МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ
Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра математического анализа

Приближение выпуклых множеств многогранниками

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы

направление 02.03.01 — Математика и компьютерные науки

механико-математического факультета

Ханбикова Динислама Марсельевича

Научный руководитель
доцент. к.ф. -м. н. М.А.Осипцев

Зав. кафедрой
и.о. зав. кафедрой, д.ф.-м.н. П.А.Терехин

Саратов 2024

Введение.В различных разделах математики возникают задачи, решение которых требует провести аппроксимацию различных объектов со сложной геометрией набором более простых однотипных фигур. В частности такие задачи возникают в теории управления, дифференциальных играх, а также приложениях транспортной логистики и экономики. С конструктивной точки зрения задача аппроксимации многогранниками состоит в следующем. По заданному выпуклому телу построить многогранник с n вершинами или гранями, приближающий тело наилучшим образом в той или иной метрике. Построить — означает выписать либо координаты вершин, либо систему линейных неравенств. В такой постановке задача решается очень редко и только в плоском случае. Например, для круга это будут правильные многоугольники.

Всего в работе 4 главы: «Аппроксимация многоугольников кругами равного радиуса», «Построение оптимальной упаковки множества в пространстве $\mathbb{E}^3$», «Приближение выпуклого тела шаром произвольной нормы с фиксированным радиусом», «Программа для нахождения наилучшей сети треугольника при $n = 2$ или $n = 3$ (python)».

Целью данной работы является изучение подходов к решению задач аппроксимации выпуклых множеств многогранниками:

- изучить методы решения данной задачи для плоских фигур;
- изучить метод решения задачи о поиске чебышевского центра множества для выпуклых многогранников и рассмотреть алгоритм поиска;
- познакомиться с алгоритмами построения упаковок для аппроксимации множеств в трехмерном евклидовом пространстве;
- изучить вопрос об аппроксимации выпуклого тела шаром произвольной нормы с фиксированным радиусом;
- разработать и отладить программу построения наилучших кругов для приближения треугольников в случае пространства $\mathbb{R}^2$ с помощью полученных теоретичеких результатов.

Основное содержание работы. Изначально нам надо ввести несколько определений и напомнить некоторые сведения из выпуклого анализа, которыми мы будем пользоваться в дальнейшем. Позже мы рассмотрим различные задачи, формулировка и терминалогия которых опираются на определения и

свойства, полученные в первом разделе "Некоторые сведения из выпуклого анализа".

Определение 1.2.(Метрика Хаусдорфа). Пусть множества X и $Y-$два непустых подмножества метрического пространства. Определим метрику Хаусдорфа, полагая:

$$h(X,Y) = \max\{\sup_{x\in X} d(x,Y),\ \sup_{y\in Y} d(y,X)\},$$

где $d(a,B)-$расстояние между a и множеством B.

Определение 1.8. (Функции расстояния) . Пусть $D \subset \mathbb{R}^p-$выпуклый компакт, $\Omega = \overline{\mathbb{R}^p} \setminus D$, $n(x)-$норма на $\mathbb{R}^p$.

1) $R(x,D) = \max\limits_{y\in D} n(x-y)-$расстояние от x до наиболее удаленной точки из D.

2) $\rho(x,\Omega) = \min\limits_{y\in\Omega} n(x-y)-$расстояние от x до ближайшей точки из Ω.

3) $P(x,D) = \rho(x,D) - \rho(x,\Omega)-$расстояние от x до границы D, причем, если $x \in D$, то со знаком минус.

Известно , что $R(x,D)$, $\rho(x,D)$, $P(x,D)-$выпуклые функции на $\mathbb{R}^p$, $\rho(x,\Omega)-$вогнутая функция на D.

