**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1-40 04 01 «Информатика и технологии программирования»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Операционные системы и среды»

на тему: **«Распределенное компонентное приложение подготовки данных для обработки и отображения поверхности»**

Исполнитель: студент гр. ИП-31

Казимов А.Р.

Руководитель: ст. преподаватель Самовендюк Н.В.

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии

по защите курсового проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2019

Содержание

[Введение 3](#_Toc8944303)

[1 Обзор существующих методов решения задачи 4](#_Toc8944304)

[1.1 Поверхности 4](#_Toc8944305)

[1.2 Решение интегралов 6](#_Toc8944306)

[1.3 Выбор языка программирования для разработки desktop приложения 11](#_Toc8944307)

[1.4 Построение поверхности 14](#_Toc8944308)

[1.5. Математические пакеты 19](#_Toc8944309)

[2. Разработка функциональной модели приложения 22](#_Toc8944310)

[2.1 Постановка задачи 22](#_Toc8944311)

[2.2 Построение функциональной модели 22](#_Toc8944312)

[3 Разработка компонентов 24](#_Toc8944313)

[3.1 Разработка программного кода 24](#_Toc8944314)

[3.2 Разработка графического интерфейса 26](#_Toc8944315)

[3.3 Определение погрешности численного метода 27](#_Toc8944316)

[4 Исследование эффективности реализации многопоточности 29](#_Toc8944317)

[5 Разработка графического интерфейса 30](#_Toc8944318)

[6 Тестирование и верификация 31](#_Toc8944319)

[7 Руководство пользователя 32](#_Toc8944320)

[Заключение 34](#_Toc8944321)

[Список использованных источников 35](#_Toc8944322)

# Введение

Ввиду интенсивного развития вычислительной техники все чаще и чаще встречаются математические задачи, для получения точного решения которых приходится прибегать к огромному количеству громоздких и трудоемких вычислений или же вовсе решение которых найти невозможно. Это происходит главным образом не потому, что мы не умеем этого сделать, а поскольку искомое решение обычно не выражается в привычных для нас элементарных или других известных функциях. В таких случаях имеет место прибегнуть к соответствующим приближенным математическим вычислениям. В связи с ранее сказанным за последние годы математического прогресса все большее применение получили приближенные и численные методы математического анализа и приобрели исключительно важный характер.

Одна из трудностей вычисления и построения площади поверхностей связанна с ограниченностью оперативной памяти компьютера. Хотя объем оперативной памяти вновь создаваемых компьютеров растет очень быстро, тем не менее, еще быстрее возрастают потребности практики в решении задач все большей сложности.

Вычисление интегралов сложных поверхностей решаются, используя все возможности компьютерного процессора. Суть данного метода заключается в разделении задачи вычисление интеграла сложной поверхности на несколько частей и одновременном их решении при помощи ограниченного числа потоков.

В данном курсовом проекте задача состоит в написании приложения для вычисления (численным методам Симпсона) и построения площади сложной поверхности, прибегнув к использованию многопоточности.

# 1 **Обзор существующих методов решения задачи**

## 1.1 Поверхности

Поверхность в [геометрии](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F) и [топологии](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BF%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) −двумерное [топологическое многообразие](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BF%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%B8%D0%B5). Наиболее известными примерами поверхностей являются границы [геометрических тел](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%BE_(%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F)) в обычном трёхмерном евклидовом пространстве. С другой стороны, существуют поверхности (например, [бутылка Клейна](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D1%82%D1%8B%D0%BB%D0%BA%D0%B0_%D0%9A%D0%BB%D0%B5%D0%B9%D0%BD%D0%B0)), которые нельзя вложить в трёхмерное евклидово пространство без привлечения сингулярности или самопересечения[1].

Концепция поверхности применяется в [физике](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D0%BA), [инженерном деле](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D0%BB%D0%BE), [компьютерной графике](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0) и прочих областях при изучении физических объектов. Например, анализ [аэродинамических качеств](https://ru.wikipedia.org/wiki/%D0%90%D1%8D%D1%80%D0%BE%D0%B4%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BA%D0%B0%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE) самолёта базируется на обтекании потоком воздуха его поверхности.

Поверхность определяется как множество [точек](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D1%87%D0%BA%D0%B0_(%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F)), [координаты](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82%D1%8B) которых удовлетворяют определённому виду уравнений:

Если функция (1) [непрерывна](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) в некоторой точке и имеет в ней непрерывные частные производные, по крайней мере одна из которых не обращается в нуль, то в окрестности этой точки поверхность, заданная уравнением (1), будет правильной поверхностью[1].

Помимо указанного выше неявного способа задания, поверхность может быть определена явно, если одну из переменных, например, z, можно выразить через остальные:

Также существует [параметрический](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE%D0%B2%D0%B5%D1%80%D1%85%D0%BD%D0%BE%D1%81%D1%82%D0%B8) способ задания (3). В этом случае поверхность определяется системой уравнений:

Интуитивно простую поверхность можно представить как кусок [плоскости](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%BE%D1%81%D0%BA%D0%BE%D1%81%D1%82%D1%8C_(%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F)), подвергнутый [непрерывным](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) [деформациям](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F) ([растяжениям, сжатиям](https://ru.wikipedia.org/wiki/%D0%90%D1%84%D0%B8%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и [изгибаниям](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B3%D0%B8%D0%B1))[1].

Более строго, простой поверхностью называется образ [гомеоморфного](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D0%BC%D0%B5%D0%BE%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC) отображения (то есть взаимно однозначного и взаимно непрерывного отображения) внутренности единичного квадрата. Этому определению можно дать [аналитическое](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7) выражение.

Пусть на плоскости с прямоугольной системой координат u и v задан [квадрат](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82), координаты внутренних точек которого удовлетворяют неравенствам 0 < u < 1, 0 < v < 1. Гомеоморфный образ квадрата в пространстве с [прямоугольной системой координат](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D1%8F%D0%BC%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82) х, у, z задаётся при помощи формул х = x(u, v), у = y(u, v), z = z(u, v) ([параметрическое задание поверхности](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BF%D0%BE%D0%B2%D0%B5%D1%80%D1%85%D0%BD%D0%BE%D1%81%D1%82%D0%B8)). При этом от функций x(u, v), y(u, v) и z(u, v) требуется, чтобы они были [непрерывными](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) и чтобы для различных точек (u, v) и (u', v') были различными соответствующие точки (x, у, z) и (x', у', z').

Примером простой поверхности является полусфера. Вся же [сфера](https://ru.wikipedia.org/wiki/%D0%A1%D1%84%D0%B5%D1%80%D0%B0) не является простой поверхностью. Это вызывает необходимость дальнейшего обобщения понятия поверхности.

Подмножество пространства, у каждой точки которого есть окрестность, являющаяся простой поверхностью, называется правильной поверхностью.

В [дифференциальной геометрии](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%84%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F) исследуемые поверхности обычно подчинены условиям, связанным с возможностью применения методов дифференциального исчисления. Как правило, это − условия гладкости поверхности, то есть существования в каждой точке поверхности определённой [касательной](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D1%8F%D0%BC%D0%B0%D1%8F) плоскости, кривизны и т. д. Эти требования сводятся к тому, что функции, задающие поверхность, предполагаются однократно, дважды, трижды, а в некоторых вопросах  − неограниченное число раз [дифференцируемыми](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%84%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) или даже [аналитическими функциями](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8). При этом дополнительно накладывается условие регулярности.

Случай неявного задания. Поверхность, заданная уравнением , является гладкой регулярной поверхностью, если , функция F [непрерывно дифференцируема](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%84%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) в своей области определения Ω, а её частные производные одновременно не обращаются в нуль (условие правильности) на всём множестве Ω:

Случай параметрического задания. Зададим поверхность [векторным уравнением](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) , или, что то же самое, тремя уравнениями в координатах (3).

Эта система уравнений (3) задаёт гладкую регулярную поверхность, если выполнены условия:

* система устанавливает взаимно однозначное соответствие между образом и прообразом Ω;
* функции непрерывно дифференцируемы в Ω;
* выполнено условие не вырожденности:

Геометрически последнее условие означает, что векторы нигде не параллельны. Параметры u, v можно рассматривать как внутренние координаты точек поверхности. Фиксируя одну из координат, мы получаем два семейства координатных кривых, покрывающих поверхность координатной сеткой.

Случай явного задания. Поверхность S может быть определена как график функции z=f(x, y); тогда S является гладкой регулярной поверхностью, если функция f дифференцируема. Этот вариант можно рассматривать как частный случай параметрического задания: x=u; y=v; z=f(u, v).

## 1.2 Решение интегралов

Интеграл − одно из важнейших понятий математического анализа, которое возникает при решении задач о нахождении площади под кривой, пройденного пути при неравномерном движении, массы неоднородного тела, и тому подобных, а также в задаче о восстановлении функции по её производной (неопределённый интеграл) [2]. Упрощённо интеграл можно представить, как аналог суммы для бесконечного числа бесконечно малых слагаемых. В зависимости от пространства, на котором задана подынтегральная функция, интеграл может быть − двойной, тройной, криволинейный, поверхностный и так далее; также существуют разные подходы к определению интеграла − различают интегралы Римана, Лебега, Стилтьеса и другие [2].

Двойные интегралы − это обобщение понятия определённого интеграла для функции двух переменных, заданной как z = f (x, y).

Записывается двойной интеграл так:

Здесь D − плоская фигура, ограниченная линиями, выражения которых (равенства) даны в задании вычисления двойного интеграла. Слева и справа − равенствами, в которых слева переменная x, а сверху и снизу − равенствами, в которых слева переменная y. Это место и далее − одно из важнейших для понимания техники вычисления двойного интеграла.

