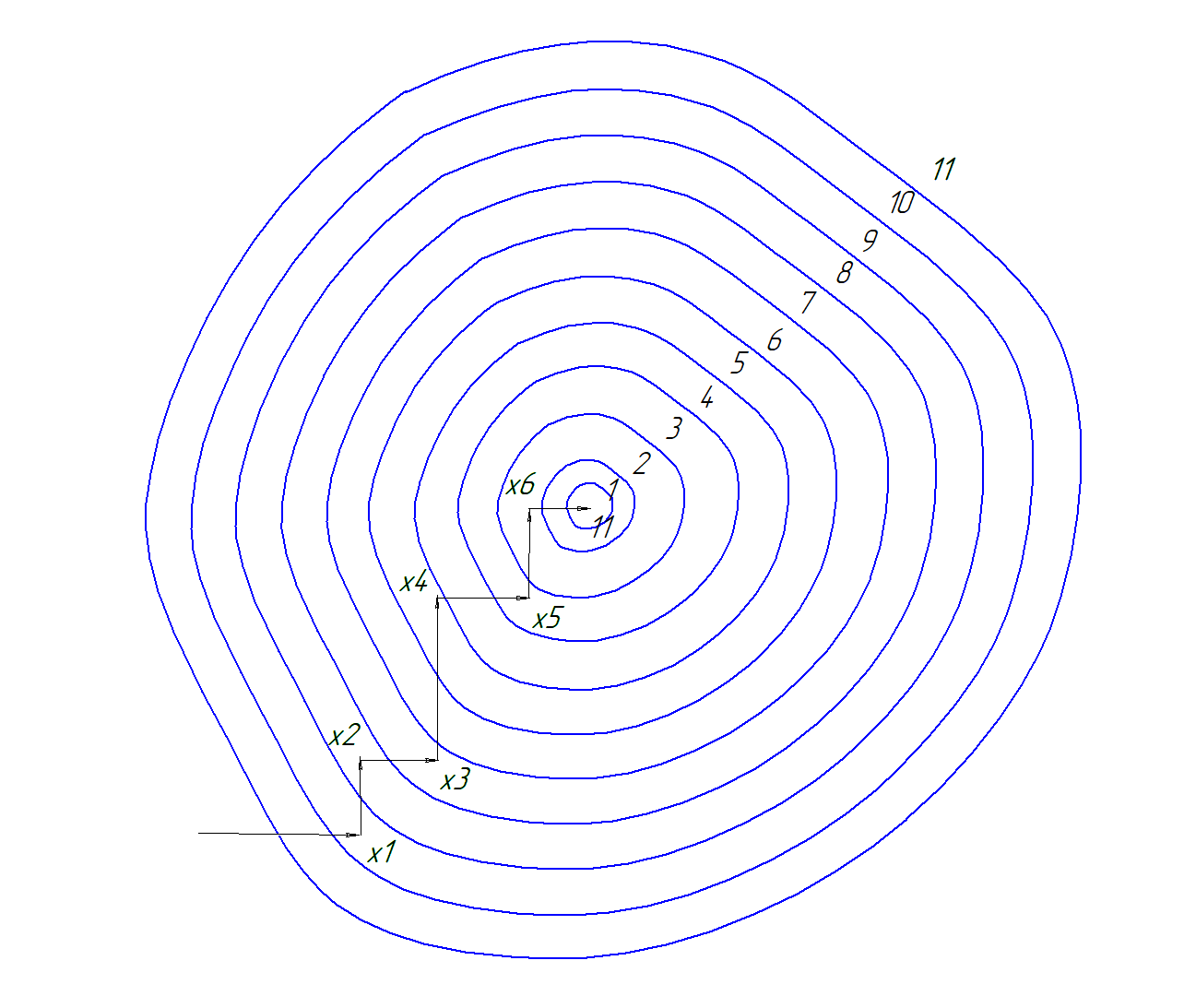


Рисунок 1. Графическая интерпретация метода покоординатного спуска

Рассмотрим программный код реализующий метод покоординатного спуска.

