МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Национальный исследовательский университет
«МЭИ»

Институт информационных и вычислительных технологий
Факультет прикладной математики и информатики
Кафедра прикладной математики и искусственного интеллекта

Курсовая работа

Интерполяция функций двух переменных.

Выполнил:
студент 3 курса
группы А-05-19
Каменев Р.Б.

Проверил:
к.ф.-м.н., доцент
Амосова О.А.

Москва, 2021

Оглавление

Постановка задачи

Требуется написать программу приближения таблично заданной функции *f(x,y)* в круге, радиуса R, методом билинейной интерполяции.

Теоретический материал

Билинейная интерполяция

Предположим, что функция $z = f(x, y)$ задана на прямоугольнике(Рис. 1)$[a, b] \times [c, d]$ таблицей значений

$$z_{ij} = f\left(x_{i,}y_j\right),\ \ 0 \le i \le n,\ \ 0 \le j \le m$$

Интерполяционный многочлен $P(x, y)$, обладающий свойством

$$P\left(x_i, y_j\right) = z_{ij},\ \ 0 \le i \le n,\ \ 0 \le j \le m$$

может быть записан в следующем виде:

$$L_{n,m}(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} z_{ij} l_{ni}(x) l_{mi}(y)\ (*)$$

Здесь $l_{ni}(x) = \prod_{s=0,s\neq i}^{n} \frac{x - x_s}{x_i - x_s},\ \ l_{mj}(y) = \prod_{k=0,k\neq j}^{m} \frac{y - y_k}{y_j - y_k}$

Многочлен (*) является аналогом многочлена Лагранжа, причём при фиксированном значении y он является многочленом степени n по переменной x, а при фиксированном значении x – многочленом степени m по переменной y.

При $n = m = 1$ этот многочлен называется *билинейным*, т.к. он линеен по каждой из переменных.

Приведём явную формулу, соответствующую билинейной интерполяции

$$L_{11}(x,y) = \frac{1}{(x_1 - x_0)(y_1 - y_0)}[z_{00}(x_1 - x)(y_1 - y) + z_{10}(x - x_0)(y_1 - y) + z_{01}(x_1 - x)(y - y_0) + z_{11}(x - x_0)(y - y_0)]$$

С помощью этой формулы мы можем вычислять значение в прямоугольнике(Рис. 1), ограниченного точками z00, z10, z10, z11

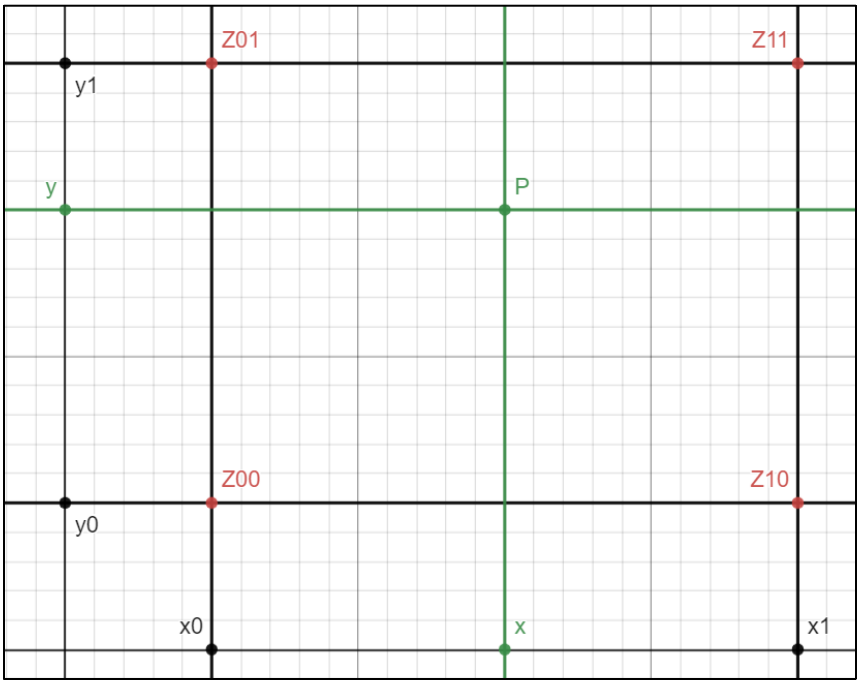


Рис. 1

Метод

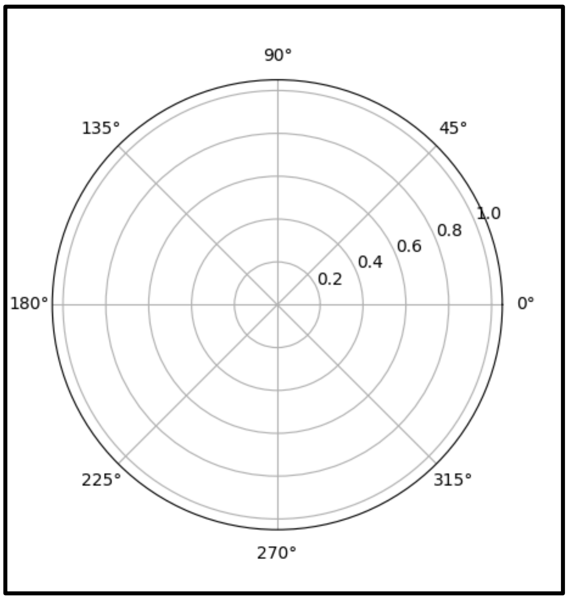
Поступим таким образом: разобьём нашу исходную область на элементарные участки – прямоугольники, для упрощения задачи размеры прямоугольников одинаковые, то есть выполним разбиение с постоянной сеткой. На каждом элементарном разбиении будем вычислять билинейный многочлен. Значения в углах прямоугольника(Рис. 1) будут являться узловыми. Таким образом, разбив всю область на элементарные участки и вычислив соответствующий билинейный многочлен, мы и получим требуемую интерполяцию.

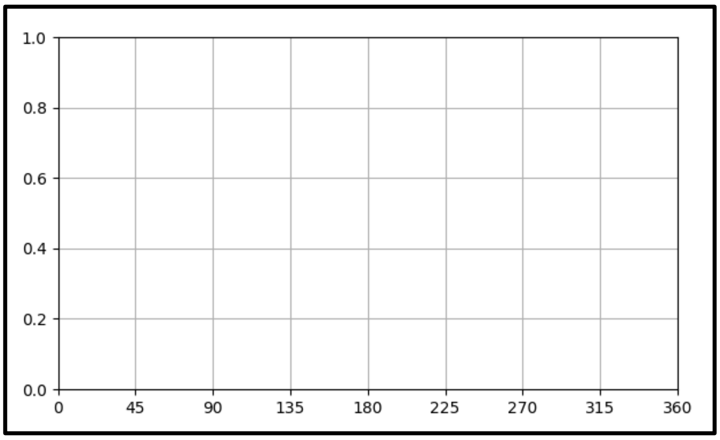
Проблема

Область, в которой нужно интерполировать функцию – круг. Если использовать билинейную интерполяцию напрямую в том виде, в котором она дана выше, то нам не удастся разбить круг прямоугольниками одинаковых размеров.

Решение проблемы

Решение проблемы – полярные координаты. Из теории (связь между декартовыми и полярными координатами) мы знаем, что, при переходе из декартовых координат в полярные, окружность переходит в прямоугольник.





Переход окружности с радиусом 1 из декартовых координат в полярные. Также, тут видно, как элементарные секторы круга из декартовых координат переходят в полярные и становятся прямоугольниками.

Интерполяция проводится в цилиндрических координатах. То есть, в формуле выше вместо x и y подставляются φ и r, соответственно.

Построение тестового примера

Выбор функции

Для начала выберем функцию, например, $f(x,y) = x^2 + y^2$(параболоид)

Исходный график функции:

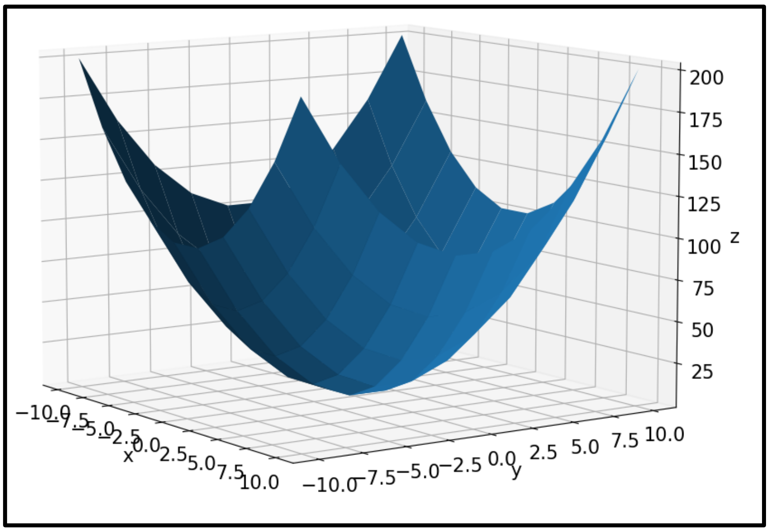


Рисунок 2(график исходной функции в декартовых координатах)