Вычислить двойной интеграл - значит найти число, равное площади упомянутой фигуры D.

Существуют следующие методы решения интегралов: метод прямоугольников (левых, правых, средних), метод трапеции, метод Симпсона.

*1.2.1* **Метод прямоугольников** − метод [численного интегрирования](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) функции одной переменной, заключающийся в замене подынтегральной функции на многочлен нулевой степени, то есть константу, на каждом элементарном отрезке. Если рассмотреть график подынтегральной функции, то метод будет заключаться в приближённом вычислении площади под графиком суммированием площадей конечного числа прямоугольников, ширина которых будет определяться расстоянием между соответствующими соседними узлами интегрирования, а высота − значением подынтегральной функции в этих узлах. [Алгебраический порядок точности](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA_%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B0) равен 0. (Для формулы средних прямоугольников равен 1) [2].

Если отрезок {\displaystyle \left[a,b\right]}[a,b] является элементарным и не подвергается дальнейшему разбиению, значение интеграла можно найти по следующим формулам:

Левых прямоугольников:

{\displaystyle \int \_{a}^{b}f(x)\,dx\approx f(a)(b-a).}

Правых прямоугольников:

{\displaystyle \int \_{a}^{b}f(x)\,dx\approx f(b)(b-a).}

Средних прямоугольников:

{\displaystyle \int \_{a}^{b}f(x)\,dx\approx f\left({\frac {a+b}{2}}\right)(b-a).}

*1.2.2* **Метод трапеций** − метод [численного интегрирования](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) функции одной переменной, заключающийся в замене на каждом элементарном отрезке подынтегральной функции на многочлен первой степени, то есть линейную функцию. Площадь под графиком функции аппроксимируется прямоугольными [трапециями](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BF%D0%B5%D1%86%D0%B8%D1%8F). [Алгебраический порядок точности](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA_%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B0) равен 1.

Если отрезок {\displaystyle \left[a,b\right]}[a,b] является элементарным и не подвергается дальнейшему разбиению, значение интеграла можно найти по формуле:

Это простое применение формулы для площади трапеции − произведение полусуммы оснований, которыми в данном случае являются значения функции в крайних точках отрезка, на высоту (длину отрезка интегрирования).

*1.2.3* **Формула Симпсона** (также [**Ньютона**](https://ru.wikipedia.org/wiki/%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD,_%D0%98%D1%81%D0%B0%D0%B0%D0%BA)−**Симпсона**) относится к приёмам [численного интегрирования](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Суть метода заключается в приближении подынтегральной функции на отрезке {\displaystyle [a,b]}[a,b] [интерполяционным многочленом](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D1%8B) второй степени {\displaystyle p\_{2}(x)}, то есть приближение графика функции на отрезке параболой. Метод Симпсона имеет [порядок погрешности](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA_%D0%BF%D0%BE%D0%B3%D1%80%D0%B5%D1%88%D0%BD%D0%BE%D1%81%D1%82%D0%B8&action=edit&redlink=1) 4 и [алгебраический порядок точности](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA_%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B0) 3.

Формулой Симпсона называется интеграл от интерполяционного многочлена второй степени на отрезке [a, b]:

где f(a), f ((a + b) / 2) и f(b) − значения функции в соответствующих точках (на концах отрезка и в его середине).

Рассмотрим вопросы численного нахождения двойного интеграла по прямоугольной области [a.b] [c,d]:

Рассмотрим два способа вывода аналога формулы прямоугольников для численного решения данного интеграла (13).

*1.2.4* **Вывод с помощью одномерных интегралов:**

Так как мы уже знаем формулы прямоугольников для интегралов от функций одной переменной, то удобно представить двойной интеграл через два интеграла, каждый из которых будет вычисляться от функции одной переменной и может быть численно найден с помощью уже известной нам одномерной формулы прямоугольников. С этой целью введем вспомогательную функцию g(x) [3]:

Каждый из интегралов

можно вычислить с помощью формул численного интегрирования для одномерных интегралов.

Воспользуемся формулой прямоугольников (7) и начнем с На отрезке [c, d] введем равномерную сетку с шагом

Тогда для рассматриваемого интеграла (13) составная формула прямоугольников будет иметь вид:

Для численного нахождения интеграла по переменной x на отрезке [a, b] введем равномерную сетку с шагом :

на которой для интеграла получим формулу прямоугольников:

Объединяя формулы по каждому направлению, получим составную формулу прямоугольников для двойного интеграла [3]:

В данном курсовом проекте мы должны использовать метод Симпсона, проделов тоже самое мы получим следующую формулу:

, (20)

где

где f (x, y) – подынтегральная функция

Вычислив данный интеграл, мы получим площадь данной нам поверхности. Составим интеграл для вычисления площади поверхности:

В данном курсовом проекте мы имею следующую поверхности для интегрирования:

Вычислим производные для нашего интеграла из уравнения (22):

Полученные производные подставим в наш интеграл (22):

Вычислив данный интеграл методом Симпсона, который был описан выше, мы получим площадь поверхности.

## 1.3 Выбор языка программирования для разработки desktop приложения

Выбор будет производиться из двух языков: C# и Java.

*1.3.1* Язык Java разработан компанией Sun Microsystems, создателем которого был Джеймс Гослинг, и выпущен в 1995 году в качестве основных компонентов компании Sun Microsystems − Java платформ (Java 1.0 [J2SE]).

По состоянию на 2018 год последней версией Java Standard Edition является 10 (J2SE). С развитием Java, и её широкой популярностью, несколько конфигураций были построены для различных типов платформ. Например: J2EE − приложения для предприятий, J2ME − для мобильных приложений.

Sun Microsystems переименовала прежнюю версию J2 и ввела новые: Java SE, Java EE и Java ME. Введение в программирование Java различных версий подтверждало знаменитый слоган компании «Напиши один раз, запускай везде».

Язык программирования Java имеет следующий список достоинств:

— Объектно−ориентированный: в Java все является объектом. Дополнение может быть легко расширено, так как он основан на объектной модели.

— Платформонезависимый: в отличие от многих других языков, включая C и C++, Java, когда был создан, он не компилировался в платформе конкретной машины, а в независимом от платформы байт−коде. Этот байт код распространяется через интернет и интерпретируется в Java Virtual Machine (JVM), на которой он в настоящее время работает.

— Простой: процессы изучения и введение в язык программирования Java остаются простыми. Если Вы понимаете основные концепции объектно-ориентированного программирования, то он будет прост для Вас в освоении.

— Безопасный: методы проверки подлинности основаны на шифровании с открытым ключом.

— Архитектурно-нейтральный: компилятор генерирует архитектурно-нейтральные объекты формата файла, что делает скомпилированный код исполняемым на многих процессорах, с наличием системе Java Runtime.

— Портативный: архитектурно-нейтральный и не имеющий зависимости от реализации аспектов спецификаций − все это делает Java портативным. Компилятор в Java написан на ANSI C с чистой переносимостью, который является подмножеством POSIX.

— Прочный: выполняет усилия, чтобы устранить ошибки в различных ситуациях, делая упор в основном на время компиляции, проверку ошибок и проверку во время выполнения.

— Многопоточный: функции многопоточности, можно писать программы, которые могут выполнять множество задач одновременно. Введение в язык Java этой конструктивной особенности позволяет разработчикам создавать отлаженные интерактивные приложения.

— Интерпретированный: Java байт−код переводится на лету в машинные инструкции и нигде не сохраняется. Делая процесс более быстрым и аналитическим, поскольку связывание происходит как дополнительное с небольшого весом процесса.

— Высокопроизводительный: введение Just-In-Time компилятора, позволило получить высокую производительность.

— Распространенный: предназначен для распределенной среды интернета.

— Динамический: программирование на Java считается более динамичным, чем на C или C++, так как он предназначен для адаптации к меняющимся условиям. Программы могут выполнять обширное количество во время обработки информации, которая может быть использована для проверки и разрешения доступа к объектам на время выполнения.

*1.3.2* C# − элегантный, типобезопасный объектно−ориентированный язык, предназначенный для разработки разнообразных безопасных и мощных приложений, выполняемых в среде .NET Framework. С помощью языка C# можно создавать обычные приложения Windows, приложения "клиент-сервер", приложения баз данных и т. д. Visual C# предоставляет развитый редактор кода, конструкторы с удобным пользовательским интерфейсом, встроенный отладчик и множество других средств, упрощающих разработку приложений на базе языка C# и .NET Framework.

Синтаксис C# очень выразителен, но прост в изучении. Все, кто знаком с языками C, C++ или Java с легкостью узнают синтаксис с фигурными скобками, характерный для языка C#. Разработчики, знающие любой из этих языков, как правило, смогут добиться эффективной работы с языком C# за очень короткое время. Синтаксис C# делает проще то, что было сложно в C++, и обеспечивает мощные возможности, такие как типы значений Nullable, перечисления, делегаты, лямбда-выражения и прямой доступ к памяти, чего нет в Java. C# поддерживает универсальные методы и типы, обеспечивая более высокий уровень безопасности и производительности, а также итераторы, позволяющие при реализации коллекций классов определять собственное поведение итерации, которое может легко использоваться в клиентском коде. Выражения LINQ делают строго типизированный запрос очень удобной языковой конструкцией [4].

Как объектно−ориентированный язык, C# поддерживает понятия инкапсуляции, наследования и полиморфизма. Все переменные и методы, включая метод Main − точку входа приложения − инкапсулируются в определения классов. Класс может наследовать непосредственно из одного родительного класса, но может реализовывать любое число интерфейсов. Для методов, которые переопределяют виртуальные методы в родительском классе, необходимо ключевое слово override, чтобы исключить случайное повторное определение. В языке C# структура похожа на облегченный класс: это тип, распределяемый в стеке, реализующий интерфейсы, но не поддерживающий наследование.