#include "stdafx.h"

#include "Approximation.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

CWinApp theApp;

using namespace std; //используем стандартное пространство имен

struct Point//структура для описания точки

{

double x, y; //координаты точки в двумерном пространстве

};

//динамический массив для точек

CArray <Point, Point> Points;

void Alert(string str) //функция вывода предупреждения

{

str += "\n";

cout << str.c\_str();

system("pause");

}

//функция вычисления среднеквадратичного отклонения для текущих значений a,b,r

double MSQ(double a, double b, double r)

{

double res = 0;

//цикл по всем точкам

for (int i = 0; i < Points.GetSize(); i++)

{

//отклонение по нормали к дуге окружности

double delta = pow(Points[i].x - a, 2) + pow(Points[i].y - b, 2) - r\*r;

res += delta\*delta; //возводим отклонение в квадрат

}

return res; //возвращаем результат в программу верхнего уровня

}

#define CT\_A 0 //константы для задания режима «заморозки» переменных a,b,r

#define CT\_B 1 // константы для задания режима «заморозки» переменных b,a,r

#define CT\_R 2 // константы для задания режима «заморозки» переменных r,a,b

//функция реализации метода половинного деления

void LinearDixotomy(double &value1, double value2, double value3,int type)

{

double epsilon = 0.000001f; //погрешность вычисления

double step = epsilon \* 100; //шаг

double V\_Left, V\_Right,MSQ1,MSQ2; //значения на границах интервала и СКО

switch (type) //переключатель по типам расчета

{

case CT\_A: //расчет по а

MSQ1 = MSQ(value1, value2, value3);

MSQ2 = MSQ(value1-step, value2, value3);

break;

case CT\_B: //расчет по b

MSQ1 = MSQ(value2, value1, value3);

MSQ2 = MSQ(value2 , value1- step, value3);

break;

case CT\_R: //расчет по r

MSQ1 = MSQ(value2, value3, value1);

MSQ2 = MSQ(value2 , value3, value1- step);

break;

}

if (MSQ1 > MSQ2)

{

V\_Right = V\_Left = value1;

while (true)

{

V\_Left = V\_Left - step;

switch (type)

{

case CT\_A: //расчет по а

MSQ1 = MSQ(V\_Left, value2, value3);

MSQ2 = MSQ(V\_Left + step, value2, value3);

break;

case CT\_B: //расчет по b

MSQ1 = MSQ(value2, V\_Left, value3);

MSQ2 = MSQ(value2, V\_Left + step, value3);

break;

case CT\_R: //расчет по r

MSQ1 = MSQ(value2, value3, V\_Left);

MSQ2 = MSQ(value2, value3, V\_Left + step);

break;

}

if (MSQ1 > MSQ2)

break;

}

}

else

{

V\_Right = V\_Left = value1;

while (true)

{

V\_Right = V\_Right + step;

switch (type)

{

case CT\_A: //расчет по а

MSQ1 = MSQ(V\_Right, value2, value3);

MSQ2 = MSQ(V\_Right + step, value2, value3);

break;

case CT\_B: //расчет по b

MSQ1 = MSQ(value2, V\_Right, value3);

MSQ2 = MSQ(value2, V\_Right + step, value3);

break;

case CT\_R: //расчет по c

MSQ1 = MSQ(value2, value3, V\_Right);

MSQ2 = MSQ(value2, value3, V\_Right + step);

break;

}

if (MSQ1 < MSQ2)

break;

}

}

double V\_Mid; //значение в середине интервала

//финальная дихотомия

while (V\_Right - V\_Left > epsilon) //цикл дихотомии

{

V\_Mid = (V\_Right + V\_Left) / 2.f; //вычисление середины интервала

switch (type) //переключатель по типам расчета

{

case CT\_A: //расчет по а

MSQ1 = MSQ(V\_Mid + epsilon / 3, value2, value3);

MSQ2 = MSQ(V\_Mid - epsilon / 3, value2, value3);

break;

case CT\_B: //расчет по b

MSQ1 = MSQ(value2, V\_Mid + epsilon / 3, value3);

MSQ2 = MSQ(value2, V\_Mid - epsilon / 3, value3);

break;

case CT\_R: //расчет по r

MSQ1 = MSQ(value2, value3, V\_Mid + epsilon / 3);

MSQ2 = MSQ(value2, value3, V\_Mid - epsilon / 3);

break;

}

if (MSQ1 < MSQ2)

{

V\_Left = V\_Mid;

}

else

{

V\_Right = V\_Mid;

}

}

value1 = (V\_Right + V\_Left) / 2.f; //середина интервала

}

int main()

{

int nRetCode = 0;

HMODULE hModule = ::GetModuleHandle(nullptr);

if (hModule != nullptr)

{

// инициализировать MFC, а также печать и сообщения об ошибках про сбое

if (!AfxWinInit(hModule, nullptr, ::GetCommandLine(), 0))

{

// TODO: измените код ошибки соответственно своим потребностям

wprintf(L"Критическая ошибка: сбой при инициализации MFC\n");

nRetCode = 1;

}

else

{

// TODO: Вставьте сюда код для приложения.