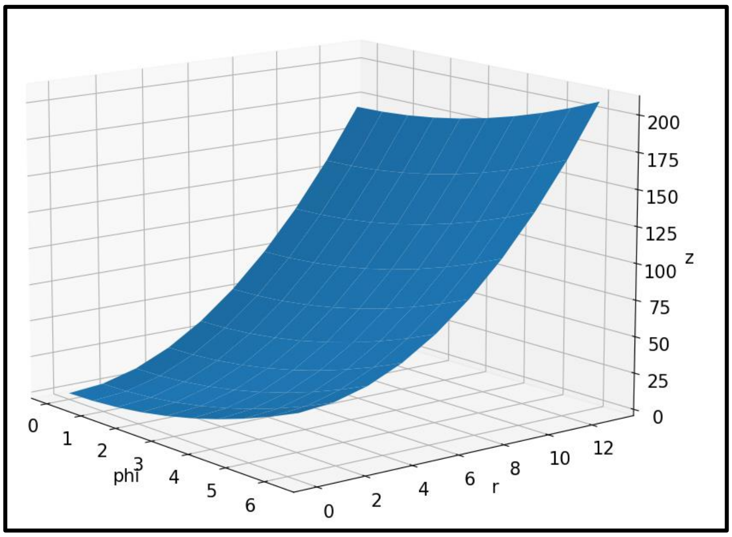


Рисунок 3(график исходной функции в цилиндрических координатах)

Выбор параметров разбиения

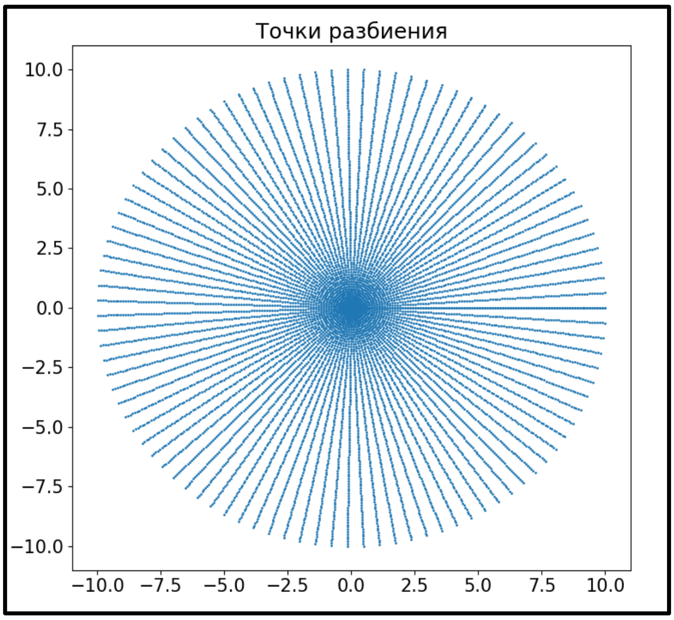
Зададим радиуса круга – 10, разбиение: φ на 100 частей и *r* на 100 частей:

```
R = 10
cnt_dr, cnt_dphi = 100, 100
```

Получение разбиения

Разбиение выполняется с помощью процедуры *get_points*:

```
polar_points = get_points(cnt_dphi, cnt_dr, R)
```



Выходное значение данной процедуры – 2-мерный список, у которого в пределах одной строки постоянное значение радиуса, в пределах одного столбца – постоянное значение угла.

Тут стоит сказать, что разбиение угла выполняется от 0 до 2π включительно, это нужно, чтобы построить график непрерывной функции, а также для построения элементарных областей (более подробно см. в коде).

В связи с этим, точки с радиусом ноль попадут в список ровно 100 раз (т.к. делим на 100 частей), хотя по сути это одна и та же точка – это уже издержки моей программы (но каких-то особых проблем это приносит, разве что, тратится лишняя память и время на подсчёт этих точек). Более подробно о работе процедуры *get_points* см. в комментариях в коде.

Получение таблично-заданной функции

Далее, нам нужно получить таблично-заданную функцию *f* (ведь именно таким образом по условию задания происходит интерполяция). Для этого вызовем процедуру *get_func_values*:

```
func_values = get_func_values(polar_f, polar_points)
```

На вход эта процедура принимает процедуру, вычисляющую значение интерполируемой функции по полярным координатам (сначала происходит перевод в декартовы координаты, а затем подсчёт значения функции), и 2-мерный список точек, заданных в полярных координатах (мы его получили раннее)

Выходным значением этой процедуры является 2-мерный список значений (конфигурация списка аналогична конфигурации списка, который возвращает процедура *get_points*).

Интерполяция

На этом этапе все необходимые данные для интерполяции подготовлены, можем начать её проводить. Для этого вызовем процедуру *get_areas*:

```
func_areas = get_areas(cnt_dphi, cnt_dr, polar_points,
func_values, bi_linear_polynom)
```

На вход эта процедура принимает количество разбиений угла и радиуса, 2-мерный список полярных точек, таблично-заданную функцию *f*, которую мы получили раннее и процедуру вычисления билинейного многочлена, формула которого приводилась выше.

Выходным значением этой процедуры является 2-мерный список областей (конфигурация списка аналогична конфигурации списка, который возвращает процедура *get_points*). В моей программе область – это класс, состоящий из 3-ёх полей: 2 точки области (левая верхняя и правая нижняя, если смотреть на разбиение в декартовых координатах, или левая нижняя и правая верхняя в полярных, соответственно), заданные в полярных координатах и процедуру для вычисления значения интерполяции в заданной области (более подробно о работе этой процедуры см. комментарии в коде).

Получение готовой процедуры интерполяции

Это финальный этап интерполирования, здесь мы получаем готовую процедуру интерполяции, то есть мы можем взять точку, в которой хотим вычислить значение интерполяции, подать её процедуре интерполяции и получить значение. Для этого сконструируем класс *Interpolated_func*:

```
interpolated = Interpolated_func(func_areas)
```

Этот класс стоит всего из одного поля - 2-мерный список областей, в которых можно вычислять значение(этот список был получен выше), его мы и передаём процедуре инициализации, также этот класс содержит процедуру *evaluate*, которая непосредственно и отвечает за подсчёт значения интерполяции, но это процедура должна понять, в какой области находится точка (то есть, к какому эл-ту 2-мерного списка *func_areas* нужно обратиться, чтобы вызвать корректную для данной области процедуру интерполяционного многочлена), чтобы это понять, нужно вызвать процедуру *search* – эта процедура найдёт соответствующий эл-т 2-мерного списка и вернёт корректную процедуру для вычисления значения в точке. Если значение точки пришло в процедуру в декартовых координатах, то сначала преобразуем их, используя процедуру *to_polar* (см. более подробно в комментариях в коде)

График

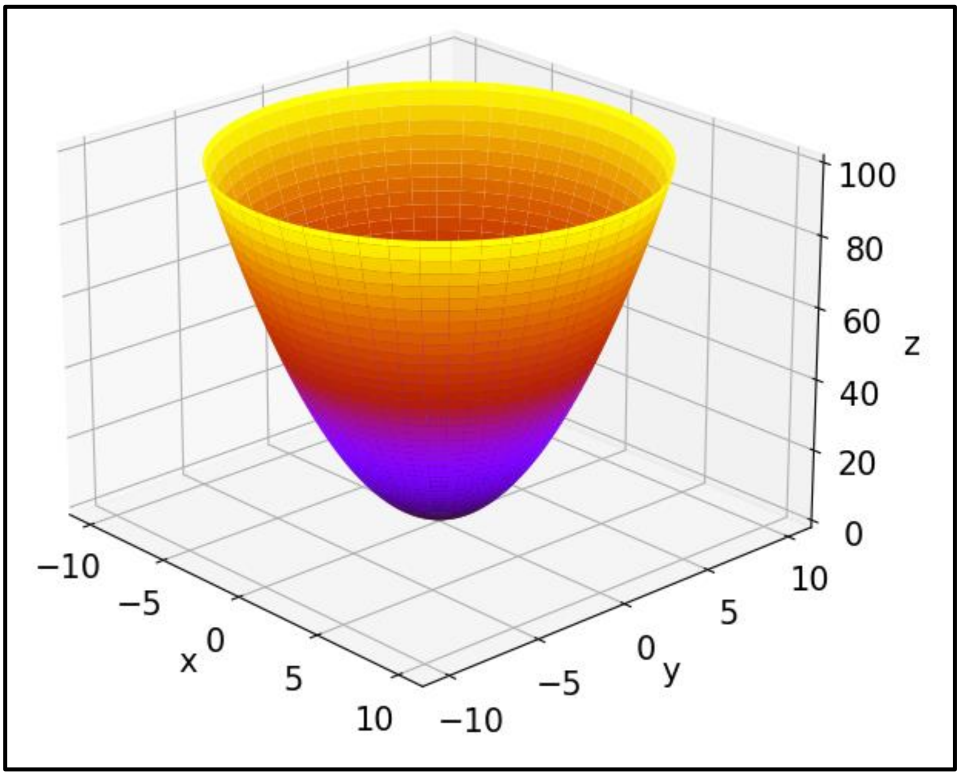


Рисунок 4(график интерполяции, построен по узловым точкам в круге, радиуса 10)