В дополнение к основным описанным объектно-ориентированным принципам, язык C# упрощает разработку компонентов программного обеспечения благодаря нескольким инновационным конструкциям языка, в число которых входят следующие:

* инкапсулированные сигнатуры методов, называемые делегатами, которые поддерживают типобезопасные уведомления о событиях;
* свойства, выступающие в роли методов доступа для закрытых переменных−членов;
* LINQ, предлагающий встроенные возможности запросов в различных источниках данных.

Если потребуется обеспечить взаимодействие с другим программным обеспечением Windows, таким как объекты COM или собственные библиотеки DLL Win32, в языке C# можно использовать процесс, который называется "Interop." Процесс Interop позволяет программам на C# выполнять практически любые действия, которые может выполнять исходное приложение на C++. Язык C# поддерживает даже указатели и понятие "небезопасного" кода для тех случаев, когда прямой доступ к памяти имеет крайне важное значение.

Процесс построения C# по сравнению с C и C++ прост и является более гибким, чем в Java. Нет отдельных файлов заголовка, а методы и типы не требуется объявлять в определенном порядке. В исходном файле C# может быть определено любое число классов, структур, интерфейсов и событий.

Таким образом остановимся на языке C#.

## 1.4 Построение поверхности

Для построения поверхности нужно: выбрать язык программирования, выбрать инструмент для построения. С языком программирования мы определились, теперь нужно выбрать инструмент для построения поверхности. При рассмотрении этой задачи будем опираться на простоту и эффективность.

*1.4.1* Первый язык для рассмотрения – это C#, так как на нем будет написан пользовательский интерфейс, то его будет рассматривать первым для построения поверхности при помощи такого инструмента как OpenGL.

Как известно, OpenGL расшифровывается как Open Graphics Library, что в переводе на русский язык означает «открытая графическая библиотека». Другими словами, OpenGL − это некая спецификация, включающая в себя несколько сотен функций. Она определяет независимый от языка программирования кросс−платформенный программный интерфейс, с помощью которого программист может создавать приложения, использующие двухмерную и трехмерную компьютерную графику. Первая базовая версия OpenGL появилась в 1992 году – она была разработана компанией Silicon Graphics Inc., занимающейся разработками в области трехмерной компьютерной графики [5].

За следующие 17 лет версия библиотеки достигла отметки 3.1. Но нельзя сказать, что библиотека видоизменялась, нет. Ее создатели заложили в нее механизм расширений, благодаря которому производители аппаратного обеспечения (к примеру, производители видеокарт) могли выпускать расширения OpenGL для поддержки новых специфических возможностей, не включенных в текущую версию библиотеки. Благодаря этому, программисты могли сразу использовать эти новые возможности, в то время как в случае использования библиотеки Microsoft Direct3D им бы пришлось ждать выхода новой версии DirectX.

Tao Framework − это свободно-распространяемая библиотека с открытым исходным кодом, предназначенная для быстрой и удобной разработки кросс−платформенного мультимедийного программного обеспечения в среде .NET Framewrok и Mono. На сегодняшний день Tao Framework − это лучший путь для использования библиотеки OpenGL при разработке в среде .NET на языке C#.

В состав библиотеки на данный момент входят все современные средства, которые могут понадобиться в ходе разработки мультимедиа программного обеспечения: реализация библиотеки OpenGL, реализация библиотеки FreeGlut, содержащей все самые новые функции этой библиотеки, библиотека DevIL (легшая в основу стандарта OpenIL − Open Image Library) и многое другое.

Самые интересные библиотеки, включенные в Tao Framework:

* OpenGL 2.1.0.12 − свободно распространяемый аппаратно-программный интерфейс для визуализации 2D и 3D графики.
* FreeGLUT 2.4.0.2 − библиотека с открытым исходным кодом, являющаяся альтернативой библиотеке GLUT (OpenGL Utility Toolkit).
* DevIL 1.6.8.3 (она же OpenIL) − кроссплатформенная библиотека, реализующая программный интерфейс для работы с изображениями. На данный момент библиотека поддерживает работу с изображениями 43-х форматов для чтения и 17-ти форматов для записи.
* Cg 2.0.0.0 − язык высокого уровня, созданный для программирования текстурных и вершинных шейдеров.
* OpenAL 1.1.0.1 − свободно распространяемый аппаратно-программный интерфейс для обработки аудиоданных (в том числе 3D звука и EAX-эффектов).
* PhysFS 1.0.1.2 − библиотека для работы с вводом / выводом файловой системы, а также различного вида архивами, на основе собственного API. SDL 1.2.13.0 − кроссплатформенная мультимедийная библиотека, активно использующаяся для написания мультимедийных приложений в операционной системе.
* GNU/Linux ODE 0.9.0.0 − свободно распространяемый физический программный интерфейс, главной особенностью которого является реализация системы динамики абсолютно твёрдого тела и система обнаружения столкновений.
* FreeType 2.3.5.0 − библиотека, реализующая растеризацию шрифтов. Данная библиотека используется в X11 (оконной системе, которая обеспечивает все стандартные инструменты и протоколы для построения GUI (графического интерфейса пользователя) в UNIX подобных операционных системах).
* FFmpeg 0.4.9.0 − набор свободно распространяемых библиотек с открытым исходным кодом. Данные мультимедийные библиотеки позволяют работать аудио и видео данными в различных форматах.

Библиотека Tao Framework является мощным свободно распространяемым инструментом для решения любых мультимедийных задач преимущественно кроссплатформенного характера.

Работая с данной библиотекой, разработчик или группа разработчиков могут использовать базу алгоритмов и реализованных за многие годы методов, что в десятки, а то и сотни раз сокращает время разработки программных продуктов.

После такого описания данного фреймворка, кажется, что построить поверхность не составит проблем. Но все оказалось куда сложнее, чем написано. Задача оказалась нетривиальной, поэтому продолжим поиск.

*1.4.2* Следующим языком для рассмотрения оказался Java. Библиотек для построения я нашел всего пару, да и то, которые работаюn на JavaFX (Написание desktop приложение на JavaFX очень сложная задача, так как порог вхождения в эту технология очень велик, что не подходит для нас).

*1.4.3* Остался один язык из тех, на которых я мог выполнить данную задачу, и это Python.

Python −  [высокоуровневый язык программирования](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. [Синтаксис](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) ядра Python минималистичен. В то же время [стандартная библиотека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_Python) включает большой объём полезных функций [6].

Основные архитектурные черты − [динамическая типизация](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [автоматическое управление памятью](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), полная [интроспекция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), механизм [обработки исключений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), поддержка [многопоточных вычислений](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C), высокоуровневые [структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Поддерживается разбиение программ на [модули](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые, в свою очередь, могут объединяться в пакеты.

Построения поверхности в этом языке программирования самая тривиальная задача, которая может быть. Для ее выполнения нужно следующие:

* в Matplotlib версии 0.99 наконец-то появилась возможность рисовать кроме двумерных также и трехмерные графики. Для построения поверхности нам понадобится версия Matplotlib не ниже 0.99.
* кроме того, нам понадобится математическая библиотека numpy, так как некоторые методы классов рисования трехмерных графиков в качестве параметров ожидают экземпляры класса numpy.array, да и сама библиотека Numpy позволяет значительно сократить количество строк кода.

Давайте рассмотри самый простой пример:

Возьмем следующий пример:

Для начала нужно подготовить данные для рисования. Нам понадобятся три двумерные матрицы: матрицы *X* и *Y* будут хранить координаты сетки точек, в которых будет вычисляться приведенная выше функция, а матрица *Z* будет хранить значения этой функции (25) в соответствующей точке.

Если мы хотим нарисовать трехмерный график на эквидистантной сетке (на сетке, у которой расстояние между точками одинаковое), то для создания матриц, которые будут хранить координаты, поможет функция *meshgrid()* из библиотеки *numpy*. Эта функция создает двумерные матрицы сеток по одномерным массивам. Работа этой функции очень наглядно показана в документации к *numpy*.

Чтобы отделить подготовку данных от самого рисования, создание сетки и расчет функции выделим в отдельную функцию:

def makeData ():

# Строим сетку в интервале от -10 до 10 с шагом 0.1 по обоим координатам

x = numpy.arange (-10, 10, 0.1)

y = numpy.arange (-10, 10, 0.1)

# Создаем двумерную матрицу-сетку

xgrid, ygrid = numpy.meshgrid(x, y)

# В узлах рассчитываем значение функции

zgrid = numpy.sin (xgrid) \* numpy.sin (ygrid) / (xgrid \* ygrid)

return xgrid, ygrid, zgrid

Эта функция возвращает три двумерные матрицы: x, y, z. Координаты x и y лежат в интервале от -10 до 10 с шагом 0.1.

Теперь возвращаемся непосредственно к рисованию. Чтобы отобразить наши данные, достаточно вызвать метод plot\_surface() экземпляра класса Axes3D, в который передадим полученные с помощью функции makeData() двумерные матрицы.

