СОДЕРЖАНИЕ

[Введение 3](#_Toc9240569)

[1 Обзор методов решения задачи 4](#_Toc9240570)

[1.1 Поверхности 4](#_Toc9240571)

[1.2 Вычисление площади поверхности 7](#_Toc9240572)

[1.3 Математический редактор Mathcad 9](#_Toc9240573)

[1.4 Использованные средства разработки 11](#_Toc9240574)

[2 Разработка функциональной модели приложения 17](#_Toc9240575)

[2.1 Постановка задачи 17](#_Toc9240576)

[2.2 Построение функциональной модели 18](#_Toc9240577)

[3 Разработка компонентов 19](#_Toc9240578)

[4 Пользовательский интерфейс 21](#_Toc9240579)

[5 Исследование эффективности реализации многопоточности 25](#_Toc9240580)

[6 Тестирование и верификация 26](#_Toc9240581)

[Заключение 27](#_Toc9240582)

[Список использованных источников 28](#_Toc9240583)

[Приложение А ­ Диаграмма компонентов 29](#_Toc9240584)

[Приложение В ­ Листинг программного кода 30](#_Toc9240585)

Введение

В связи с динамичным развитием вычислительной техники и постоянно прогрессирующими технологиями, количество задач, для получения точного решения которых приходится прибегать к огромному количеству громоздких и трудоемких вычислений, нелинейно растет. Порой, найти решение некоторых из них вручную и вовсе 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.

Численные методы решения задач математического анализа - это изучение алгоритмов и условий сходимости итерационных методов, определение границ применимости методов, исследования оценок погрешностей методов и вычислений. Основной целью использования численных методов является их реализация на ЭВМ, то есть составление программы для требуемого алгоритма и решения с ее помощью конкретной задачи.

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.

Целью данного курсового проекта является разработка компонентного приложения подготовки данных для обработки и отрисовки поверхности с использованием многопоточности.

Задачи данного курсового проекта следующие:

* изучить теоретический материал о методах решения задачи;
* разработать функциональную модель;
* разработать необходимые программные компоненты;
* выполнить алгоритмический анализ поставленной задачи;
* исследовать эффективность организации многопоточности.

Поверхность в геометрии и топологии — двумерное топологическое многообразие. Наиболее известными примерами поверхностей являются границы геометрических тел в обычном трёхмерном евклидовом пространстве.

1. Обзор методов решения задачи
   1. Поверхности

* + 1. 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](https://ru.wikipedia.org/wiki/%D0%9E%D1%81%D0%BE%D0%B1%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) или самопересечения.

«Двухмерность» поверхности подразумевает возможность реализовать на ней [метод координат](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BA%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82), хотя и необязательно для всех точек. Так, поверхность [Земли](https://ru.wikipedia.org/wiki/%D0%97%D0%B5%D0%BC%D0%BB%D1%8F) (в идеале) представляет собой двумерную [сферу](https://ru.wikipedia.org/wiki/%D0%A1%D1%84%D0%B5%D1%80%D0%B0), широта и долгота каждой точки которой являются её координатами (за исключением [полюсов](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D0%BB%D1%8E%D1%81) и [180-го меридиана](https://ru.wikipedia.org/wiki/180-%D0%B9_%D0%BC%D0%B5%D1%80%D0%B8%D0%B4%D0%B8%D0%B0%D0%BD)).

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.

* + 1. 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 которых удовлетворяют определённому виду уравнений:

(1)

Если функция [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.

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

 (2)

* + 1. 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)).

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 и 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 , . 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) , , задаётся при помощи формул , , ([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 , и 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 и были различными соответствующие точки и .

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.

* + 1. 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:

(3)

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:

(4)

Эта система уравнений 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:

(5)

Геометрически последнее условие означает, что векторы нигде не параллельны. Параметры  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  тогда S является гладкой регулярной поверхностью, если функция f дифференцируема. Этот вариант можно рассматривать как частный случай параметрического задания: ; ; [1].

* 1. Вычисление площади поверхности
     1. Если везде в области D на координатной плоскости для формулы

положить , то, в соответствии со своим геометрическим смыслом, двойной интеграл будет численно равен площади S области интегрирования D, то есть

В полярной системе координат эта же самая формула приобретает вид

* + 1. Численное интегрирование методом Симпсона.

Численное интегрирование — вычисление значения определённого интеграла (как правило, приближённое). Численное интегрирование применяется, когда:

* подынтегральная функция не задана аналитически (например, она представлена в виде массива значений в узлах некоторой расчётной сетки);
* аналитическое представление подынтегральной функции известно, но аналитическое выражение ее первообразной затруднено.

Основная идея большинства методов численного интегрирования состоит в замене подынтегральной функции на более простую, интеграл от которой легко вычисляется аналитически. Наиболее часто с этой целью используется метод трапеций.

Пусть требуется вычислить определенный интеграл

Принцип метода Симпсона состоит в замене подынтегральной функции интерполяционным многочленом Ньютона второй степени. Тогда для отрезка имеем следующее значение площади подынтегральной кривой:

где – значения функции в соответствующих точках (на концах отрезка и в его середине). На рисунке 1.1 можно увидеть функцию , аппроксимированную квадратичным полиномом .

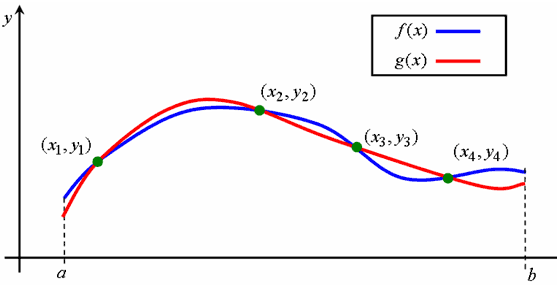


Рисунок 1.1 – Аппроксимация функции f(x) квадратичным полиномом g(x)

* + 1. 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):

Каждый из интегралов

можно вычислить с помощью формул численного интегрирования для одномерных интегралов [2].

* 1. Математический редактор Mathcad
     1. Mathcad – это приложение для математических и инженерных вычислений, промышленный стандарт проведения, распространения и хранения расчетов. Mathcad является универсальной системой, т.е. может использоваться в любой области науки и техники – везде, где применяются математические методы.

Документы Mathcad представляют расчеты в виде, очень близком к стандартному математическому языку, что упрощает постановку и решение задач. Mathcad содержит текстовый и формульный редактор, вычислитель, средства научной и деловой графики, а также огромную базу справочной информации, как математической, так и инженерной. Редактор формул обеспечивает естественный «многоэтажный» набор формул в привычной математической нотации (деление, умножение, квадратный корень, интеграл, сумма и т.д.). Мощные средства построения графиков и диаграмм сочетают простоту использования и эффектные способы визуализации данных и подготовки отчетов.