Результат вполне похож на правду, но это лишь только график, давайте посмотрим на график погрешности:

Погрешность

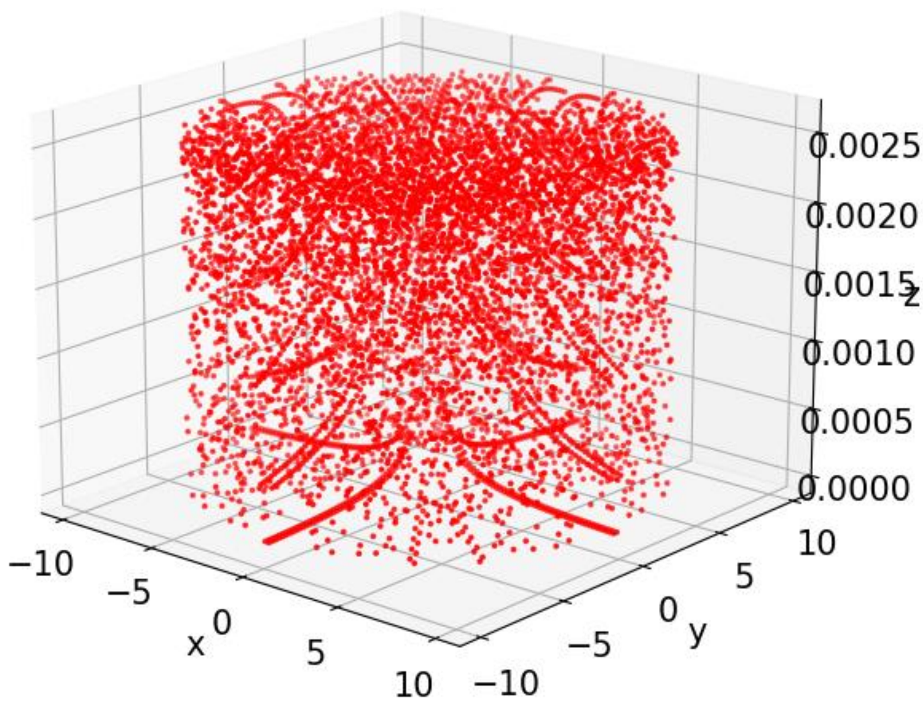


График погрешности построен по декартовым точкам (разбиение от -10 до 10) кол-во точек по каждой оси 100, всего точек 100 * 100

Максимум погрешности: 0.0025507433560250092.

В принципе, вполне неплохой результат для разбиения 100 по радиусу на 100 по углу.

Эксперименты

Пример №1

Функция: $f(x, y) = x * y$

Число разбиений по радиусу и углу: 32 на 15

Исходный график функции:

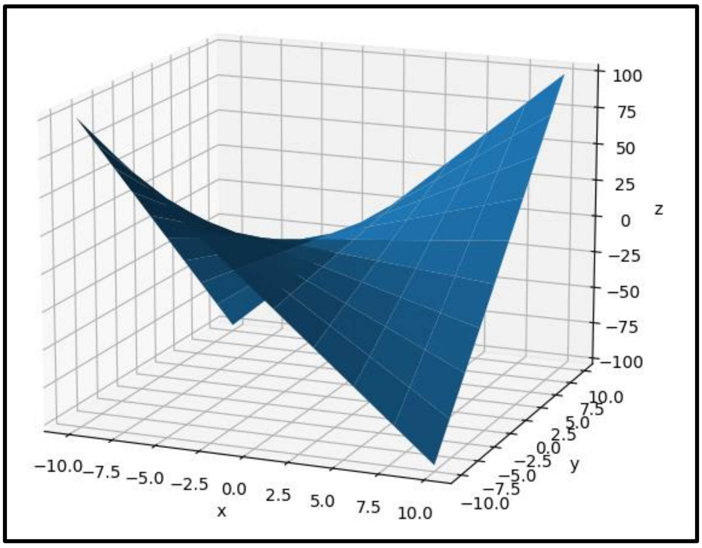


Рисунок 6(график в декартовых координатах)

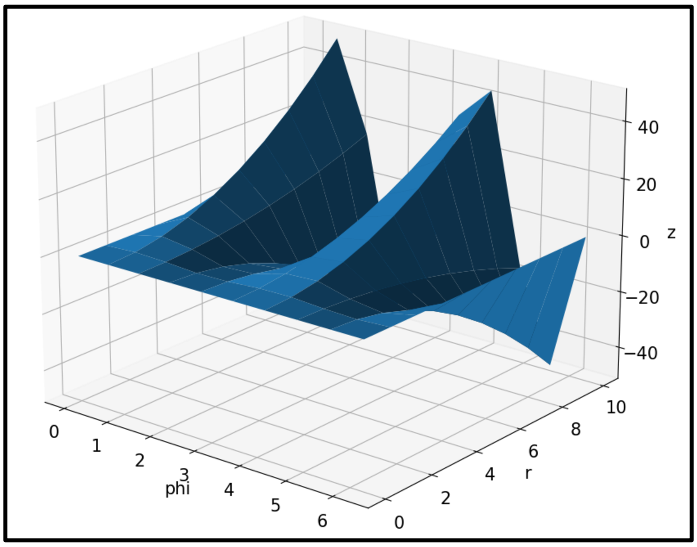


Рисунок 5(график в цилиндрических координатах)

График интерполяции, построенный по узлам в области круга, радиуса 10:

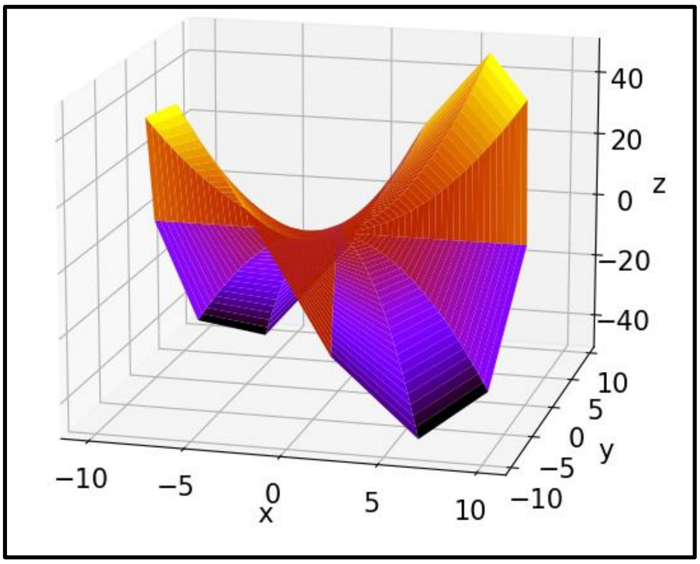


График погрешности:

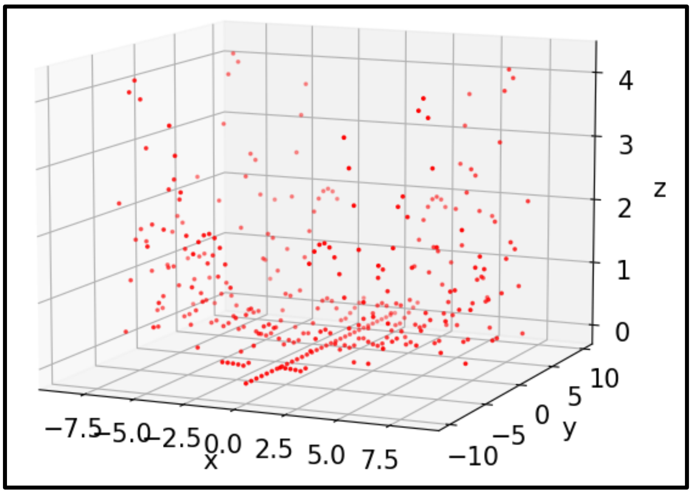


Рисунок 7(погрешность интерполяции при исходном разбиении в точках круга, радиуса 10, на каждой оси было взято 10 точек, всего 100 точек)

Максимум погрешности: 4.180925282645461

Также, приведём график погрешности в зависимости от числа разбиений угла и радиуса(в качестве значений такой функции будем принимать максимум погрешности на данном разбиении):

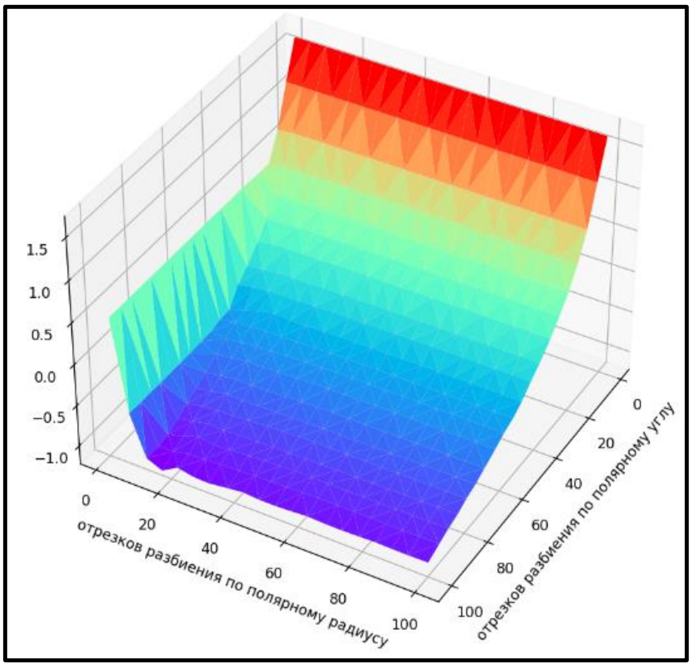


Рисунок 8(График погрешности в зависимости от числа разбиений. Построен в логарифмической шкале(шкала Lg)).