Теперь наш пример выглядит следующим образом:

**import** pylab

**from** mpl\_toolkits.mplot3d **import** Axes3D

**import** numpy

**def** makeData ():

x = numpy.arange (**-10**, **10**, **0.1**)

y = numpy.arange (**-10**, **10**, **0.1**)

xgrid, ygrid = numpy.meshgrid(x, y)

zgrid = numpy.sin (xgrid) \* numpy.sin (ygrid) / (xgrid \* ygrid)

**return** xgrid, ygrid, zgrid

x, y, z = makeData()

fig = pylab.figure()

axes = Axes3D(fig)

axes.plot\_surface(x, y, z)

pylab.show()

Если мы запустим этот скрипт, то появится окно со следующей синей поверхностью:

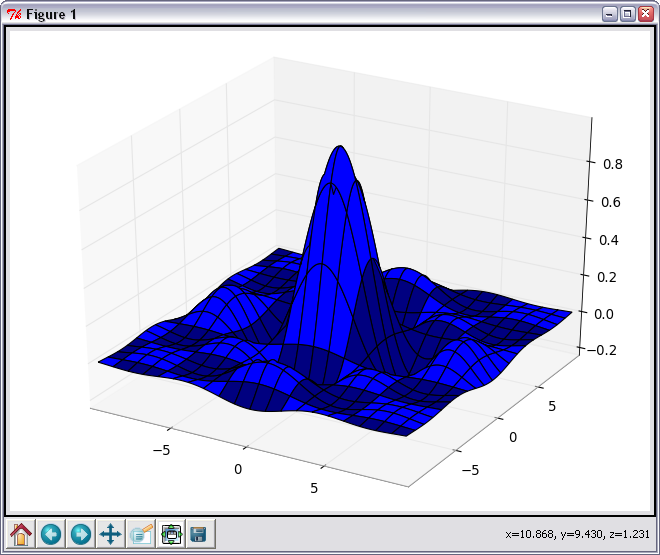


Рисунок 1 – График поверхности, построенный в Python

Отсюда сделаем вывод, что мы будем использовать скрипт Python для построения поверхности, так как он прост и предоставляют стандартный шаблон, что очень удобно.

## 1.5. Математические пакеты

SciLab – пакет прикладных [математических](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) программ, предоставляющий открытое окружение для инженерных (технических) и научных расчётов. Это самая полная общедоступная альтернатива [MATLAB](https://ru.wikipedia.org/wiki/MATLAB).

Scilab содержит сотни математических функций, и есть возможность добавления новых, написанных на различных языках (C, C++, Fortran и т. д.). Также имеются разнообразные структуры данных (списки, полиномы, рациональные функции, линейные системы), интерпретатор и язык высокого уровня.

Scilab был спроектирован как открытая система, и пользователи могут добавлять в него свои типы данных и операции путём перегрузки.

В системе доступно множество инструментов:

* 2D и 3D графики, анимация;
* линейная алгебра, разреженные матрицы (sparse matrices);
* полиномиальные и рациональные функции;
* интерполяция, аппроксимация;
* симуляция: решение ОДУ и ДУ;
* scicos: гибрид системы моделирования динамических систем и симуляции;
* дифференциальные и не дифференциальные оптимизации;
* обработка сигналов;
* параллельная работа;
* статистика;
* работа с компьютерной алгеброй;
* интерфейс к Fortran, Tcl/Tk, C, C++, Java, LabVIEW.

Scilab имеет схожий с MATLAB язык программирования. В состав пакета входит утилита, позволяющая конвертировать документы Matlab в Scilab.

Scilab позволяет работать с элементарным и большим числом специальных функций (Бесселя, Неймана, интегральные функции), имеет мощные средства работы с матрицами, полиномами (в том числе и символьно), производить численные вычисления (например, численное интегрирование) и решение задач линейной алгебры, оптимизации и симуляции, мощные статистические функции, а также средство для построения и работы с графиками.

Программы в Scilab пишутся в виде файлов-сценариев. Файл-сценарий -это список команд Scilab, сохраненный на диске. Ввод сценария программы может происходить несколькими способами. В небольших сценариях обычно ввод данных производится через командную строку. Однако, с ростом сложности проекта такой ввод становится неудобным и недружественным, т.е. пользователь должен знать какие данные и каких типов требует от него сценарий. Другой способ, который чаще всего используется в купе с графическим интерфейсом пользователя, является ввод через графическое окно, специальный редактор SciNotes(в более ранних версиях он назывался SciPad). Окно редактора файлов-сценариев выглядит стандартно, т.е. имеет заголовок, меню, строку состояния.

Mathcad — система компьютерной алгебры из класса систем автоматизированного проектирования, ориентированная на подготовку интерактивных документов с вычислениями и визуальным сопровождением, отличается лёгкостью использования и применения для коллективной работы.

Mathcad был задуман и первоначально написан Алленом Раздовом из Массачусетского технологического института (MIT), соучредителем компании Mathsoft, которая с 2006 года является частью корпорации PTC (Parametric Technology Corporation).

Mathcad имеет интуитивный и простой для использования интерфейс пользователя. Для ввода формул и данных можно использовать как клавиатуру, так и специальные панели инструментов.

Некоторые из математических возможностей Mathcad (версии до 13.1 включительно) основаны на подмножестве системы компьютерной алгебры Maple (MKM, Maple Kernel Mathsoft). Начиная с 14 версии — использует символьное ядро MuPAD.

Работа осуществляется в пределах рабочего листа, на котором уравнения и выражения отображаются графически, в противовес текстовой записи в языках программирования. При создании документов-приложений используется принцип WYSIWYG (What You See Is What You Get — «что видишь, то и получаешь»).

Несмотря на то, что эта программа, в основном, ориентирована на пользователей, не являющихся программистами, Mathcad также используется в сложных проектах, чтобы визуализировать результаты математического моделирования путём использования распределённых вычислений и традиционных языков программирования. Также Mathcad часто используется в крупных инженерных проектах, где большое значение имеет трассируемость и соответствие стандартам.

Mathcad достаточно удобно использовать для обучения, вычислений и инженерных расчетов. Открытая архитектура приложения в сочетании с поддержкой технологий .NET и XML позволяют легко интегрировать Mathcad практически в любые ИТ-структуры и инженерные приложения. Есть возможность создания электронных книг (e-Book).

Mathcad содержит сотни операторов и встроенных функций для решения различных технических задач. Программа позволяет выполнять численные и символьные вычисления, производить операции со скалярными величинами, векторами и матрицами, автоматически переводить одни единицы измерения в другие.

Среди возможностей Mathcad можно выделить:

* решение дифференциальных уравнений, в том числе и численными методами
* построение двумерных и трёхмерных графиков функций (в разных системах координат, контурные, векторные и т. д.)
* использование греческого алфавита как в уравнениях, так и в тексте
* выполнение вычислений в символьном режиме
* выполнение операций с векторами и матрицами
* символьное решение систем уравнений
* аппроксимация кривых
* выполнение подпрограмм
* поиск корней многочленов и функций
* проведение статистических расчётов и работа с распределением вероятностей
* поиск собственных чисел и векторов
* вычисления с единицами измерения
* интеграция с САПР-системами, использование результатов вычислений в качестве управляющих параметров
* с помощью Mathcad инженеры могут документировать все вычисления в процессе их проведения.

# 2. Разработка функциональной модели приложения

## 2.1 Постановка задачи

Необходимо разработать распределенное компонентное приложение подготовки данных для обработки и отображения поверхности.

Методом численного интегрирования выбрать метод Симпсона.

Вид поверхности:

Для данной поверхности Xn, Xk, Yn, Yk принять -100, 100, -20, 20 соответственно.

В программе предусмотреть возможность отображения поверхности, расчет площади заданной поверхности в заданных пределах, анализ влияния распараллеливания выполнения расчета площади на время.

Проверить результат нахождения площади и построения поверхности в математическом пакете, используя стандартные функции, и сравнить результаты.

## 2.2 Построение функциональной модели

Для выполнения поставленной задачи, программа должна реализовывать следующий функционал:

* расчет площади заданной поверхности заданным численным методом (метод Симпсона)
* построение графика поверхности
* анализ влияния распараллеливания вычисления площади поверхности заданным численным методом

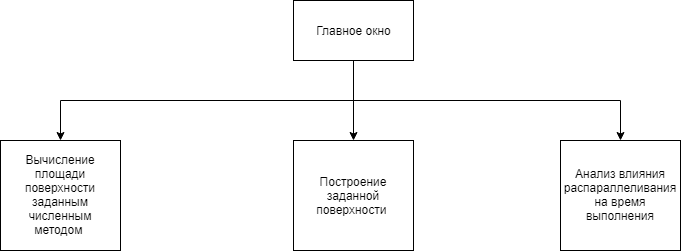


Рисунок 2.1 – Структурная схема программы

Приложение представляет собой комплекс модулей для расчета площади, отображения поверхности и анализа влияния распараллеливания подсчета площади. При этом под анализом будет пониматься наглядное сравнение погрешности вычисления и затраченного времени на выполнение задачи.

Модули комплексного приложения для решения задачи (Приложение А):

* главный модуль
* математический модуль
* графический модуль

Главный модуль − главное приложение WinForms, в котором будет осуществляться управление комплексом, отображение результатов вычислений и анализа.

Математический модуль − класс, в котором находятся функции для вычисления площади заданной поверхности по варианту.

Графический модуль − python приложение, которое читает координаты из сформированного файла и строит по ним графическое отображение поверхности.

# 3 Разработка компонентов

## 3.1 Разработка программного кода

Программа состоит из трех модулей: CourseWork (рис. 3.1), MathFunctions (рис. 3.1), grafik.exe.