Вычислительные средства Mathcad обеспечивают расчеты по сложным математическим формулам, включая численные методы и аналитические преобразования. Mathcad имеет большой набор встроенных математических функций, позволяет вычислять ряды, суммы, произведения, интегралы, производные, работать с комплексными числами, решать линейные и нелинейные уравнения, а также дифференциальные уравнения и системы, проводить минимизацию и максимизацию функций, выполнять векторные и матричные операции, статистический анализ и т.д. [3].

* + 1. Построение графика поверхности в системе Mathcad может осуществляться несколькими способами.

Построение трехмерных графиков без задания матрицы.

В данном случае для построения достаточно задать функцию переменных х и у. В результате построение графиков поверхностей выполняется также просто, как и построение двухмерных графиков. Недостат­ками такого построения являются неопределенность в масштабировании и то, что не все поверхности второго порядка можно построить таким образом.

Построение поверхностей по матрице аппликат их точек.

Поскольку элементы матрицы М – индексированные переменные с целочисленными ин­дексами, то перед созданием матрицы требуется задать индексы в виде ранжированных пе­ременных с целочисленными значениями, а затем из них сформировать сетку значений  и  – координат для аппликат . Значения х и у могут быть любыми действительными числами.

После указанных выше определений вводится шаблон графика (либо с помощью подменю меню Вставка, либо с помощью панели Graph). Левый верхний угол шаблона помещается в место расположения курсора. Шаблон содержит единственное место ввода – темный прямоугольник у левого нижнего угла основного шаблона. В него надо занести имя мат­рицы аппликат поверхности. После этого надо установить указатель мыши в стороне от графического блока и щелкнуть левой кнопкой.

Следует заметить, так как график строится на основе матрицы, содержащей только координаты высот фигуры, то истинные масштабы по осям абсцисс и ординат неизвестны и на рисунках не проставляются. Однако можно выводить порядковые номера элементов матриц в заданном направлении. Необходимо следить за тем, как сформировать векторы  и , чтобы поверхность выглядела естественно и была видна нужная часть поверхности.

Форматирование трехмерных графиков.

Принцип форматирования трехмерных графиков такой же, как и форматирования двухмерных графиков. Отличие состоит лишь в большем количестве параметров форматирования [4].

* 1. Использованные средства разработки
     1. [.NET Core](https://docs.microsoft.com/ru-ru/dotnet/core/about) — это универсальная платформа разработки с [открытым кодом](https://github.com/dotnet/coreclr/blob/master/LICENSE.TXT), которую поддерживает корпорация Майкрософт и сообщество .NET на сайте [GitHub](https://github.com/dotnet/core). Она является кроссплатформенной (поддерживает Windows, macOS и Linux) и может использоваться для создания приложений для устройств, облака и Интернета вещей.

.NET Core — это очень похожая на другие продукты .NET, но в то же время уникальная платформа. При ее разработке ставилась цель обеспечить максимальную адаптируемость к новым платформам и рабочим нагрузкам. У нее есть порты на несколько ОС и ЦП, и она поддерживает перенос на многие другие платформы.

Продукт состоит из нескольких компонентов, каждый из которых может быть адаптирован к новым платформам отдельно и в разное время. Среда выполнения и основные библиотеки, связанные с платформой, должны переноситься как единое целое. Не зависящие от платформы библиотеки должны работать "как есть" на любой платформе. В рамках проекта имеется тенденция к ограничению реализаций, предназначенных для конкретных платформ, с целью повышения эффективности разработки: если алгоритм или интерфейс API можно полностью или частично реализовать с помощью независимого от платформы кода C#, предпочтение отдается именно такому варианту.

.NET Core позволяет создавать приложения и библиотеки на языках C#, Visual Basic и F#. Эти языки уже интегрированы или могут быть интегрированы в ваши любимые текстовые редакторы и интегрированные среды разработки, такие как [Visual Studio](https://visualstudio.microsoft.com/vs/?utm_medium=microsoft&utm_source=docs.microsoft.com&utm_campaign=inline+link), [Visual Studio Code](https://marketplace.visualstudio.com/items?itemName=ms-vscode.csharp), Sublime Text и Vim.

Основные различия между .NET Core и .NET Framework:

* .NET Core поддерживает не все модели приложений платформы .NET Framework. В частности, не поддерживаются веб-формы ASP.NET и ASP.NET MVC, однако поддерживается ASP.NET Core MVC. Было объявлено, что [.NET Core 3 будет поддерживать WPF и Windows Forms](https://devblogs.microsoft.com/dotnet/net-core-3-and-support-for-windows-desktop-applications/).
* API-интерфейсы — .NET Core включает обширное подмножество библиотеки базовых классов .NET Framework с отличающейся организацией кода (другие имена сборок и ключевые отличия в членах, предоставляемых для типов). Из-за этих различий в некоторых случаях необходимы изменения при портировании исходного кода на .NET Core.
* платформа .NET Framework поддерживает Windows и Windows Server, в то время как .NET Core также поддерживает Mac OS и Linux.
* платформа .NET Core имеет открытый исходный код. В случае с .NET Framework открытый исходный код имеет только [подмножество библиотек, доступных только для чтения](https://github.com/microsoft/referencesource).

Хотя платформа .NET Core является уникальной и существенно отличается от .NET Framework и других реализаций .NET, вы можете легко использовать один и тот же код в разных реализациях как в формате исходного кода, так и в виде двоичных файлов.

* + 1. ASP.NET MVC является многофункциональной платформой для создания веб-приложений и API-интерфейсов с помощью структуры проектирования Model-View-Controller.

Структура архитектуры MVC предполагает разделение приложения на три основные группы компонентов: модели, представления и контроллеры. Это позволяет реализовать принципы [разделения задач](https://docs.microsoft.com/ru-ru/dotnet/standard/modern-web-apps-azure-architecture/architectural-principles#separation-of-concerns). Согласно этой структуре запросы пользователей направляются в контроллер, который отвечает за работу с моделью для выполнения действий пользователей и (или) получение результатов запросов. Контроллер выбирает представление для отображения пользователю со всеми необходимыми данными модели.

На рисунке 1.2 показаны три основных компонента и существующие между ними связи.