Как видно из графика, погрешность интерполяции зависит от числа разбиений по радиусу лишь при малом его разбиении и сильно зависит от числа разбиений по углу.

Пример №2

Функция: $f(x, y) = x + y$

Число разбиений по радиусу и углу: 32 на 15

Исходный график функции:

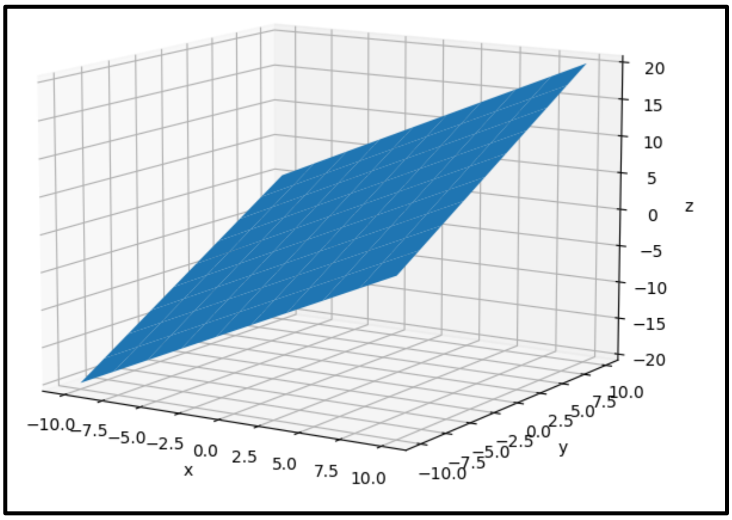


Рисунок 10(график в декартовых координатах)

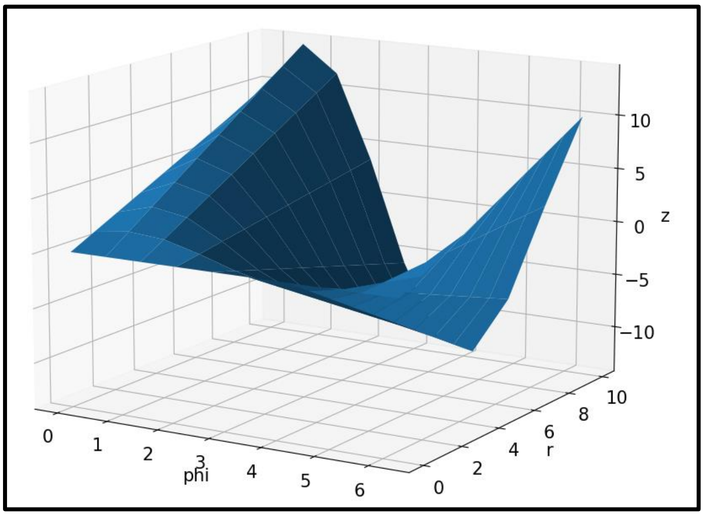


Рисунок 9(график в цилиндрических координатах)

График интерполяции, построенный в области круга, радиуса 10:

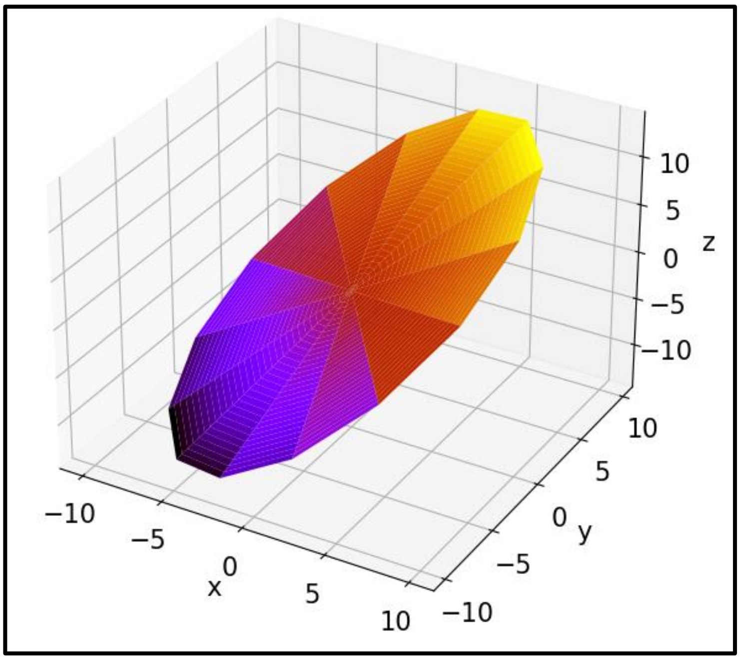


График погрешности:

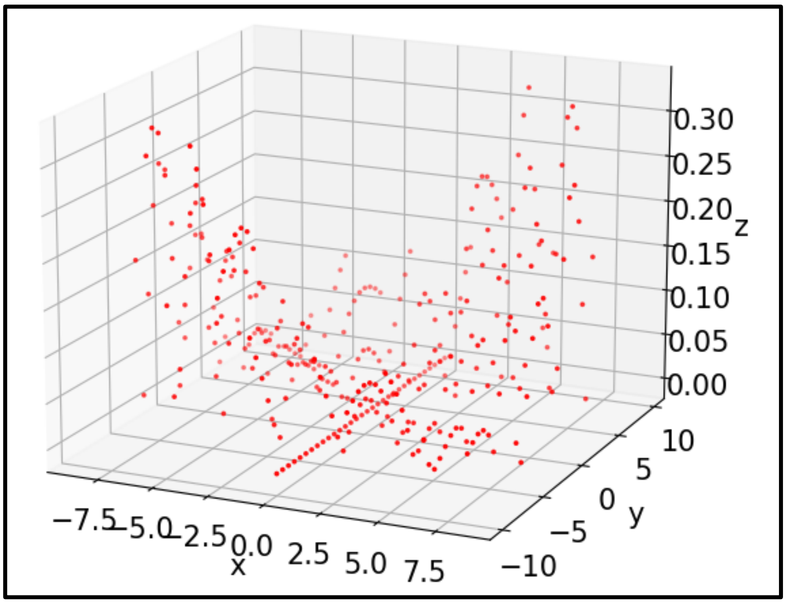


Рисунок 11 (погрешность интерполяции при исходном разбиении в точках круга, радиуса 10, на каждой оси было взято 10 точек, всего 100 точек)

Максимум погрешности: 0.32472496268461626

Также, приведём график погрешности в зависимости от числа разбиений угла и радиуса:

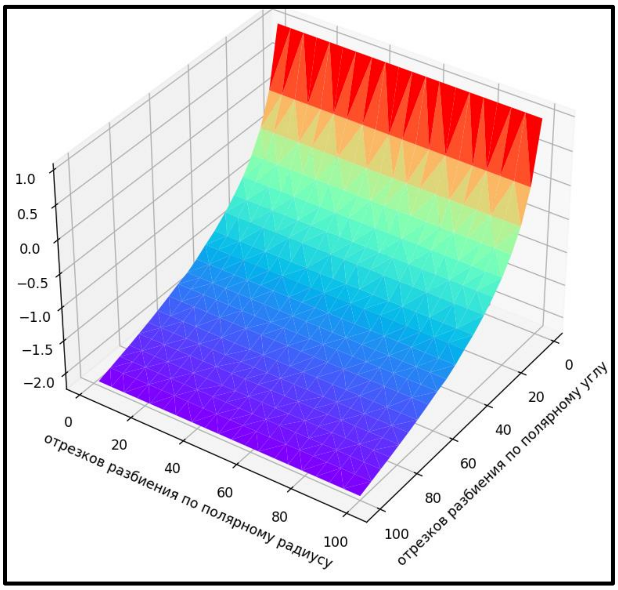


Рисунок 12(График погрешности в зависимости от числа разбиений. Построен в логарифмической шкале(шкала lg)).