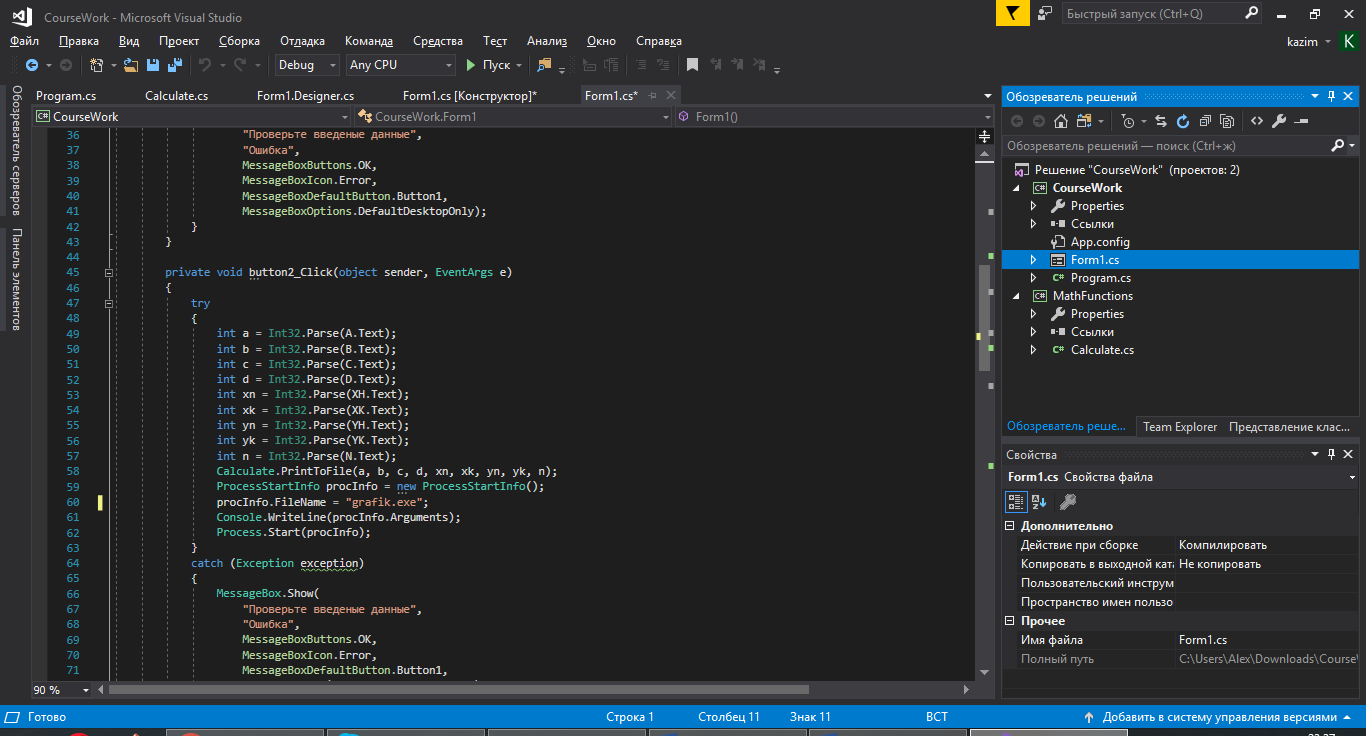


Рисунок 3.1 – Структура проекта CourseWork

Проект test (рис. 3.2) — приложение на языке python, которое компилируется в приложение grafik.exe. Оно читает массив координат из файла «cordinats.txt», где записаны все точки по x, y, z и по ним создает поверхность. Код этого компонента представлен в приложение B.

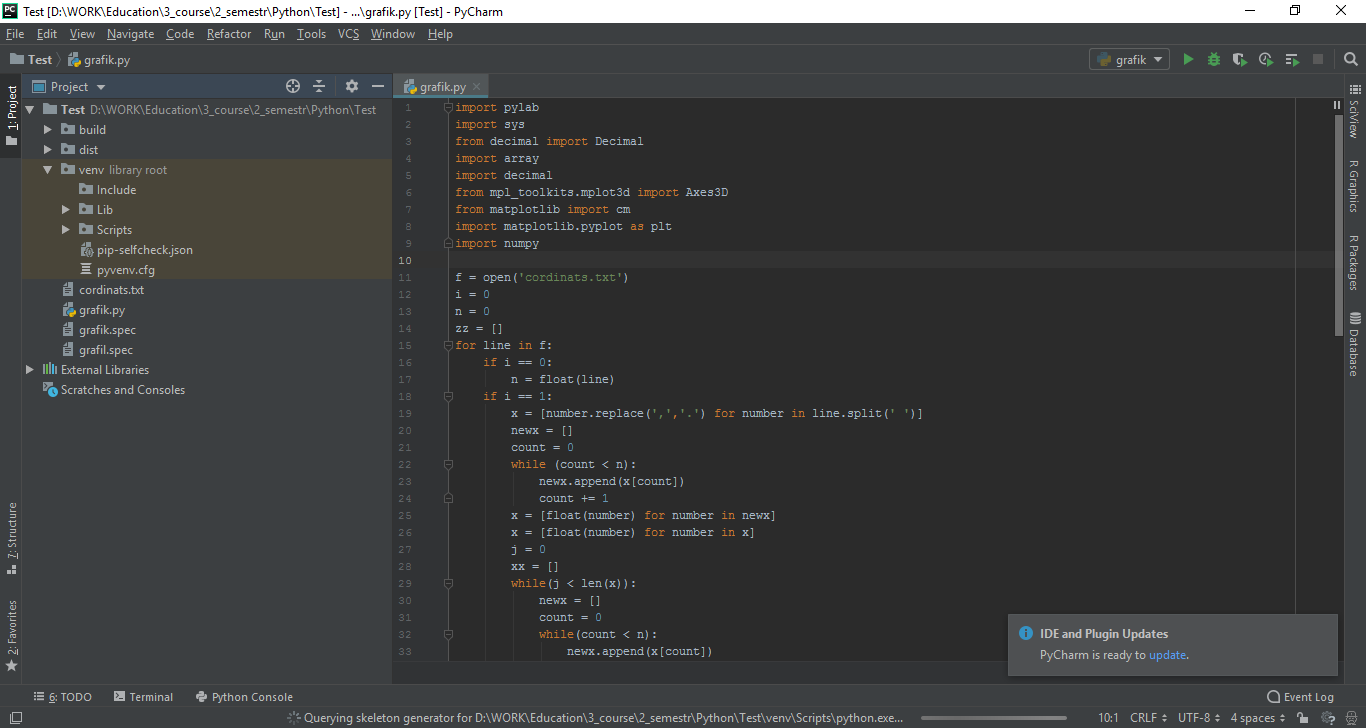


Рисунок 3.2 – Структура проекта Test

Проект MathFunctions − модуль, отвечающий за математику приложения. Он рассчитывает двойной интеграл методом Симпсона. Код этого компонента представлен в приложение B.

Содержит файл Calculate.cs. Он содержит следующие функции:

* double f(double x, double y) − возвращает значение функции в f в точках x и y.
* double g(double x, double hy, double n, double a) − функция, которая рассчитывает первый интеграл, где x − значение х, для функции f(x,y), hy − шаг y, n − количество точек для расчета, a − начальное значение для y.
* double SurfaceArea(double xn, double xk, double yn, double, yk, double n) − функция, которая рассчитывает второй интеграл, где xn − начальное значение x, xk − конечное значение х, yn − начальное значение у, yk − конечное значение у, n − количество точек для расчета.
* void PrintToFile(double a, double b, double c, double d, double xn, double xk, double yn, double yk, double n) — функция, которая создает файл с координатами поверхности для скрипта Python.

Проект CourseWork − главный, собирающий модуль. Он имеет 2 файла: Form1.cs и Program.cs.

Файл Form1.cs − содержит методы управления графической формой:

a) void button1\_Click(object sender, EventArgs e) − обработчик нажатия кнопки на форме. Вызывает метод для вычисления площади и отдает результат в label.

b) void button2\_Click(object sender, EventArgs e) − обработчик нажатия кнопки на форме. Пишет координаты для построения плоскости в файлы “cordinats.txt”, затем вызывает приложение “grafik.exe” для построения плоскости.

c) async void button3\_Click(object sender, EventArgs e) − обработчик нажатия кнопки на форме. Запускает анализ влияния распределения потоков на время и точность выполнения задачи.

d) async Task<(double,TimeSpan)> Analysis(int number,int n, double xn, double xk, double yn, double yk) – функция для разбиения задачи на потоки и подсчета времени выполнения. Возвращает кортеж, где первый параметр − результат подсчета площади, второй − затраченное время.

## 3**.2 Разработка графического интерфейса**

В приложении присутствует два окна. Первое − построенная плоскость:

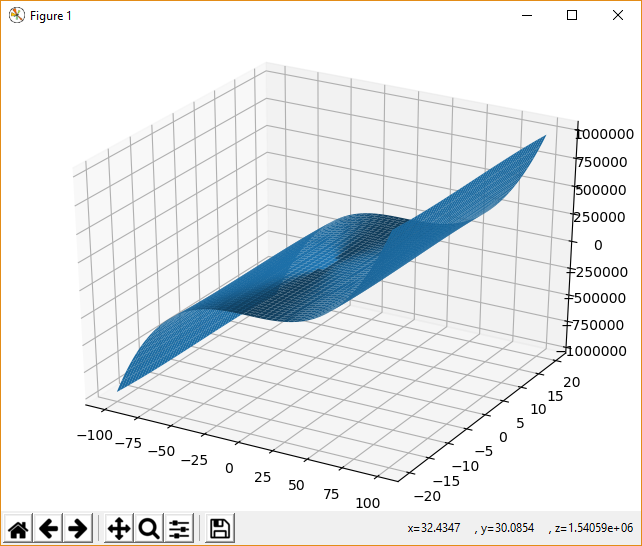


Рисунок 3.3 – Построенная плоскость

Второе окно − главная форма, где происходит управление приложением:

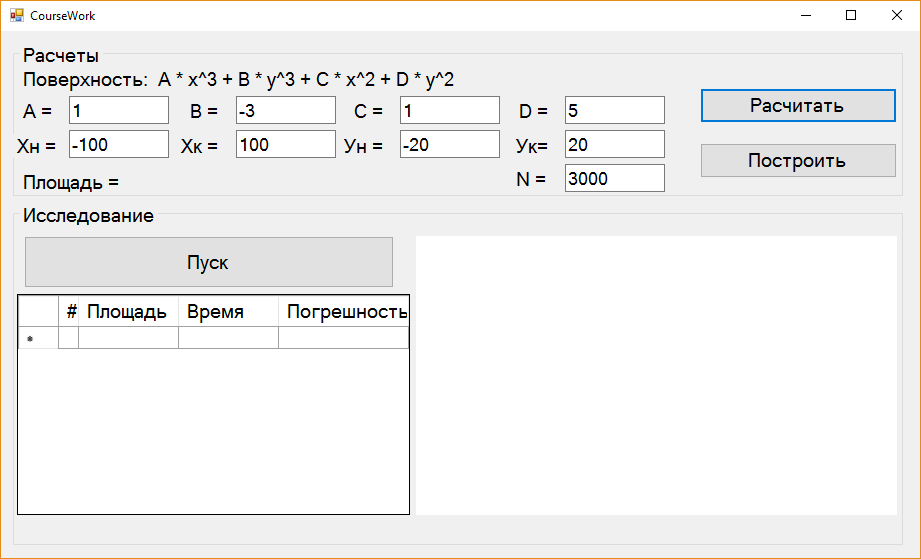


Рисунок 3.4 – Главное окно

## 3.3 Определение погрешности численного метода

Для определения погрешности вычислений численного метода интегрирования (метод Симпсона), заранее было вычислено значение площади поверхности в математическом пакете и внесено в память программы:

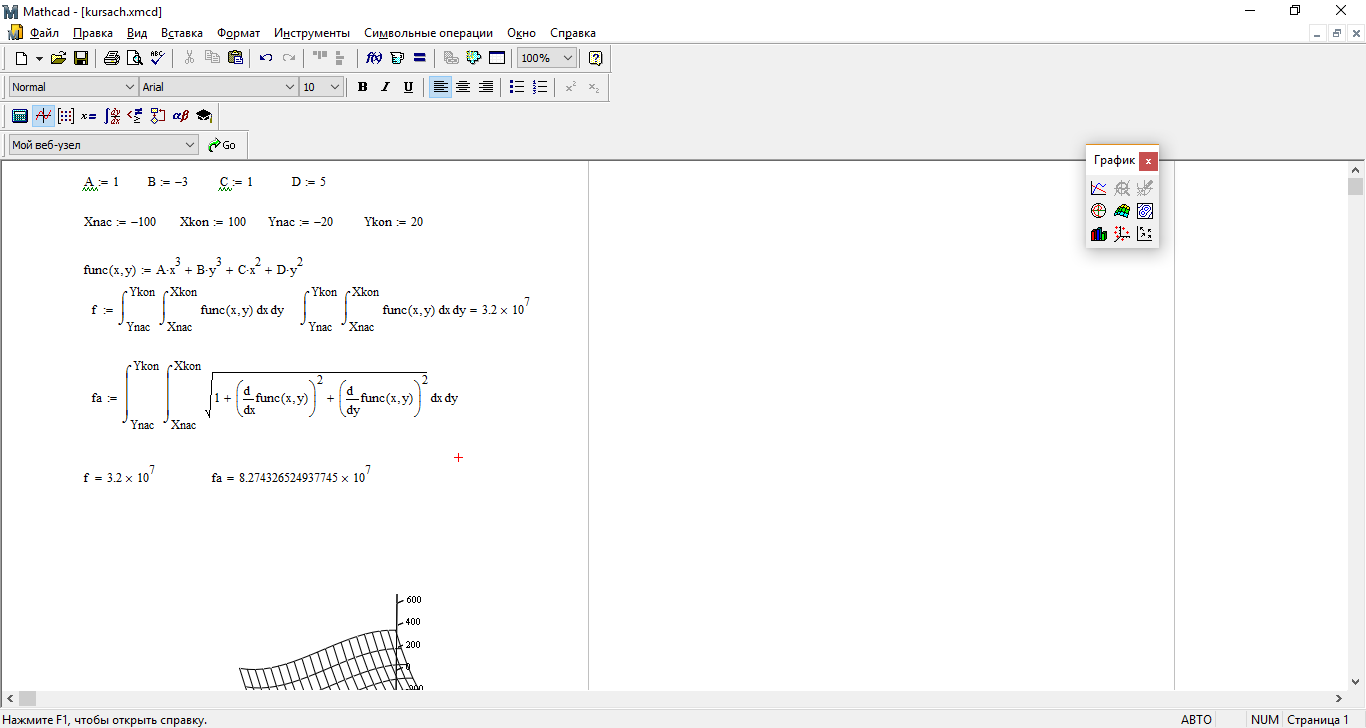


Рисунок 3.5 – Вычисление в математическом пакете

Далее от полученного в математическом пакете значения отнимается полученное численным методом в приложении значение:

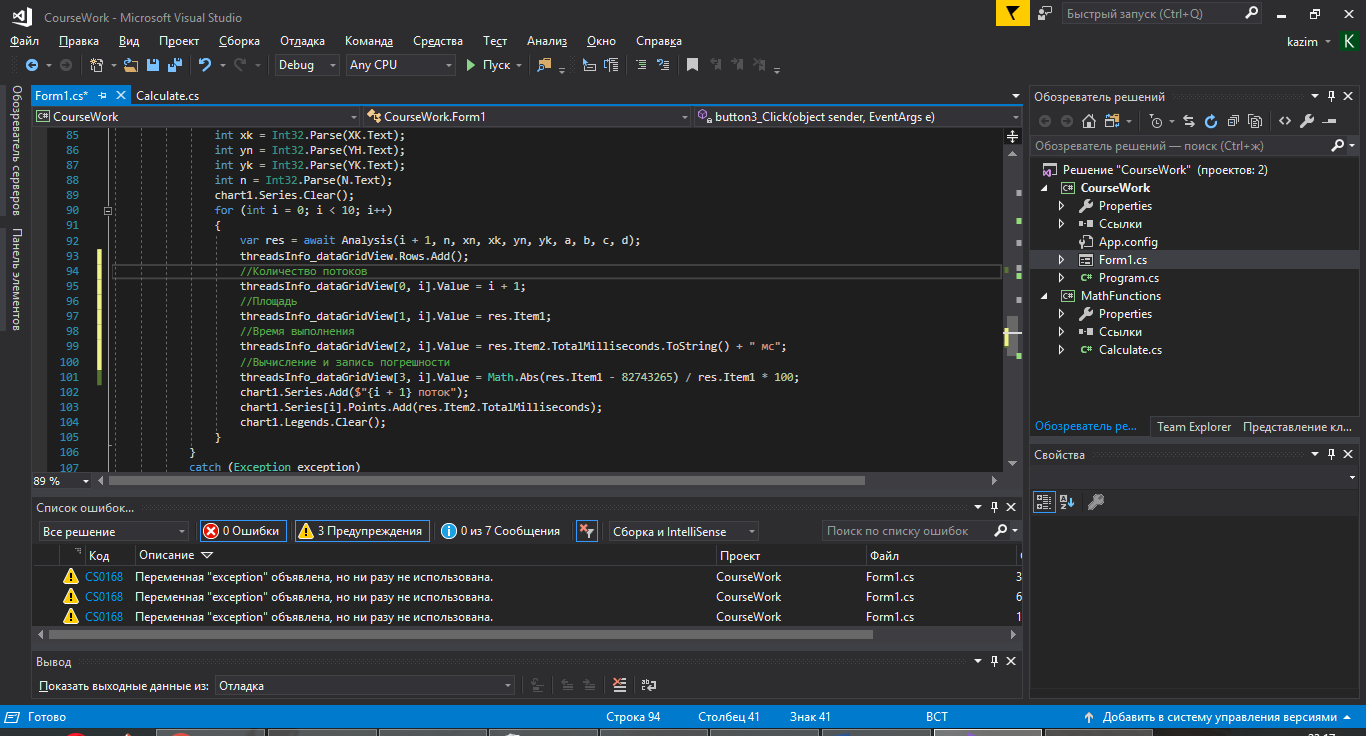


Рисунок 3.6 – Вычисление погрешности

# 4 Исследование эффективности реализации многопоточности

Для реализации данной задачи были использованы стандартные механизмы работы с многопоточностью в C#.

Чтобы получить данные из потока был использован класс Task, которые позволяет после выполнения потока вернуть результат. Для их хранения использовалась обычная коллекция List, в которую записывались все выполняемые потоки. После цикл проходил по коллекции и складывал данные, которые хранились в Task. Таким образом считались площади. Задачу поочередно выполняли один, два, три, четыре и пять потоков и так до десяти. После каждого выполнения записывался результат и затраченное время и выводилось в диаграмму:

Исходя из результатов (рис. 4.1) видно, что один поток имеет наименьшую погрешность, но тратит очень много времени на выполнение задачи. А задача, выполненная на десяти потоках, имеет минимальное время выполнения, но наибольшую погрешность.