Определение 1.10. (Субдифференциал функции). Множество субградиентов в точке x_0 называется субдифференциалом выпуклой собственной функции $f(x)$ и обозначается $\partial f(x_0)$. Это записывается:

$$\partial f(x_0) = \{x^* : f(x) - f(x_0) \geq \langle x - x_0, x^*\rangle,\ \forall x\}$$

Определение 1.11.(Чебышёвский центр множества). Чебышёвским центром ограниченного множества $M \subset \mathbb{R}^2$ называется такая точка $\mathbf{c}(M)$, что

$$\sup_{\mathbf{x}\in M} \|\mathbf{x} - \mathbf{c}(M)\| = \sup_{\mathbf{x}\in M} \inf_{\mathbf{y}\in\mathbb{R}^2} \|\mathbf{x} - \mathbf{y}\|$$

Теорема 1.6. Пусть $M-$треугольник с вершинами $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ и длинами сторон $A_1 = \|\mathbf{a}_2 - \mathbf{a}_3\|$, $A_2 = \|\mathbf{a}_1 - \mathbf{a}_3\|$, $A_3 = \|\mathbf{a}_2 - \mathbf{a}_3\|$. Если выполняется

оценки

$$A_1^2 \geq A_2^2 + A_3^2$$

и

$$A_2 \geq A_3,$$

то

$$S = \{\mathbf{x}_i\}_{i=1}^2 = \{(\mathbf{g} + \mathbf{a}_i)/2\}_{i=1}^2$$

является наилучшей 2−сетью для M. Здесь $\mathbf{g}$−пересечение прямой, содержащей отрезок $[\mathbf{a}_1, \mathbf{a}_3]$, со срединным перпендикуляром Λ к отрезку $[\mathbf{a}_1, \mathbf{a}_2]$ (рис. 1.3).

Теорема 1.7. Пусть M — треугольник с вершинами $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ и углами при них $\alpha_1, \alpha_2, \alpha_3$ соответственно. Если величины углов $\alpha_1, \alpha_2, \alpha_3$ лежат на отрезке $[\pi/6, \pi/2]$, то:

$$S = \{\mathbf{x}_i\}_{i=1}^3 = \{(\mathbf{a}_i + \mathbf{c})/2\}_{i=1}^3$$

где $\mathbf{c}$−центр описанной вокруг треугольника окружности, является наилучшей 3−сетью для множества M.

Теперь сформулируем задачу об аппроксимации выпуклого тела шаром произвольной нормы с фиксированным радиусом.

$$\varphi(x, r) = \max_{\substack{j=\overline{1,k} \\ i=\overline{1,m}}} \{\langle B_j, x\rangle - b_{i_1} - r, b_{i_2} - \langle B_j, x\rangle - r, \langle C_j, x\rangle - c_j + r\} \to \min_{x \in \mathbb{R}^p}, \quad (3.17)$$

где $b_{i_1} = \min\limits_{y \in D}\langle B_i, y\rangle$, $b_{i_2} = \max\limits_{y \in D}\langle B_i, y\rangle$, $c_j = \max\limits_{y \in D}\langle C_j, y\rangle$.

Известным приемом эта задача сводится к задача линейного программирования. Можно сформулировать следующую теорему.

Теорема 3.2. Задача (3.17) эквивалента задаче:

$$\begin{cases} z \to \min \\ \langle B_j, x\rangle - b_{i_1} - r \leq z, \ i = \overline{1, m}, \\ b_{i_2} - \langle B_j, x\rangle - r \leq z, \ i = \overline{1, m}, \\ \langle C_j, x\rangle - c_j + r \leq z, \ j = \overline{1, k}, \end{cases} \quad (3.18)$$

При этом, если x^*−одно из решений задачи (3.17), то $\widehat{x^*} = (x^*, z^*) \in \mathbb{R}^{p+1}$, где $z^* = \varphi(x^*, r)$, одно из решений (3.18). И наоборот, если $\widehat{x^*} = (x^*, z^*)$−одно из решений задачи (3.18), то x^*−одно из решений задачи (3.17), а $z^* = \varphi(x^*, r)$−оптимальное значение целевой функции $\varphi(x, r)$.