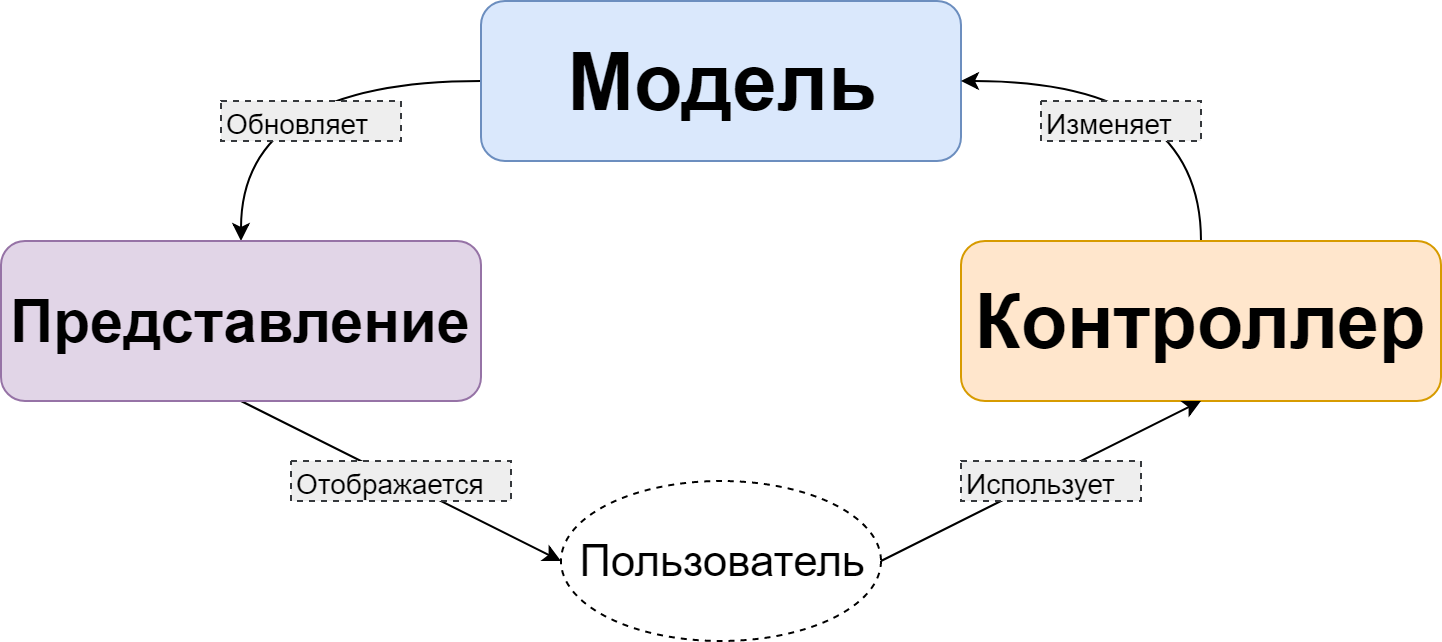


Рисунок 1.2 – Компоненты MVC

Такое распределение обязанностей позволяет масштабировать приложение в контексте сложности, так как проще писать код, выполнять отладку и тестирование компонента (модели, представления или контроллера) с одним заданием. Гораздо труднее обновлять, тестировать и отлаживать код, зависимости которого находятся в двух или трех этих областях. Например, логика пользовательского интерфейса, как правило, подвергается изменениям чаще, чем бизнес-логика. Если код представления и бизнес-логика объединены в один объект, содержащий бизнес-логику, объект необходимо изменять при каждом обновлении пользовательского интерфейса. Это часто приводит к возникновению ошибок и необходимости повторно тестировать бизнес-логику после каждого незначительного изменения пользовательского интерфейса.

Модель в приложении MVC представляет состояние приложения и бизнес-логику или операций, которые должны в нем выполняться. Бизнес-логика должна быть включена в состав модели вместе с логикой реализации для сохранения состояния приложения. Как правило, строго типизированные представления используют типы ViewModel, предназначенные для хранения данных, отображаемых в этом представлении. Контроллер создает и заполняет эти экземпляры ViewModel из модели.

Представления отвечают за представление содержимого через пользовательский интерфейс. Для внедрения кода .NET в разметку HTML они используют [подсистему просмотра Razor](https://docs.microsoft.com/ru-ru/ASPNET/core/mvc/overview?view=aspnetcore-2.1#razor-view-engine). Представления должны иметь минимальную логику, которая должна быть связана с представлением содержимого. Если есть необходимость выполнять большую часть логики в представлении для отображения данных из сложной модели, рекомендуется воспользоваться [компонентом представления](https://docs.microsoft.com/ru-ru/ASPNET/core/mvc/views/view-components?view=aspnetcore-2.1), ViewModel или шаблоном представления, позволяющими упростить представление.

Контроллеры — это компоненты для управления взаимодействием с пользователем, работы с моделью и выбора представления для отображения. В приложении MVC представление служит только для отображения информации. Обработку введенных данных, формирование ответа и взаимодействие с пользователем обеспечивает контроллер. В структуре MVC контроллер является начальной отправной точкой и отвечает за выбор рабочих типов моделей и отображаемых представлений (именно этим объясняется его название — он контролирует, каким образом приложение отвечает на конкретный запрос) [5].

* + 1. C# — простой, современный объектно-ориентированный и типобезопасный язык программирования. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript. Здесь представлен обзор основных компонентов языка. Если вы хотите изучить язык с помощью интерактивных примеров, рекомендуем поработать с нашими [вводными руководствами по C#](https://docs.microsoft.com/ru-ru/dotnet/csharp/tutorials/intro-to-csharp/index).

C# является объектно-ориентированным языком, но поддерживает также и компонентно-ориентированное программирование. Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности. Важная особенность таких компонентов — это модель программирования на основе свойств, методов и событий. Каждый компонент имеет атрибуты, предоставляющие декларативные сведения о компоненте, а также встроенные элементы документации. C# предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому C# отлично подходит для создания и применения программных компонентов.

Вот лишь несколько функций языка C#, обеспечивающих надежность и устойчивость приложений: сборка мусора автоматически освобождает память, занятую уничтоженными и неиспользуемыми объектами; обработка исключений предоставляет структурированный и расширяемый способ выявлять и обрабатывать ошибки; строгая типизация языка не позволяет обращаться к неинициализированным переменным, выходить за пределы индексируемых массивов или выполнять неконтролируемое приведение типов.

В C# существует единая система типов. Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Таким образом, все типы используют общий набор операций, и значения любого типа можно хранить, передавать и обрабатывать схожим образом. Кроме того, C# поддерживает пользовательские ссылочные типы и типы значений, позволяя как динамически выделять память для объектов, так и хранить упрощенные структуры в стеке.