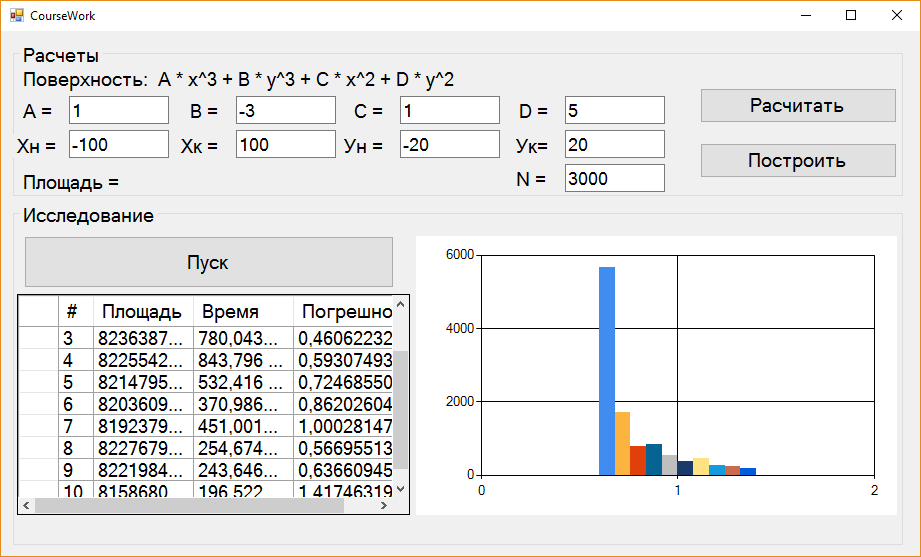


Рисунок 4.1 – исследование эффективности многопоточности

Таким образом, чем большее количество потоков применено при решении задачи, тем большую погрешность мы получаем.

# 5 Разработка графического интерфейса

Для построение графического интерфейса в курсовом проекте использовался Windows Forms.

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

Приложение Windows Forms представляет собой событийно-ориентированное приложение, поддерживаемое Microsoft .NET Framework. В отличие от пакетных программ, большая часть времени тратится на ожидание от пользователя каких-либо действий, как, например, ввод текста в текстовое поле или клика мышкой по кнопке.

В графическом интерфейсе используются такие элементы как (рис 5.1):

* TextBox – элемент графического интерфейса, который используется для получения входных данных от пользователя или для отображения текста.
* Button – элемент графического интерфейса, представляющий собой кнопку.
* Label – элемент графического интерфейса для вывода статической информации.
* Chart – элемент графического интерфейса для создания графика.
* DataGridView – элемент графического интерфейса, предоставляющий мощный и гибкий способ отображения данных в табличном формате.

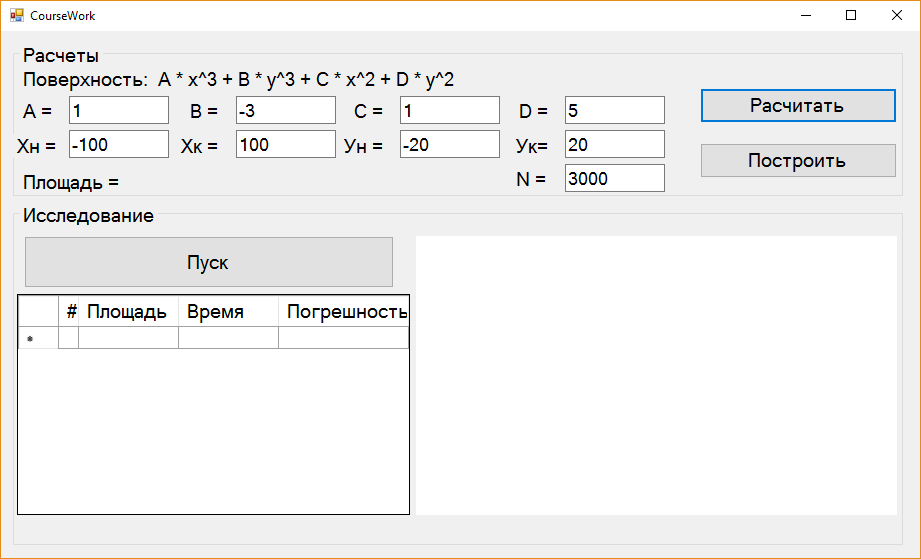


Рисунок 5.1 – Приложение Windows Forms

6 Тестирование и верификация

В ходе разработки данного приложения были найдены и устранены следующие ошибки:

* не введен один из параметров (рис 6.1);
* неверно введен один из параметров (рис 6.1);
* отсутствие python файла (рис 6.2);

Данные ошибки найдены и устранены

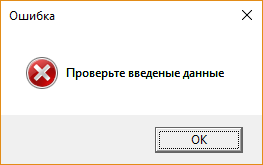


Рисунок 6.1 – Ошибка заполнения начальных условий

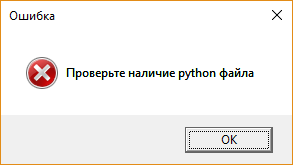


Рисунок 6.2 – Отсутствие скрипта python

# 7 Руководство пользователя

Для запуска и использования программы компьютер должен обладать представленными ниже техническими требованиями.

Минимальные технические требования – ОС Windows XP/Vista. Процессор Intel Pentium IV 1.2 ГГц. 128 Мб ОЗУ. 16 Мб места на жёстком диске, клавиатура и мышь.

Рекомендуемые технические требования – ОС Windows 7/8/8.1/10. Процессор Intel Pentium B950 2.1 ГГц и лучше. 2 Гб ОЗУ, 256 Мб места на жёстком диске, клавиатура и мышь.

Перед запуском приложения убедитесь, что присутствуют следующие файлы:

* CourseWork.exe
* grafik.exe

Для запуска приложения кликнете дважды левой кнопкой мыши по файлу grafik.exe. После запуска приложения пользователю предоставляется возможность ввода исходных данных. Если пользователь хочет высчитать площадь поверхности, то нажав на кнопку «Рассчитать» программа выдаст результат на экран (рис 7.1).

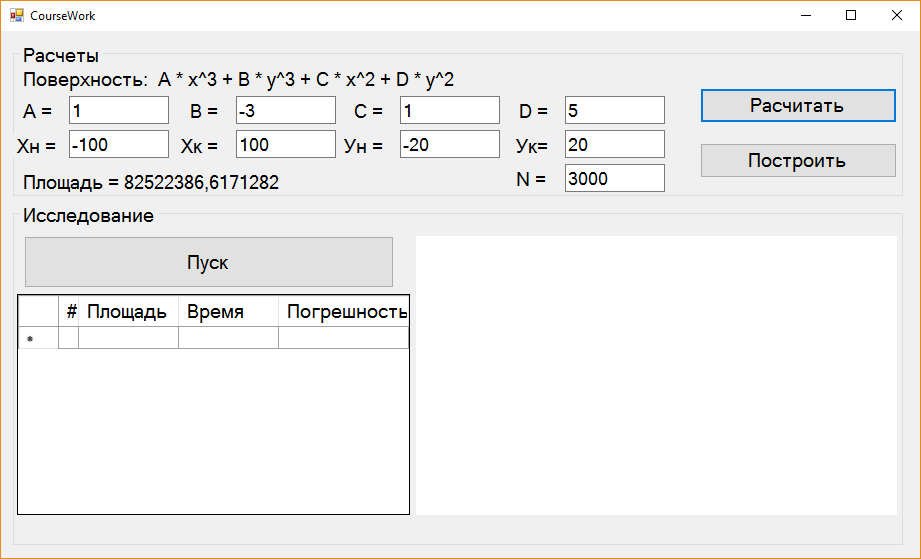


Рисунок 7.1 – Вычисление площади поверхности

Если пользователь хочет построить поверхность с заданными начальными условиями, то ему нужно нажать на кнопку «Построить», после чего запустить скрипт Python, который построит поверхность.

Если пользователь исследовать эффективность реализации многопоточности с заданными начальными условиями, то нужно нажать на кнопку «Пуск», программа сделает расчёты и выдаст их на экран (рис 7.2).

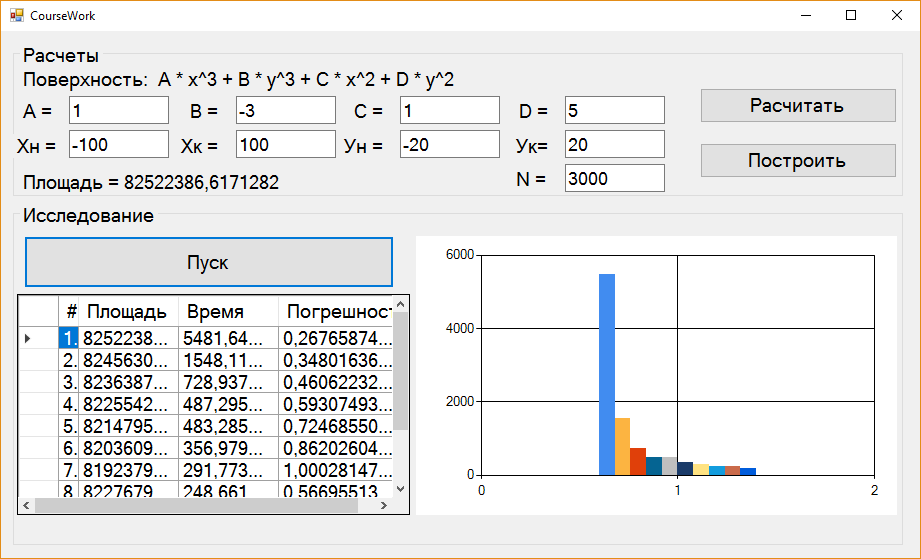


Рисунок 7.2 - Исследование эффективности реализации многопоточности

# Заключение

В результате выполнения данного курсового проекта было разработано приложение, предназначенное для вычисления площади заданной поверхности в заданных пределах, построен график данной поверхности и исследование влияния распределения вычисления площади на время выполнения и точность результатов. Были проведены такие этапы разработки, как изучение теоретического материала, алгоритмический анализ задачи и составление алгоритма работы приложения.

В течение выполнения данного курсового проекта был изучен теоретический материал о численном методе Симпсона для вычисления интеграла, на основе которого в дальнейшем были разработаны алгоритмы решения поставленной задачи.