}

}

else

{

// TODO: Измените код ошибки соответственно своим потребностям

wprintf(L"Критическая ошибка: неудачное завершение GetModuleHandle\n");

nRetCode = 1;

}

//читаем данные

FILE \*fl;

fopen\_s(&fl, "D:\\data.txt", "rt");

if (!fl) //что-то не так с файлом

{

Alert("Проблема с файлом");

return 0; //выходим

}

char buf[100]; //буфер под текстовые строки

double x, y; //буфер под координаты

while (!feof(fl)) //цикл пока не кончится файл

{

fgets(buf, 100, fl); //берем строку в текстовый буфер

if (sscanf\_s(buf, "%lf %lf", &x, &y) == 2) //парсим взятую из файла строку

{//распарсилось две координаты

Point pt; //структура-буфер под точки

pt.x = x; //запоминаем х

pt.y = y; //запоминаем y

Points.Add(pt); //добавляем точку в массив структур

}

else//в противном случае

{

Alert("Проблема с данными");//предупреждение об ошибке

return 0; //выходим

}

}

//закрываем файл

fclose(fl);

double A, B, R; //переменные под координаты центра и радиус окружности

A = B = R = 0.f; //обнуляем

//находим начальные приближения

for (int i = 0; i < Points.GetSize(); i++)//цикл по всем точкам

{

A += Points[i].x; //суммируем х

B += Points[i].y; //суммируем y

}

A /= Points.GetSize();//находим среднее значения

B /= Points.GetSize();//находим среднее значения

for (int i = 0; i < Points.GetSize(); i++)

{//радиус каждой точки

R += sqrt(pow(Points[i].x-A,2)+pow(Points[i].y-B,2));

}

R /= Points.GetSize();//вычисляем среднее значение радиуса

double A\_old, B\_old, R\_old; //буфер под предыдущие значения

while (true) //цикл пока не достигнем требуемой точности

{

A\_old = A; //запоминаем старое значение А

B\_old = B; //запоминаем старое значение B

R\_old = R; //запоминаем старое значение R

LinearDixotomy(A, B, R, CT\_A); //запускаем дихотомию по А

LinearDixotomy(B, A, R, CT\_B); //запускаем дихотомию по B

LinearDixotomy(R, A, B, CT\_R); //запускаем дихотомию по R

//вычисляем «длину» интервала и сравниваем с допустимой погрешностью

if (sqrt(pow(A - A\_old, 2) + pow(B - B\_old, 2) + pow(R - R\_old, 2)) < 0.0001)

break; //если достигли требуемой точности - выходим

}

cout << A << endl; //выводим результат по А

cout << B << endl; //выводим результат по B

cout << R << endl; //выводим результат по R

system("pause");

return nRetCode;

}

***Задание для самостоятельной работы***

Написать Dialog-based приложение MFC для реализации нахождения диаметра и отклонения от круглости шейки вала. Координаты точек взять из текстового файла data.tat, который должен быть сформирован с помощью файла «Генератор задач» (столбцы x\* и y\* предварительно установив Xc, Yc, R). Указание файла с данными осуществить через стандатный диалог открытия файла (класс СFileDialog). Допустимую погрешность принять равной 0,00001. Вы можете использовать код примера выше, но в этом случае Вам необходимо модернизировать алгоритм поиска с учетом данных о градиенте функции в текущей точке.

Отклонение от круглости рассчитать, как разность расстояний от центра полученной окружности до максимально и минимально удаленной от него точек разделив на 2.

Результат вычислений выводить в EditControl с точностью 4 знака после запятой в формате:

Для данных из файла …..путь к файлу… Xс=…… Yc = ….. R=….

Отклонение от круглости …..

Данные получены при Xс=…… Yc = ….. R=….

***Форма представления выполненного задания***

На проверку преподавателю требуется представить архив (RAR, ZIP) с проектом MS Visual Studio, реализующим задание для самостоятельной работы.