Чтобы обеспечить совместимость программ и библиотек C# при дальнейшем развитии, при разработке C# много внимания было уделено управлению версиями. Многие языки программирования обходят вниманием этот вопрос, и в результате программы на этих языках ломаются чаще, чем хотелось бы, при выходе новых версий зависимых библиотек. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса [6].

* + 1. Язык JavaScript — это клиентский язык web-программирования, который был создан в 1995 году, разработчиком Бренданом Айком.

JavaScript обычно применяется для манипулирования объектами в различных приложениях, но наибольшую популярность он приобрел как один из основных языков применяемых при создании сайтов (и как единственный клиентский язык web-программирования).

Код языка JavaScript, обычно исполняется в окне браузера, на открытой странице сайта. Это происходит благодаря тому, что в веб-браузере по-умолчанию имеется интерператор языка JavaScript, благодаря которому браузер имеет возможность понимать и исполнять код написанный на языке JavaScript.

Интерпретатор языка JavaScript, является частью веб-браузера, когда веб браузер открывает страницу сайта, браузер создаёт объектную модель документов (DOM), интерпретатор JavaScript получает доступ к DOM и благодаря этому, вебмастера создавая различные скрипты (web-приложения) могут управлять и манипулировать объектами страницы (всеми тегами: абзацами, заголовками, таблицами, формами и т.д.).

JavaScript так же, как и языки HTML и CSS, принадлежит всему человечеству, им не владеет ни одна компания или персона. Однако само слово — «JavaScript», принадлежит компании Oracle Corporation и чтобы не иметь проблем с авторскими правами, ученые разрабатывающие данный язык в научных документах называют его ECMAScript [7].

1. **Разработка функциональной модели приложения**
   1. Постановка задачи

Необходимо провести анализ заданной поверхности в математическом пакете. Анализ включает: построение графика поверхности, вычисления площади поверхности, используя аналитическую формулу

Вид поверхности:

Метод численного интегрирования – метод Симпсона. Значение коэффициентов и пределы интегрирования представлены в таблице 1.

Таблица 1 – Значения коэффициентов и пределы интегрирования

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Значения коэффициентов | | | | Пределы интегрирования | | | |
| A | B | C | D | Xнач | Xкон | Yнач | Yкон |
| 2 | 2 | 3 | 3 | -60 | 60 | -60 | 60 |

Необходимо провести численный анализ заданной поверхности, используя формулы численного интегрирования. Программный код оформить в виде компонента, позволяющий принимать в качестве параметров границы области интегрирования, вычислять площадь заданной поверхности в пределах области интегрирования, готовить данные для построения графика в математическом пакете.

Необходимо провести анализ погрешности численных методов, передав данные, полученные в пункте 2 в математический пакет. Программный код оформить в виде компонента.

Необходимо провести анализ скорости обработки заданной поверхности и погрешности численных методов, в зависимости от количества потоков, используемых для расчетов.

* 1. Построение функциональной модели

Структурная схема приложения представлена на рисунке 2.1

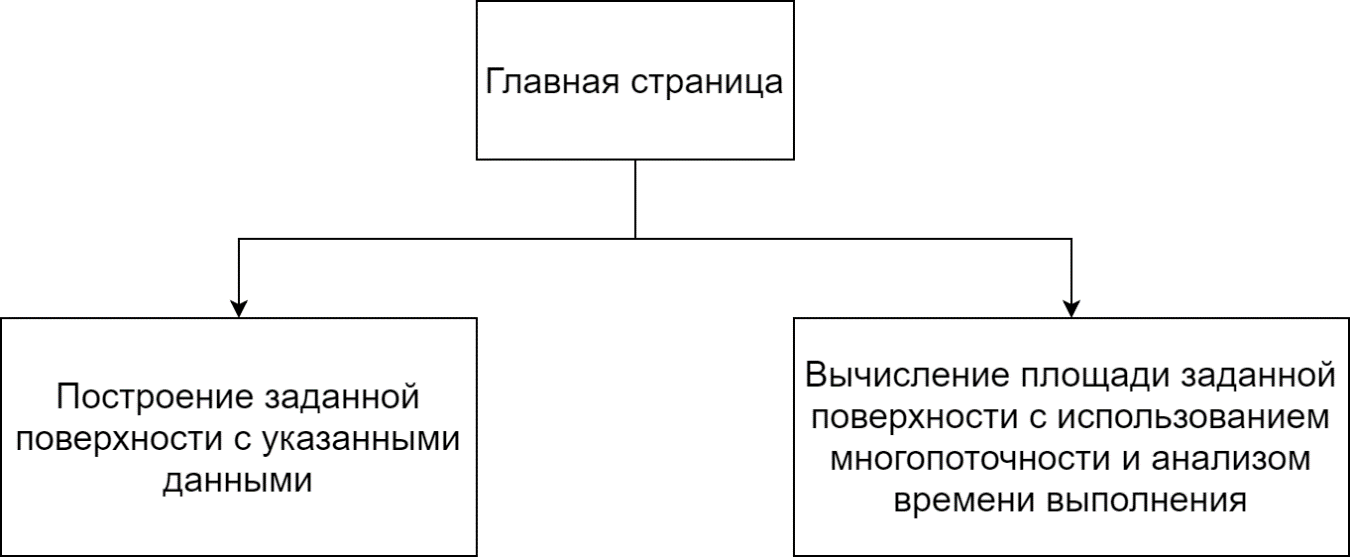


Рисунок 2.1 – Структурная схема программы

Разработанное приложение разбито на несколько модулей:

* библиотека SimpsonMethod;
* библиотека Surface;
* проект CourseWork;
* скрипт view.js.

Библиотека SimpsonMethod содержит метод решения двойного интеграл методом Симпсона.

Библиотека Surface содержит описание класса поверхности. В этой библиотеке содержатся функция вычисления всех точек поверхности и функция вычисления площади поверхности.

JavaScript файл view содержит код, который при помощи сторонней библиотеки Plot.ly отрисовывает поверхность и график, содержащий данные об обработки поверхности с использованием различного количества потоков.

1. **Разработка компонентов**

В классе Surface осуществляется вычисление точек координат для построения поверхности, а также вычисление площади этой поверхности. Описание всех функция и полей содержится в таблице 2.