Как видно из графика, погрешность интерполяции не зависит от числа разбиений по радиусу, но зависит от числа разбиений по углу. Это объясняется тем, что в полярном представлении исходная функция выглядит так: $f = x + y = r(\cos(\varphi) + \sin(\varphi))$. Мы видим, что радиус входит в выражение линейно, отсюда и независимость от числа разбиений по радиусу при билинейной интерполяции в цилиндрических координатах.

Объяснение результатов тестовых примеров (1 и 2):

Как видно из 1-ого примера, погрешность довольная высока (больше 4 при разбиении 32 на 15), но поверхность достаточно сложная, да и разбиение не очень большое. Если сделать разбиение, например, 100 по радиусу и 100 по углу, то максимальная погрешность уменьшится до ≈0.097 – вполне неплохо.

Что касается тестового примера 2: исходная функция линейная по обеим переменным *x* и *y*, поэтому на первый взгляд должна очень хорошо приближаться билинейным многочленом, но погрешность говорит об обратном и дело тут скорее всего не в методе вычислений. Такой результат выходит из-за того, что исходная функция задана в декартовых координатах, а приближаем мы её с помощью цилиндрических.

Рассмотрим эту гипотезу более наглядно: построим график интерполяции при малом разбиении, так мы сможем увидеть, в каким местах исходной функции, заданной в декартовых координатах, высокая погрешность:

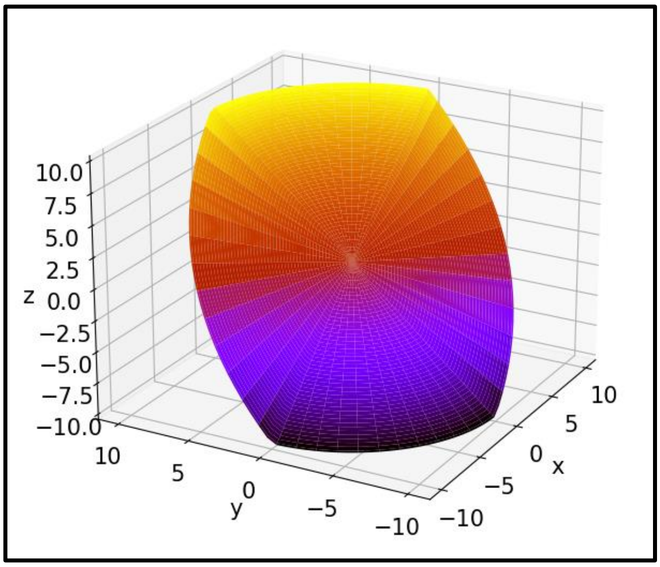
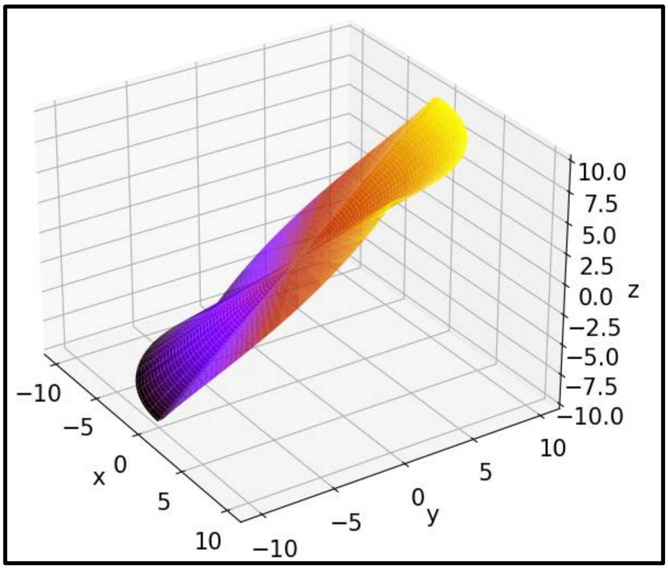


Рисунок 13(график интерполяции функции f(x, y) = x + y, построенный при разбиении 2 по радиусу и 4 по углу.

График был построен на всей плоскости xOy, изначально было взято 100 * 100 точек в полярных координатах (100 по полярному углу и 100 по полярному радиусу) и преобразовано в декартовы.

Из графика уже видно, что с нашей функцией явно происходит что-то плохое. Взглянем на график с другого ракурса:



Мы видим искривление функции в первой и третьей четвертях исходной области(круга)(это можно заметить, если посмотреть на оба графика и мысленно нарисовать на плоскости xOy круг). Как раз в местах искривления и выходит такая высокая погрешность. График погрешности:

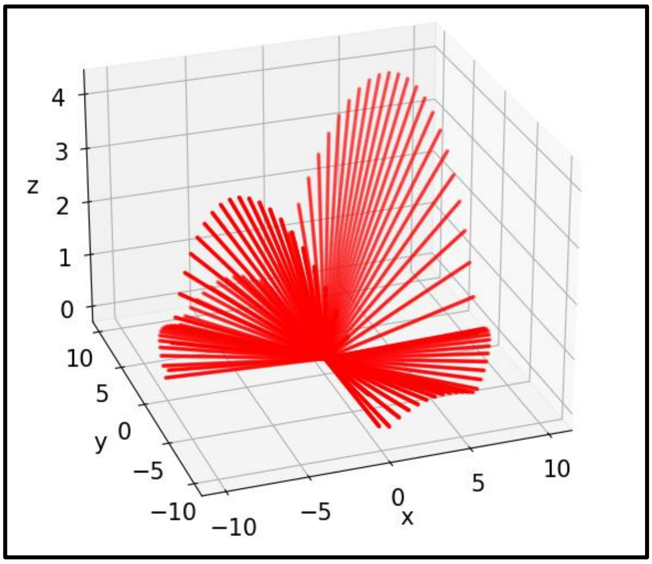
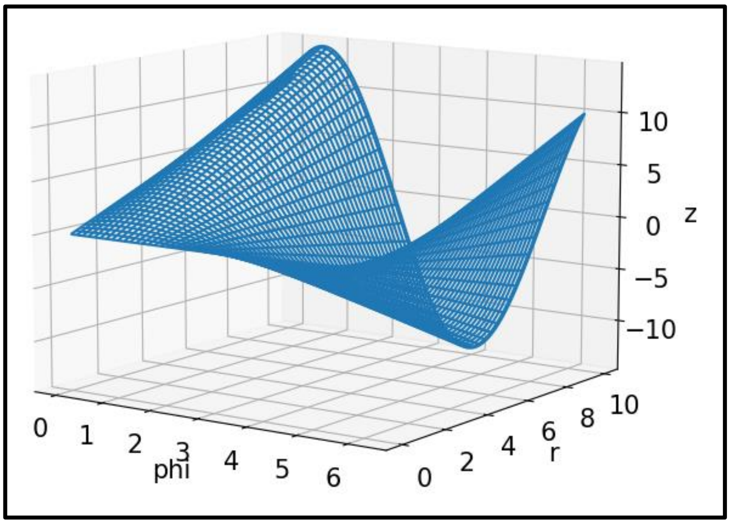


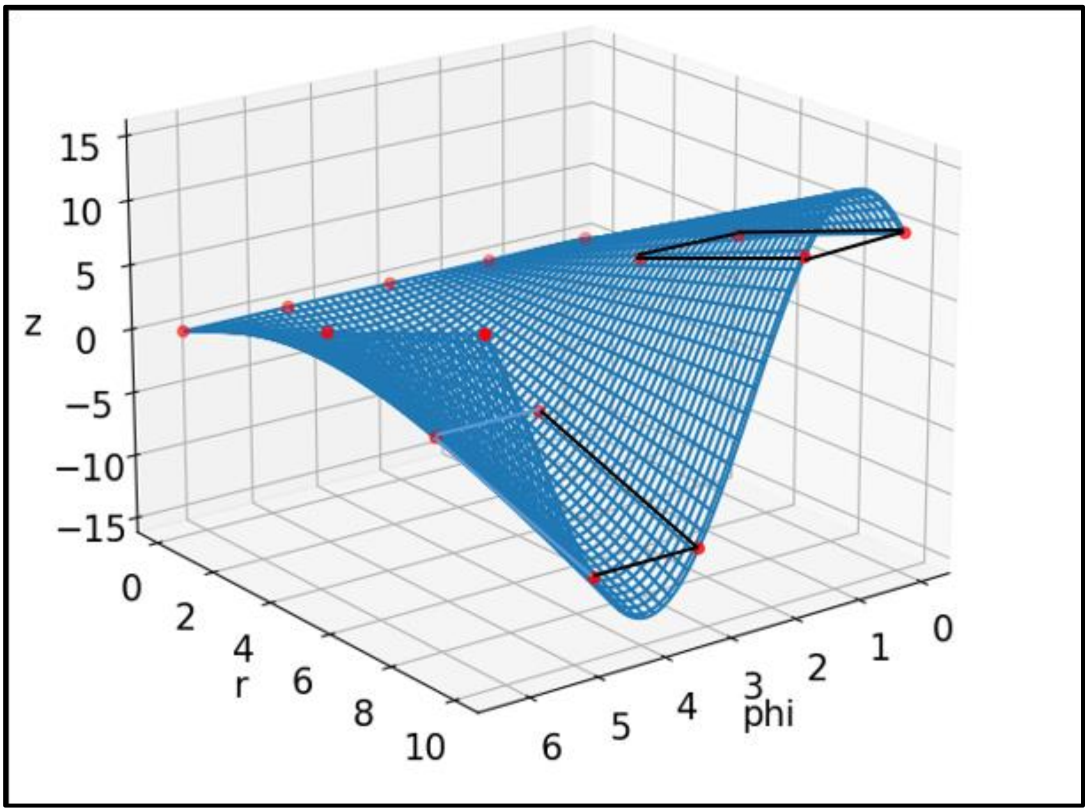
Рисунок 14(погрешность интерполяции при исходном разбиении в точках круга, радиуса 100, на каждой оси было взято 100 точек, всего 10000 точек)

Из графика погрешности убеждаемся, что проблема как раз в первой и третьей четвертях.