После получения вышеуказанных сведений был разработан указанный программный продукт. При его использовании конечный пользователь может вычислить площадь данной по варианту поверхности, построить график этой поверхности и провести анализ влияния распределения вычисления площади поверхности на время выполнения и точность. В итоге был приобретен опыт разработки подобных приложений с помощью языка программирования C# и Python. Приложение было разработано в среде разработки Microsoft Visual Studio Community 2017.

Для проверки правильности полученного решения в разработанной программе был использован бесплатный математический пакет SMath.com. Проверка осуществлялась на основе заданной поверхности и была пройдена успешно, что свидетельствует о корректной работе программы.

Поставленные задачи в курсовом проекте были решены полностью.

# Список использованных источников

1. Ильин В. А., Позняк Э. Г. Аналитическая геометрия. — М.: ФИЗМАТЛИТ, 2002. — 240 с.

2. Калиткин, Н.Н. Численные методы / Н.Н. Калиткин. – СПб.:BHV, 2014. – 202 с.

3. Основные понятия. Численные методы линейной алгебры и анализа [Электронные ресурс].: Свободная энциклопедия. – Электронные данные. – Режим доступа: https://slemeshevsky.github.io/python-num-pde/term1/build/html/\_computing-integrals/double-triple.html – Дата доступа: 15.04.2019.

4. Рихтер. Программирование на платформе Microsoft .NET/ В.Л.Григо-рьев. – М.: Энергоатомиздат, 2010. – 209 с.

5. Фролов, Г.В. Библиотека системного программиста. Том 14. Графический интерфейс GDI в Microsoft Windows/ Г. В. Фролов. – М.: Диалог-МИФИ, 2005. – 288 с.

6. Бизли. Python. Подробный справочник / В.И.Юров. – Спб: Питер, 2014.– 624 с.

7. Практическое руководство к курсовому проектированию по курсу «Информатика» для студентов технических специальностей дневной и заочной форм обучения – Гомель : ГГТУ им. П.О. Сухого, 2004. – 32 с.

# Приложение А



# Приложение B

**Файл Form1.cs:**

using MathFunctions;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseWork

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

int a = Int32.Parse(A.Text);

int b = Int32.Parse(B.Text);

int c = Int32.Parse(C.Text);

int d = Int32.Parse(D.Text);

int xn = Int32.Parse(XH.Text);

int xk = Int32.Parse(XK.Text);

int yn = Int32.Parse(YH.Text);

int yk = Int32.Parse(YK.Text);

int n = Int32.Parse(N.Text);

label3.Text = "Площадь = " + Calculate.SurfaceArea(a, b, c, d, xn, xk, yn, yk, n);

}

catch(Exception exception)

{

MessageBox.Show(

"Проверьте введеные данные",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Error,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

int a = Int32.Parse(A.Text);

int b = Int32.Parse(B.Text);

int c = Int32.Parse(C.Text);

int d = Int32.Parse(D.Text);

int xn = Int32.Parse(XH.Text);

int xk = Int32.Parse(XK.Text);

int yn = Int32.Parse(YH.Text);

int yk = Int32.Parse(YK.Text);

int n = Int32.Parse(N.Text);

Calculate.PrintToFile(a, b, c, d, xn, xk, yn, yk, n);

}

catch (Exception exception)

{

MessageBox.Show(

"Проверьте введеные данные",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Error,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

try

{

ProcessStartInfo procInfo = new ProcessStartInfo();

procInfo.FileName = "grafik.exe";

Process.Start(procInfo);

}

catch (Exception ex)

{

MessageBox.Show(

"Проверьте наличие python файла",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Error,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

private async void button3\_Click(object sender, EventArgs e)

{

try

{

int a = Int32.Parse(A.Text);

int b = Int32.Parse(B.Text);

int c = Int32.Parse(C.Text);

int d = Int32.Parse(D.Text);

int xn = Int32.Parse(XH.Text);

int xk = Int32.Parse(XK.Text);

int yn = Int32.Parse(YH.Text);

int yk = Int32.Parse(YK.Text);

int n = Int32.Parse(N.Text);

chart1.Series.Clear();

for (int i = 0; i < 10; i++)

{

var res = await Analysis(i + 1, n, xn, xk, yn, yk, a, b, c, d);

threadsInfo\_dataGridView.Rows.Add();

//Количество потоков

threadsInfo\_dataGridView[0, i].Value = i + 1;

//Площадь

threadsInfo\_dataGridView[1, i].Value = res.Item1;

//Время выполнения

threadsInfo\_dataGridView[2, i].Value = res.Item2.TotalMilliseconds.ToString() + " мс";

//Вычисление и запись погрешности

threadsInfo\_dataGridView[3, i].Value = Math.Abs(res.Item1 - 82743265) / res.Item1 \* 100;

chart1.Series.Add($"{i + 1} поток");

chart1.Series[i].Points.Add(res.Item2.TotalMilliseconds);

chart1.Legends.Clear();

}

}

catch (Exception exception)

{

MessageBox.Show(

"Проверьте введеные данные",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Error,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

private async Task<(double,TimeSpan)> Analysis(int number,int n, double xn, double xk, double yn, double yk,

double a, double b, double c,double d)

{

DateTime time1 = DateTime.Now;

List<Task<double>> tasks = new List<Task<double>>();

double h = (xk - xn) / number;

for (int i = 0; i < number; i++)

{

double xnStep = xn + h \* i;

double xkStep = xn + h \* (i + 1);

tasks.Add(new Task<double>(() => { return Calculate.SurfaceArea(a, b, c, d, xnStep, xkStep, yn, yk, n / number); }));

tasks[i].Start();

}

await Task.WhenAll(tasks);

DateTime time2 = DateTime.Now;

return (tasks.Sum(p=>p.Result), time2 - time1);

}

}

}

**Файл Calculate.cs:**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Math;

namespace MathFunctions

{

public class Calculate

{

private static string FILE = "cordinats.txt";

static double f(double a, double b, double c, double d, double x, double y)

{

return Sqrt(1 + Pow(3 \* a \* Pow(x, 2) + 2 \* c \* x, 2) + Pow((3 \* b \* Pow(y, 2) + 2 \* d \* y), 2));

}

public static double SurfaceArea(double a, double b, double c, double d, double xn, double xk, double yn, double yk, double n)

{

double hx = (xk - xn) / n;

double hy = (yk - yn) / n;

double sum = 0;

for (int i = 1; i < n - 1; i += 2)

{

double xi = xn + hx \* i;

sum += g(xi - hx, hy, n, yn, a, b, c, d) + 4 \* g(xi, hy, n, yn, a, b, c, d) + g(xi + hx, hy, n, yn, a, b, c, d);

}

return sum \* hx / 3;

}

static double g(double x, double hy, double n, double yn, double a, double b, double c, double d)

{

double res = hy / 3;

double sum = 0;

for (int j = 1; j < n - 1; j += 2)

{

double yj = yn + hy \* j;

sum += f(a, b, c, d, x, yj - hy) + 4 \* f(a, b, c, d, x, yj) + f(a, b, c, d, x, yj + hy);

}

return res \* sum;

}

public static void PrintToFile(double a, double b, double c, double d, double xn, double xk, double yn, double yk, double n)

{

double hx = (xk - xn) / n;

double hy = (yk - yn) / n;

using (StreamWriter sw = new StreamWriter(FILE, false, System.Text.Encoding.Default))

{

sw.WriteLine(n);

List<double> coordinatX = new List<double>();

for(double x = xn; x <= xk; x+=hx)

{

coordinatX.Add(x);

sw.Write(x.ToString() + " ");

}

sw.WriteLine();

List<double> coordinatY = new List<double>();

for (double y = yn; y <= yk; y += hy)

{

coordinatY.Add(y);

sw.Write(y.ToString() + " ");

}

sw.WriteLine();

foreach (double y in coordinatY)

{

foreach (double x in coordinatX)

{

sw.Write(a \* Pow(x,3) + b \* Pow(y, 3) + c \* Pow(x, 2) + d \* Pow(y, 2) + " ");

}

sw.WriteLine();

}

sw.Flush();

sw.Close();

}

}

}

}

**Скрипт Puthon:**

import pylab

import sys

from decimal import Decimal

import array

import decimal

from mpl\_toolkits.mplot3d import Axes3D

from matplotlib import cm

import matplotlib.pyplot as plt

import numpy

f = open('cordinats.txt')

i = 0

n = 0

zz = []

for line in f:

if i == 0:

n = float(line)

if i == 1:

x = [number.replace(',','.') for number in line.split(' ')]

newx = []

count = 0

while (count < n):

newx.append(x[count])

count += 1

x = [float(number) for number in newx]

x = [float(number) for number in x]

j = 0

xx = []

while(j < len(x)):

newx = []

count = 0

while(count < n):

newx.append(x[count])

count += 1

xx.append(newx)

j += 1

xx = numpy.array(xx)

if i == 2:

y = [number.replace(',', '.') for number in line.split(' ')]

newy = []

count = 0

while (count < n):

newy.append(y[count])

count += 1

y = [float(number) for number in newy]

j = 0

yy = []

while (j < len(y)):

newy = []

count = 1

while (count <= n):

newy.append(y[j])

count += 1

yy.append(newy)

j += 1

yy = numpy.array(yy)

if (i > 2 and i <= n + 2):

z = [number.replace(',', '.') for number in line.split(' ')]

newz = []

count = 0

while(count < n):

newz.append(z[count])

count += 1

newz = [float(number) for number in newz]

j = 0

zz.append(newz)

i = i + 1

zz = numpy.array(zz)

print(len(xx))

print(len(yy))

print(len(zz))

fig = pylab.figure()

axes = Axes3D(fig)

axes.plot\_surface(xx, yy, zz)

pylab.show()