Таблица 2 – Описание класса Surface

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Модификатор доступа | Назначение |
| a, b, c, d | double | protected | Коэффициенты заданной поверхности. |
| XSt | double | public | Начальная координата по оси Х |
| XFin | double | public | Конечная координата по оси Х |
| YSt | double | public | Начальная координата по оси Y |
| YFin | double | public | Конечная координата по оси Y |
| Step | double | public | Шаг |
| SetIntervals  (double xSt, double xFin, double ySt, double yFin) | void | public | Метод, устанавливающий интервал, на котором будут проводится расчёты поверхности. |
| GetSurfaceArea(…) | double | public | Метод для подсчета площади |
| GetPoints() | IEnumerable <Point> | public | Метод, вычисляющий все координаты поверхности для дальнейшего её построения. |
| InitialFunction (double x,  double y) | double | protected | Вспомогательный метод для вычисления площади. Описывает подынтегральную функцию. |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Модификатор доступа | Назначение |
| DerivativeOfFuncion (double x,  double y) | double | protected | Функция вычисления координаты Z по X и Y. |

В классе Points определяются координаты X, Y и Z. Класс Points описан в таблице 3.

Таблица 3 – Описание класса Point

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Параметры | Назначение |
| X | double | public | Координата Х |
| Y | double | public | Координата Y |
| Z | double | public | Координаты Z |

В классе SimpsonMethod вычисляется заданная функция поверхности при помощи метода Симпсона. Класс SimpsonMethod описан в таблице 4.

Таблица 4 – Описание класса SimpsonMethod

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Параметры | Назначение |
| Calculate  (double xSt,  double xFin,  double ySt,  double yFin,  Func<double, double, double> f) | double | public | Вычисление функции по методу Симпсона на заданном промежутке. |

1. Пользовательский интерфейс

Т.к. данное приложение является web-приложением, для его запуска пользователю будет необходим web-браузер. Дизайн приложения выполнен в виде слайдера.

При запуске приложения пользователь попадёт на главную страницу, изображённую на рисунке 4.1.



Рисунок 4.1 – Начальное окно программы

На этой странице пользователь может увидеть навигационное меню по центру, карту сайта снизу и стрелку, переносящую пользователя в следующую секцию. Нажатие по пункту навигационного меню или точке на карте осуществит переход на соответствующую страницу. При щелчке на стрелку справа пользователь будет переправлен на секцию Calculations. Далее рассмотрим все секции данного web-приложения.

В секции Calculations, которая изображена на рисунке 4.2, пользователь сможет задать границы построения поверхности и шаг, с которым будут высчитываться координаты и площадь поверхности.

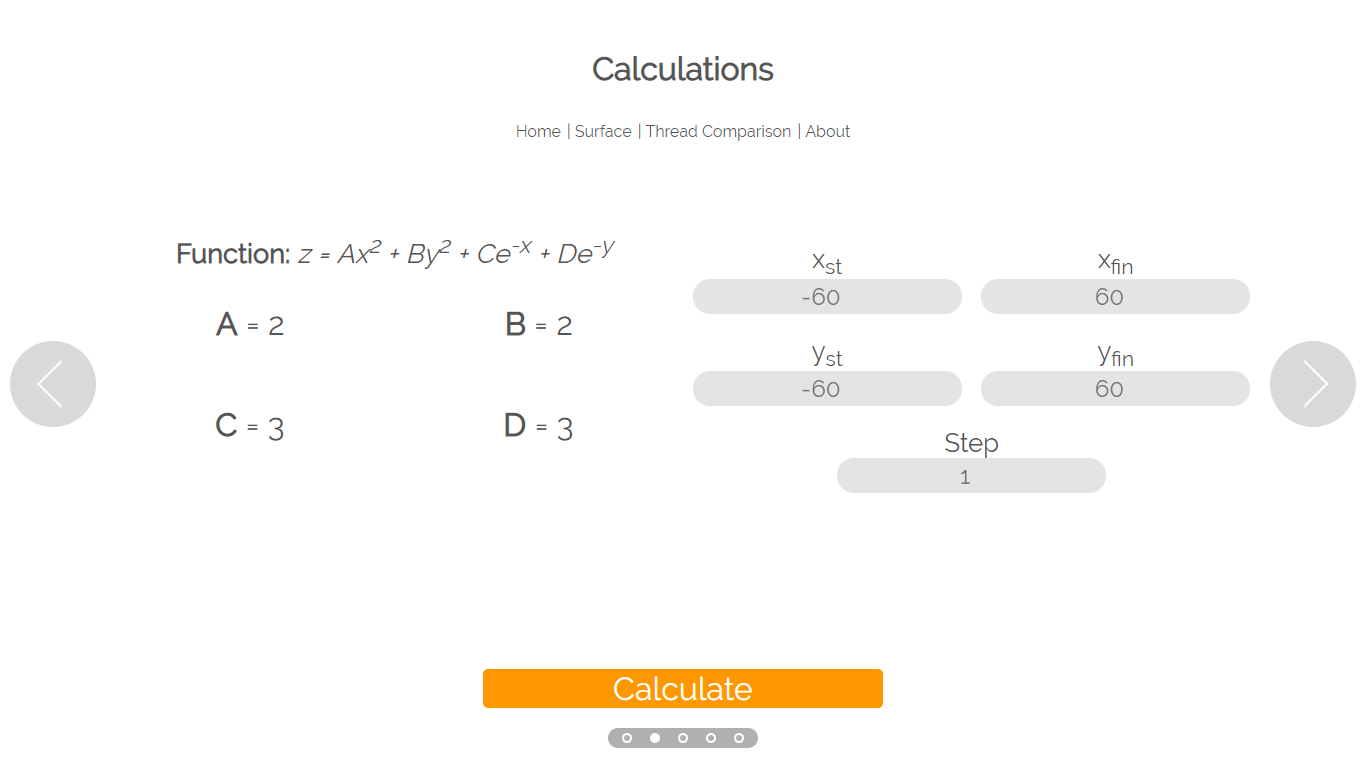


Рисунок 4.2 – Секция Calculations

Слева находятся статичные данные: функция поверхности и коэффициенты. Справа же находятся данные, которые пользователь может изменить: пределы и шаг интегрирования.

При щелчке на кнопку Calculate будет пользователь будет перенесён в секцию Surface, изображённую на рисунке 4.3, где после ответа сервера будет отрисован график поверхности и под ним будут отображены его площадь и её погрешность вычисления.