Давайте ещё раз посмотрим на график исходной функции (неинтерполированной) в цилиндрических координатах:



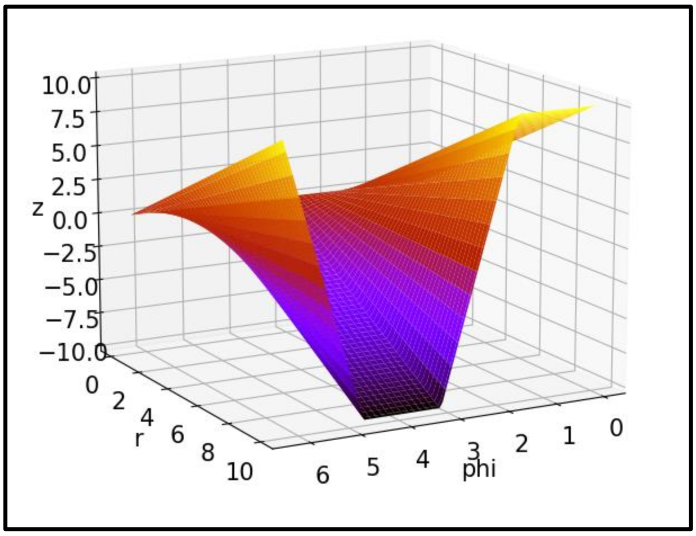
Кажется, гипотеза подтверждается. Поясню чуть подробнее, но сперва тот же график, но с другого ракурса, и при этом с узлами интерполяции на нём (красные точки):

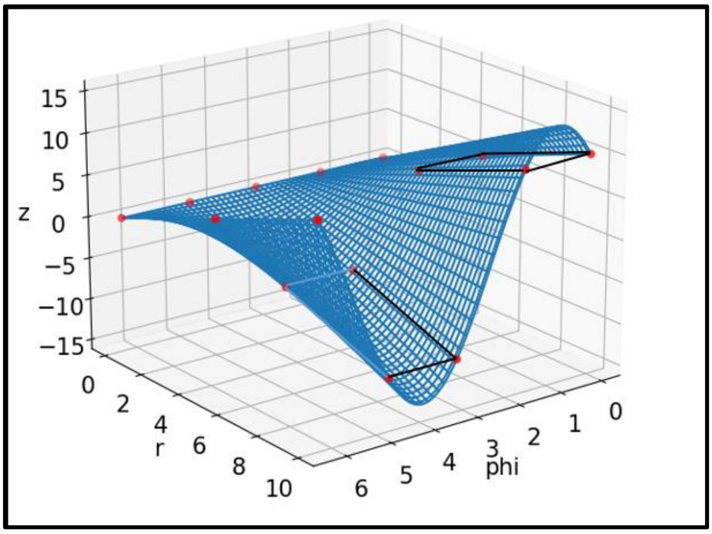


Как мы видим, график исходной функции из декартовых координат перешёл в график поверхности с “горками” и “впадинами” (локальным максимумом для горки и минимум для впадины) в цилиндрических. Всё дело именно во впадинах. На графике я специально отметил узлы интерполяции и соединил некоторые узлы прямоугольниками (ведь у нас билинейная интерполяция, в начале мы договорились, что переводим область в полярные координаты, чтобы можно было применять билинейную интерполяцию, то есть интерполировать прямоугольниками), отмеченные прямоугольники – это то, как произойдёт интерполяция. Например, взглянем на прямоугольник, который ближе всего расположен к началу координат по оси φ, мы видим, как грубо прямоугольник аппроксимирует “горку” - покатая часть поверхности заменяется на плоскую часть, и эта грубо аппроксимированная часть как раз находится в первой четверти – отсюда такая высокая погрешность. Аналогично для второго прямоугольника (он, кстати, расположен в третьей четверти).

К тому же, если функция в цилиндрическом представлении будет иметь более высокую скорость изменения 1-ой производной (“горки” и “впадины” будут более крутыми), то интерполяции будет ещё хуже, т.к. прямоугольники ещё грубее будут аппроксимировать эти проблемные участки.

А теперь продемонстрирую график интерполированной функции рядом с графиком выше:





Слева график интерполированной функции, справа график исходной функции (построены оба графика в цилиндрических координатах). График слева лишь подтверждает слова выше о том, как грубо интерполируются “горки” и “впадины” исходной функции.

Выход из такой ситуации – это более точное разбиение или интерполирование многочленом более высокого порядка.

Попробуем провести ещё эксперимент: зададим функцию сразу в цилиндрических координатах, линейную по r и φ (аналог функции $f(x, y) = x + y$, но уже в цилиндрических координатах): $f(\varphi, r) = r + \varphi$ и посмотрим, что выйдет

График функции:

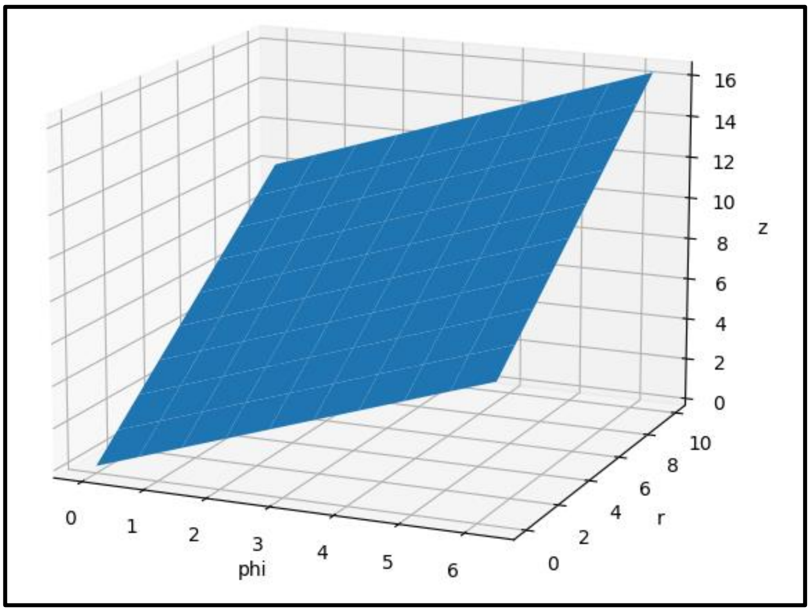


График интерполяции, построенный по узлам интерполяции:

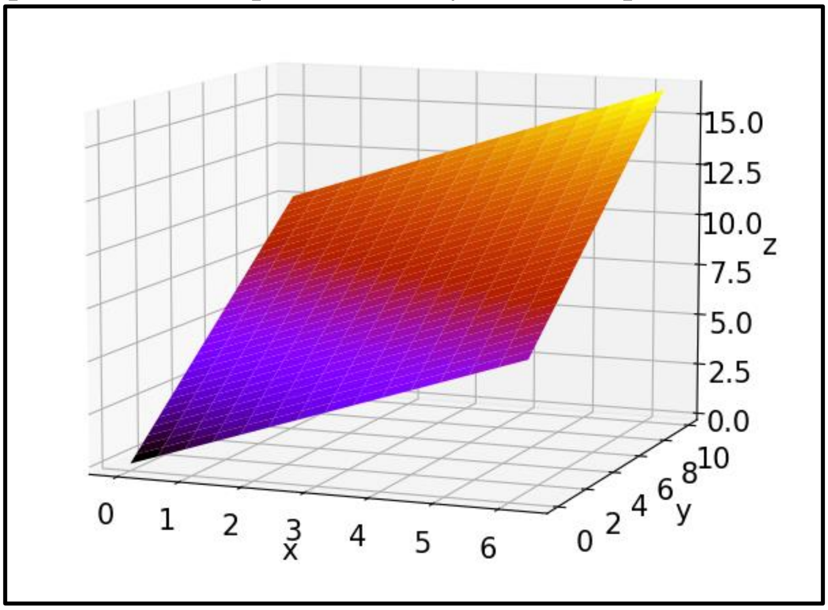


График погрешности:

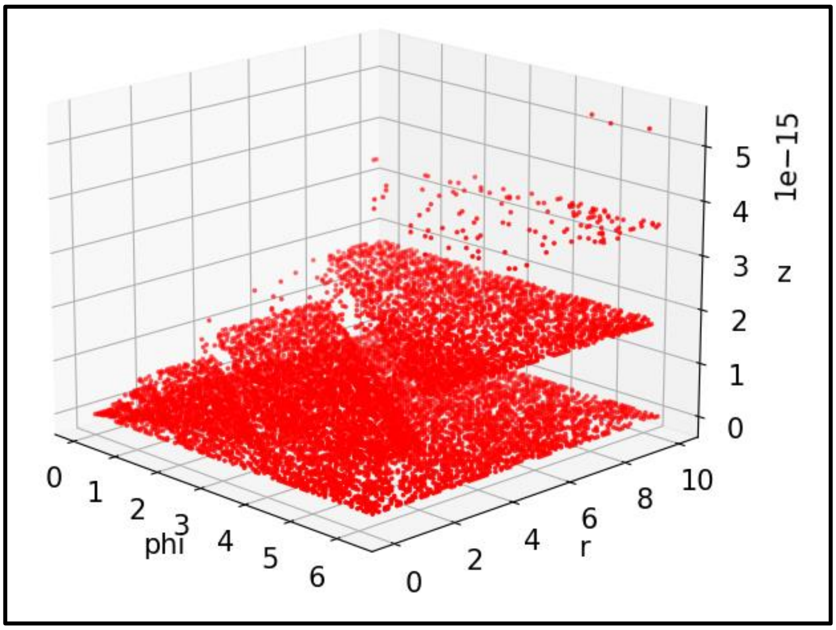


Рисунок 15(погрешность интерполяции при исходном разбиении в точках прямоугольника 6 на 10, на каждой оси было взято 100 точек, всего 10000 точек)

Максимальная погрешность: 5.329070518200751e-15

Видим достаточно хороший результат, а значит билинейная интерполяция(проще говоря, интерполяция прямоугольниками) работает для линейной функции (линейной по каждой переменной φ и r).

Подытожим: в ходе такого эксперимента мы выяснили, почему линейная функция по x и y приближается плохо с помощью нашего метода, также выяснили, что линейная функция в цилиндрических координатах будет приближаться очень хорошо. В целом, можно сделать такой вывод: билинейная интерполяция в цилиндрических координатах тем лучше, чем исходная функция в цилиндрическом представлении имеет меньше локальных минимумов/максимумов и чем меньше скорость изменения 1-ой производной.

Приведём график погрешности в зависимости от числа разбиений угла и радиуса (в качестве значений такой функции будем принимать максимум погрешности на данном разбиении

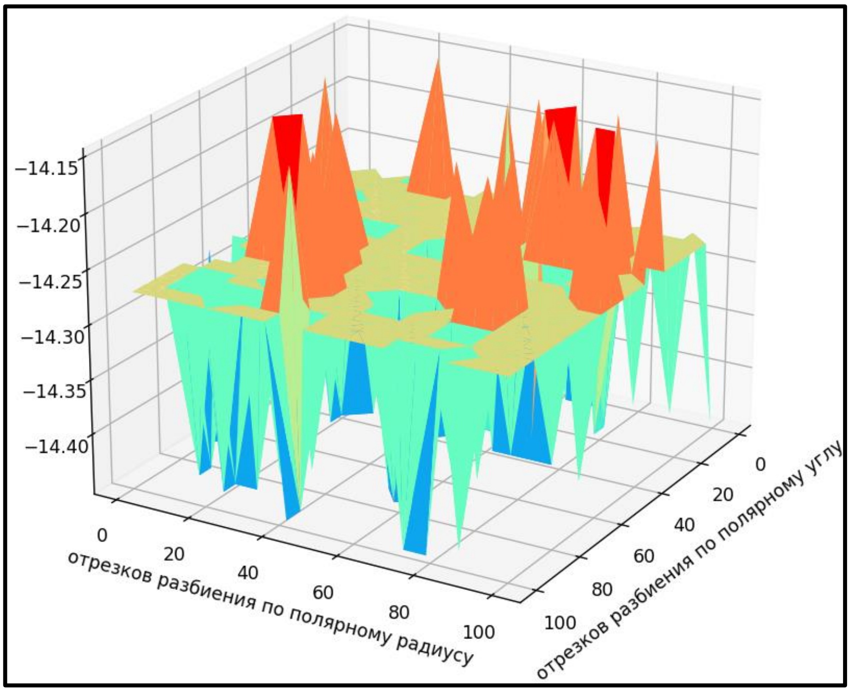


Рисунок 16(график погрешности в зависимости от числа разбиений, приведён в шкале lg)

Конечно, погрешность скачет от разбиения к разбиению, но при этом находится в допустимых пределах (не превышает 10^{-14}), при этом размер скачков очень маленький($10^{-14.15} - 10^{-14.40} \approx 3 \times 10^{-15}$), так что можно полагать, что функция не зависит от разбиения (такого результата и стоило ожидать, потому что исходная функция линейная по φ и r).

Пример №3(пример, у которого погрешность зависит от обоих параметров разбиения)

Функция: $f(x,y) = (x^2+y^2)^3 \cdot y$

Исходный график функции:

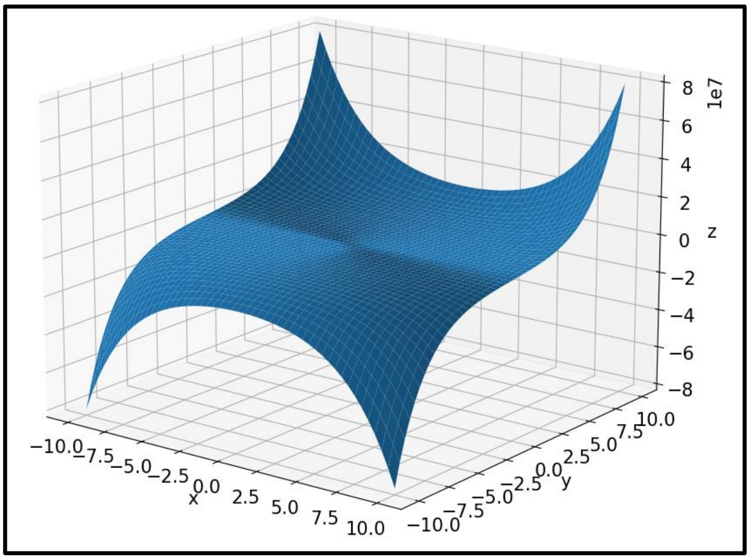


Рисунок 18(график в декартовых координатах)

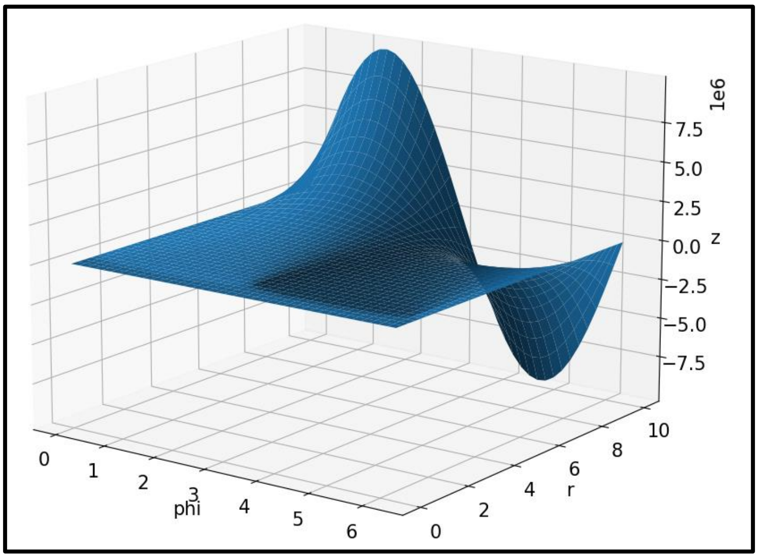


Рисунок 17(график в цилиндрических координатах)

Это функция изначально плохая для интерполяция билинейным многочленом в цилиндрических координатах, потому что имеет те самые горки и впадины (минимумы и максимумы), причём перепады с минимума на максимум достаточно “крутой”. Из примера выше мы убедились, что такие функции наш метод будет интерполировать плохо.