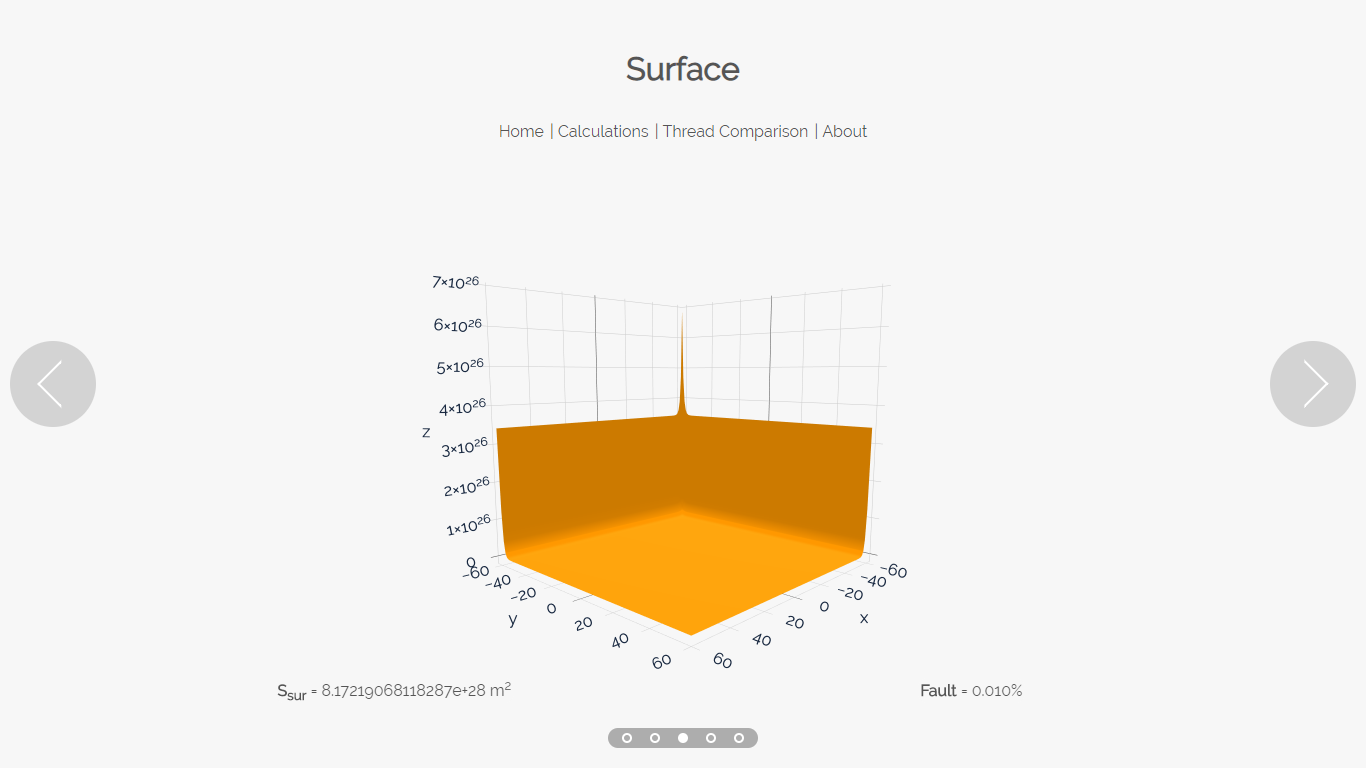


Рисунок 4.3 – Секция Surface с введёнными данными

В случае, если пользователь перейдёт в секцию Surface без ввода данных, изображённую на рисунке 4.4, он увидит заглушку.

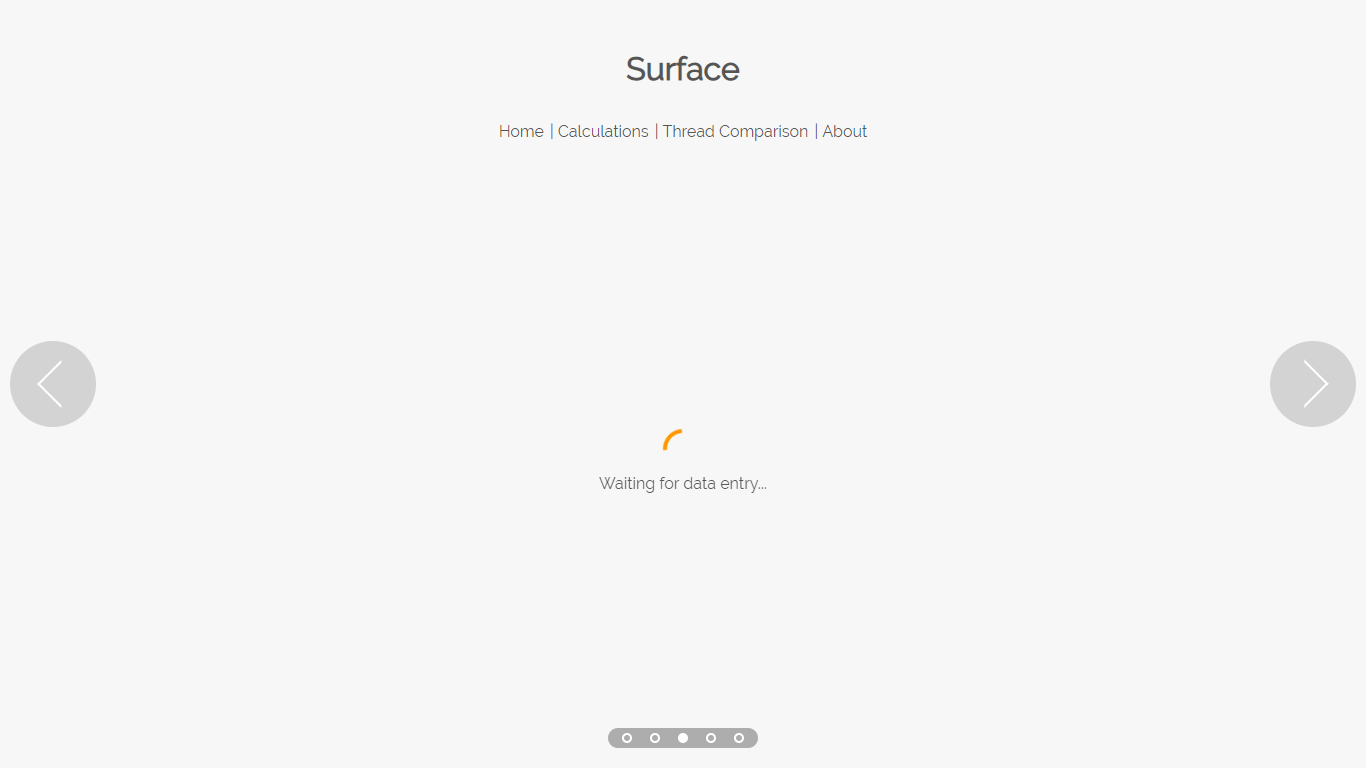


Рисунок 4.4 – Секция Surface без введённых данных

Следующей секцией после Surface является секция Thread Comparison, изображённая на рисунке 4.5, на которой располагается график с результатами вычисления площади поверхности с разным количеством потоках в миллисекундах.