Сразу приведём график погрешности в зависимости от числа разбиений угла и радиуса (в качестве значений такой функции будем принимать максимум погрешности на данном разбиении):

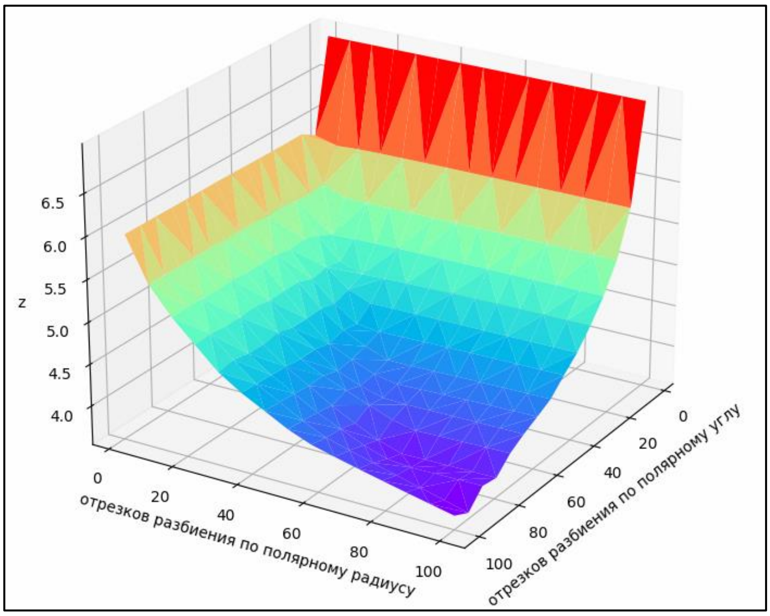


Рисунок 19(график погрешности в зависимости от числа разбиений, приведён в шкале lg)

Как мы видим, погрешность сильно зависит от числа разбиений по углу и от числа разбиений по радиусу, действительно, в полярном представлении эта функция выглядит так: $f = (x^2 + y^2)^3 \cdot y = r^7 \cdot \sin(\varphi)$, видим степенную зависимость от радиуса (радиус аж в 7-ой степени) и синусоидальную по углу.

Функция проинтерполировалась достаточно плохо (минимальная погрешность порядка 10^4). Как уже упоминалось выше, наш метод не пригоден для подобно рода функций.

Пример№4 (зависимость от скорости изменения 1-ой производной (зависимость от 2-ой производной))

Немного изменим прошлый пример:

Функция: $f(x, y) = (x^2 + y^2)^3 \cdot y^2$

Исходный график функции:

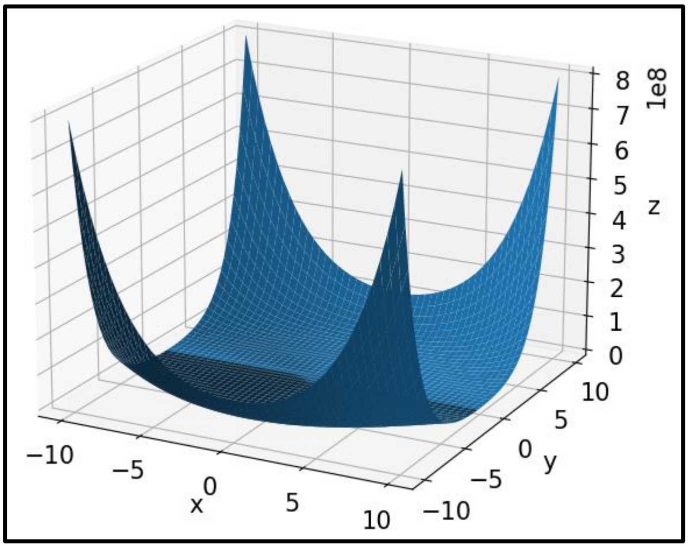


Рисунок 21(график в декартовых координатах)

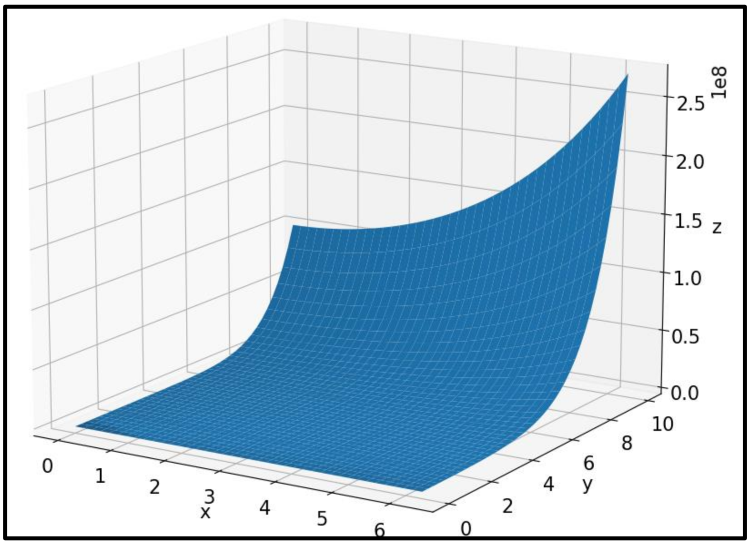


Рисунок 20(график функции в цилиндрических координатах)

Мы увеличили степень y до второй.

Выше мы говорили о скорости изменения первой производной, скорость изменения первой производной – вторая производная. Сравним вторую производную по r прошлого и текущего примера:

$$f1 = (x^2 + y^2)^3 \cdot y = r^7 \cdot \sin(\varphi),\ f1''_r = 42r^5 \sin(\varphi)$$

$$f2 = (x^2 + y^2)^3 \cdot y^2 = r^8 \cdot \sin(\varphi)^2,\ f2''_r = 56r^6 \sin(\varphi)^2$$

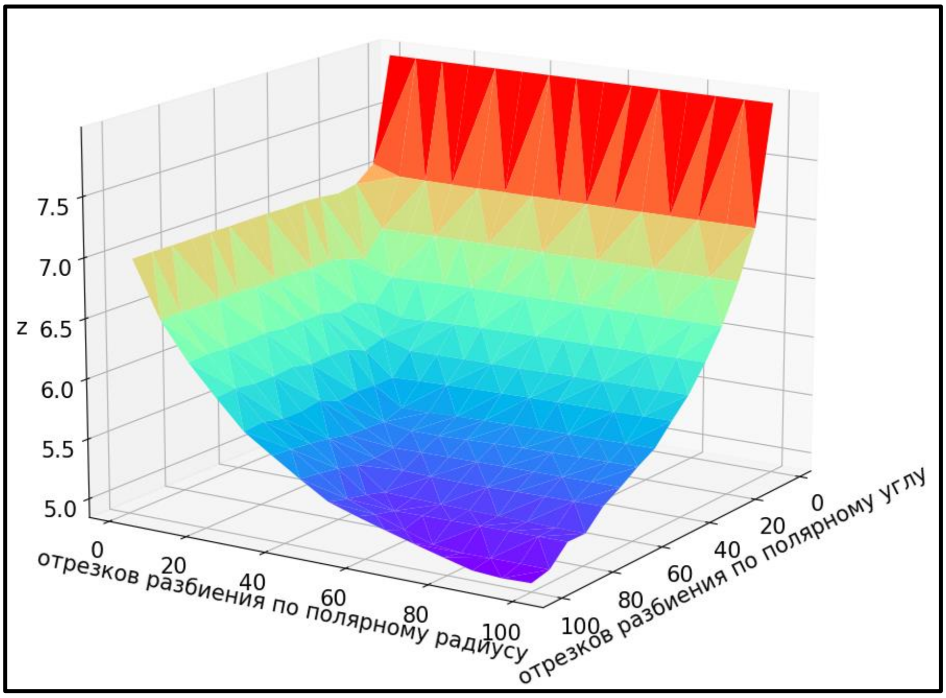
Сравним порядки производных, взяв предел при $r \to +\infty$

$$\lim_{r\to+\infty} \frac{f1'_r}{f2'_r} = \lim_{r\to\infty} \frac{42r^5 \sin(\varphi)}{56r^6 (\sin(\varphi))^2} = \frac{42}{56 r sin(\varphi)} = 0\ (*)$$

Из предела (*) следует, что порядок второй производной функции $f2$ выше порядка второй производной функции $f1$, а это значит, что скорость изменения первой производной функции $f2$ выше скорости изменения первой производной функции $f1$, а значит к такой функции наш метод более чувствителен (погрешность будет выше).

Предел со вторыми производными по φ брать не будем, потому что он будет не определён (т.к. sin и cos буду принимать всевозможные значения из их области определения).

Сразу приведём график погрешности в зависимости от числа разбиений угла и радиуса (в качестве значений такой функции будем принимать максимум погрешности на данном разбиении):



Видим более высокую погрешность, а это значит, что наши рассуждения верны.

Вывод:

В данной работе мы познакомились с методом интерполирования билинейными многочленами, построили программу интерполяции функции в круге с помощью цилиндрических координатах. Провели эксперименты с разными функциями. Пришли к выводу, что плоскости, заданные в декартовых координатах, интерполируются плохо, но если задавать их в цилиндрических координатах, то будет намного лучше. С помощью экспериментов пришли к выводу о зависимости погрешности интерполяции от скорости изменения первой производной.

Литература

1. Амосов А. А., Дубинский Ю. А., Копченова Н. В. Вычислительные методы: Учебное пособие. - 4-е изд., стер. - СПб.: Издательство Лань, 2014.
2. Билинейная_интерполяция

Приложение

Ссылка на код: код