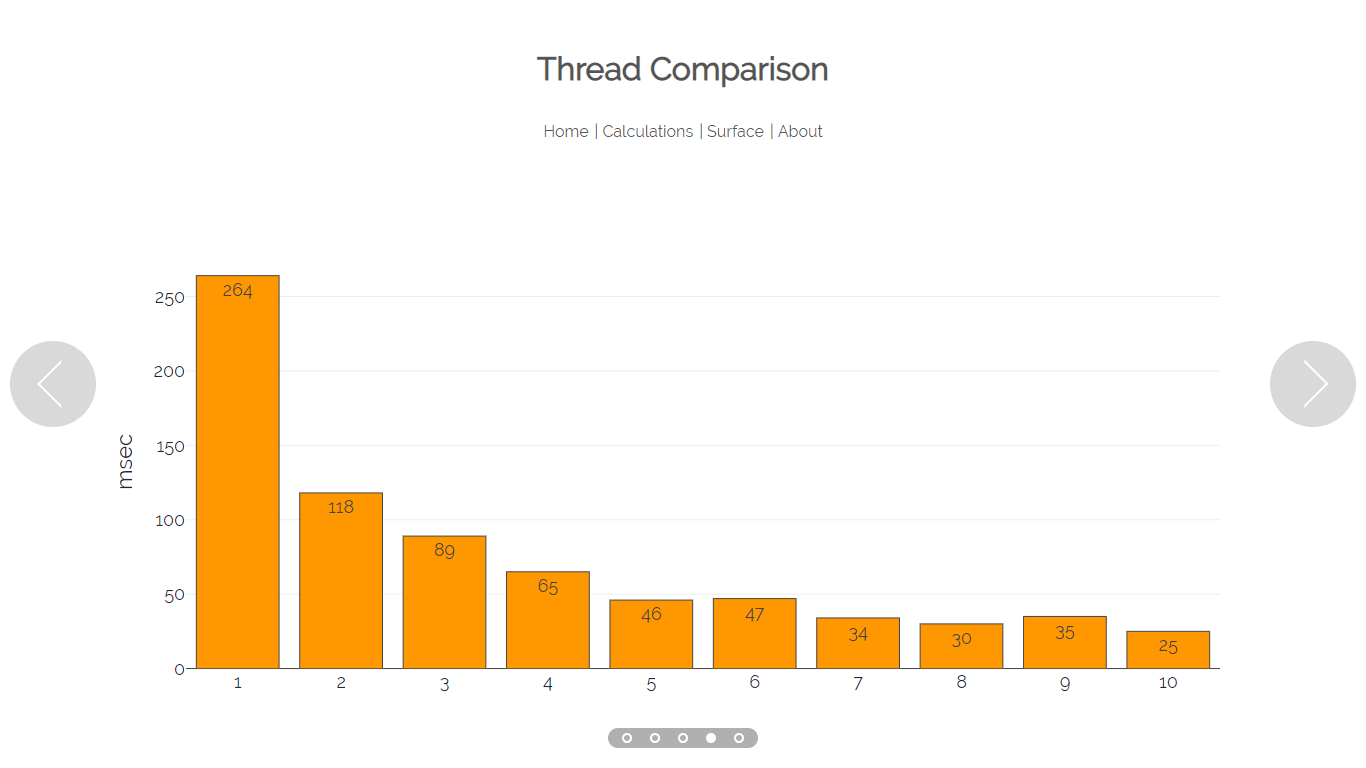


Рисунок 4.5 – Секция Thread Comparison

Для проверки, правильно ли была построена поверхность, использовался математический пакет MathCad. В нём была построена поверхность с описанными ранее данными. Результат, полученный в Mathcad, можно увидеть на рисунке 4.6.

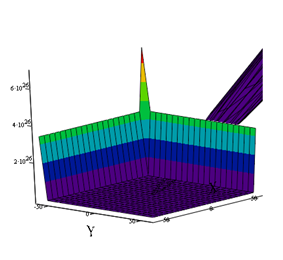


Рисунок 4.6 – График поверхности в пакете «MathCad»

На рисунке 4.7 можно увидеть сравнение графика, полученного при помощи приложения курсового проекта, и графика полученного в математическом пакете MathCad.

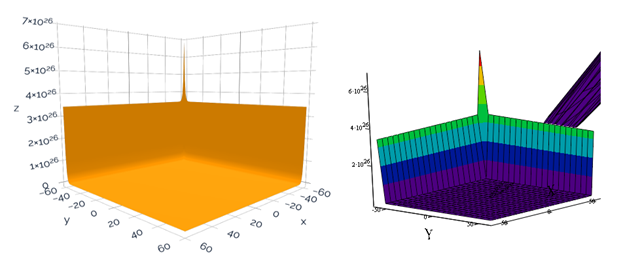


Рисунок 4.7 – Сравнение графиков поверхности приложения и в пакете «MathCad»

1. Исследование эффективности реализации многопоточности

В данном приложении исследование эффективности реализации многопоточности реализовано в виде графика и чисел в таблице. Задачу поочередно выполняют от одного до десяти потоков. После каждого выполнения записывается результат и затраченное время (в миллисекундах) и выводится в диаграмму, указанную на рисунке 4.5.

Исходя из результатов, можно сделать вывод, что в большинстве случаев с ростом количества потоков идёт увеличение скорости вычислений результатов, но только до определённого количества потоков. Исходя из рисунка 4.5, при достижении вычисления при помощи семи потоков, вычисление при увеличении количества потоков производительность практически не изменится.

1. Тестирование и верификация

В приложении предусмотрено различное количество проверок на неверный ввод данных:

* значение начальной границы интегрирования больше значения конечной;
* шаг меньше либо равен нулю;
* попытка ввести данные во время построения графика.

При неверном вводе данных будет высвечена предупреждающая надпись, а также будет отключена кнопка Calculate. На рисунке 6.1 приведён пример, когда значение больше, чем значение .

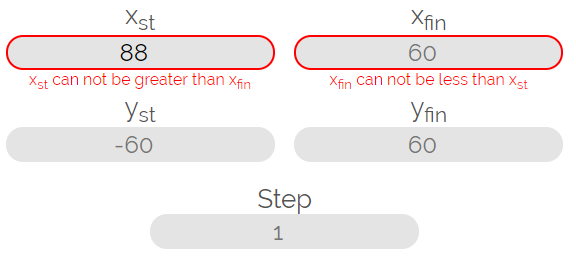


Рисунок 6.1 – Попытка некорректного ввода значения

На рисунке 6.2 изображена ситуация, схожая с ситуацией, изображённой на рисунке 6.1, с таким отличием, что ошибка не в значениях и , а в значениях и .

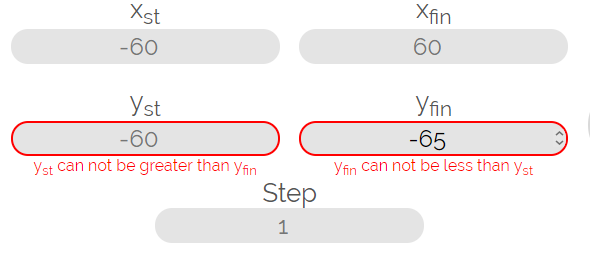


Рисунок 6.2 – Попытка некорректного ввода значения

На рисунке 6.3 можно увидеть ситуацию, когда введён неверный шаг, т.е он равен либо меньше нуля.

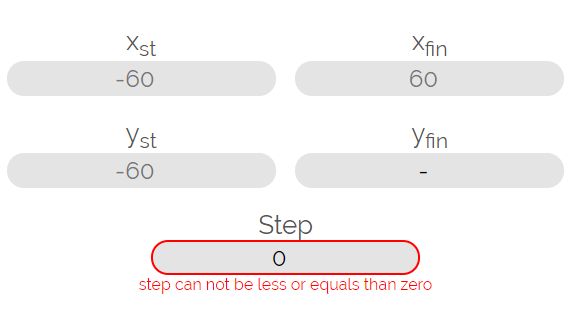


Рисунок 6.3 – Попытка некорректного ввода шага

При попытке нажать на кнопку «Calculate», во время построения графика ничего не произойдёт, т.к. на время его построения кнопка «Calculate» отключается во избежание переполнения стека и возникновения непредвиденных ошибок. На рисунке 6.4 изображена эта ситуация.

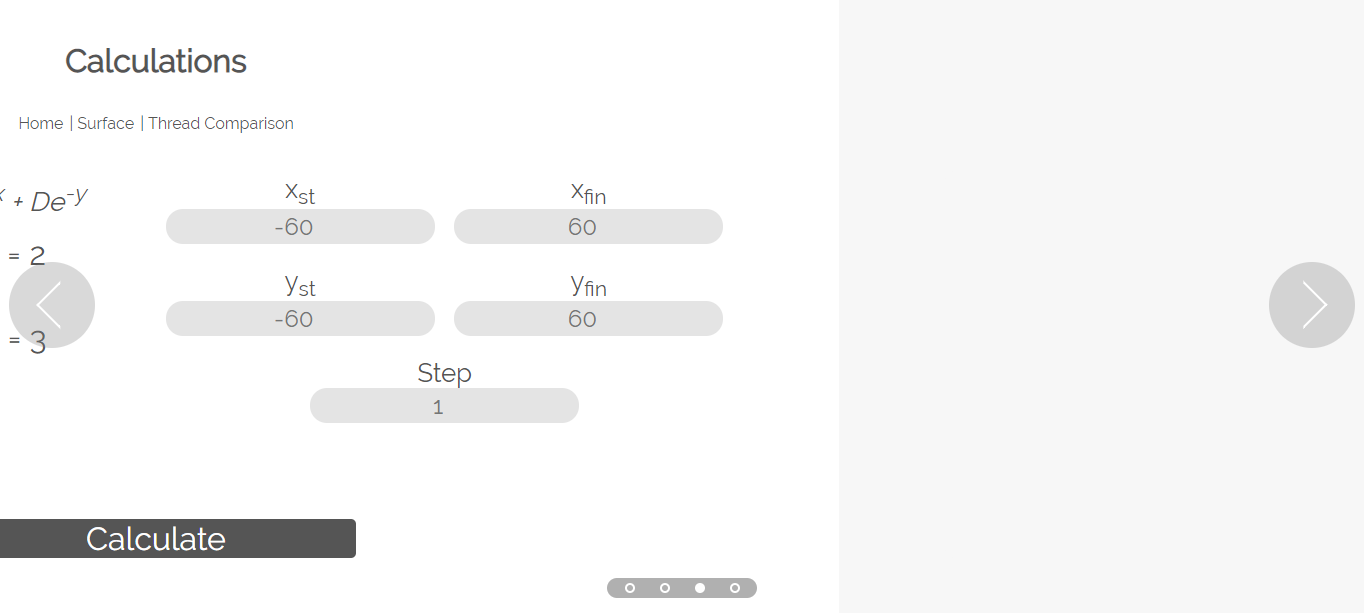


Рисунок 6.4 – Отключённая кнопка «Calculate» во время перехода в секцию «Surface»

Заключение

В ходе выполнения данного курсового проекта было разработано Web-приложение, предназначенное для подсчета площади и построения поверхности.

В приложении предусмотрена возможность ввода собственных значений, отображение диаграмм, позволяющих убедиться в эффективности использования многопоточности.

Для большей наглядности предусмотрена возможность построения поверхности при помощи JavaScript, а также отображение полученных площади поверхности и погрешность вычислений.

Приложение было разработано в средах разработки Microsoft Visual Studio 2019 и JetBrains PhpStorm 2018.3.4.

Для проверки правильности полученного решения в разработанной программе был использован бесплатный математический пакет MathCad. 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. Калиткин, Н.Н. Численные методы / Н.Н. Калиткин. – СПб.:BHV, 2014. – 202 с.
2. Холл, Дж. Современные численные методы решения интегралов/ Дж. Холл, Дж. Уамл. — М.: Мир, 2007. – 312 с.
3. Кудрявцев, Е. М. Справочник по Mathcad / Е. М. Кудрявцев. – М.: ДМK Пресс. – 2005. – 111 с.
4. Построение поверхностей в среде Mathcad[Электронный ресурс]. – 2016. – Режим доступа: https://infopedia.su/14xa2e8.html. – Дата доступа: 30.03.2019.
5. Рихтер, Джефри. Программирование на платформе Microsoft .NET/ В.Л.Григорьев. – М.: Энергоатомиздат, 2010. – 209 с.
6. Троелсен Э., Язык программирования C# 5.0 и платформа .NET/ Э. Троелсен — М.: ООО “И.Д. Вильямс”, 2013. — 1312 с.
7. Современный учебник JavaScript[Электронный ресурс] – 2007. –Режим доступа: http://learn.javascript.ru. – Дата доступа: 27.03.2019.
8. Фролов, Г.В. Библиотека системного программиста. Том 14. Графический интерфейс GDI в Microsoft Windows/ Г. В. Фролов. – М.: Диалог-МИФИ, 2005. – 288 с.

Приложение А

Приложение В