Нам нужно создать программу на языке программирования python, в которой будут использоваться теоремы, полученные в работе под номером 1.6 и 1.7

```
1.import math
2.import numpy as np
3.
4.def size_corners(vertices):
5.    # координаты вершин
6.    x1, y1 = vertices[0][0], vertices[0][1]
7.    x2, y2 = vertices[1][0], vertices[1][1]
8.    x3, y3 = vertices[2][0], vertices[2][1]
9.
10.    # длины сторон
11.    a = math.sqrt((x2-x1)**2 + (y2-y1)**2)
12.    b = math.sqrt((x3-x2)**2 + (y3-y2)**2)
13.    c = math.sqrt((x3-x1)**2 + (y3-y1)**2)
14.
15.    # углы в радианах
16.    alpha_rad = math.acos((b**2 + c**2 - a**2) / (2*b*c))
17.    beta_rad = math.acos((a**2 + c**2 - b**2) / (2*a*c))
18.    gamma_rad = math.acos((a**2 + b**2 - c**2) / (2*a*b))
19.
20.    # для удобства сделаем перевод углов из радиан в градусы
21.    alpha_deg = math.degrees(alpha_rad)
22.   beta_deg = math.degrees(beta_rad)
23.    gamma_deg = math.degrees(gamma_rad)
24.
25.    corner = [alpha_deg, beta_deg, gamma_deg]
```

```
26.     return corner
27.
28.
29.def theorem7(vertices):
30.     ax = vertices[0][0]
31.     ay = vertices[0][1]
32.     bx = vertices[1][0]
33.     by = vertices[1][1]
34.     cx = vertices[2][0]
35.     cy = vertices[2][1]
36.     # Центр описанной окружности
37.     d = 2 * (ax * (by - cy) + bx * (cy - ay) + cx * (ay - by))
38.     ux = ((ax * ax + ay * ay) * (by - cy) + (bx * bx + by * by) *
39.     (cy - ay) + (cx * cx + cy * cy) * (ay - by)) / d
40.     uy = ((ax * ax + ay * ay) * (cx - bx) + (bx * bx + by * by) *
41.     (ax - cx) + (cx * cx + cy * cy) * (bx - ax)) / d
42.     # Центры искомых кругов
43.     x1 = (ax + ux)/2
44.     y1 = (ay + uy)/2
45.     x2 = (bx + ux)/2
46.     y2 = (by + uy)/2
47.     x3 = (cx + ux)/2
48.     y3 = (cy + uy)/2
49.     # Радиус r кругов
50.     R = math.sqrt((ux - ax) ** 2 + (uy - ay) ** 2)
51.     r = R/2
52.
53.     return (r, (x1, y1), (x2, y2), (x3, y3))
54.
55.def theorem6(vertices):
56.     # Координаты вершин
57.     x1, y1 = vertices[0][0], vertices[0][1]
58.     x2, y2 = vertices[1][0], vertices[1][1]
```

```
59.     x3, y3 = vertices[2][0], vertices[2][1]
60.     # Длины сторон
61.     a = math.sqrt((x2-x1)**2 + (y2-y1)**2)
62.     b = math.sqrt((x3-x2)**2 + (y3-y2)**2)
63.     c = math.sqrt((x3-x1)**2 + (y3-y1)**2)
64.     # Координаты пересечения серпера бОльшей стороны с другой стороно
65.     # *здесь учитываем |[(x1, y1), (x2, y2)]| >
66.     #|[(x2, y2), (x3, y3)]| > |[(x1, y1), (x3, y3)]|,
67.     # иначе можем переобозначить стороны
68.     x_M = (x1 + x2) / 2
69.     y_M = (y1 + y2) / 2
70.     if y2 == y1:
71.       a11 = y2 - y3
72.       a12 = x3 - x2
73.       a21 = 1
74.       a22 = 0
75.       b11 = x3*(y2 - y3) - y3*(x2 - x3)
76.       b21 = x_M
77.
78.       A = np.array([[a11, a12], [a21, a22]])
79.       B = np.array([b11, b21])
80.       xy = np.linalg.solve(A, B)
81.
82.     else:
83.       k_M = (x2 - x1) / (y2 - y1)
84.       a11 = y2 - y3
85.       a12 = x3 - x2
86.       a21 = -k_M
87.       a22 = -1
88.       b11 = x3*(y2 - y3) - y3*(x2 - x3)
89.       b21 = -k_M * x_M - y_M
90.
91.       A = np.array([[a11, a12], [a21, a22]])
```

```
92.       B = np.array([b11, b21])
93.       xy = np.linalg.solve(A, B)
94.
95.     # Найдем центры искомых кругов
96.     u1, v1 = (x1 + xy[0]) / 2, (y1 + xy[1]) / 2
97.     u2, v2 = (x2 + xy[0]) / 2, (y2 + xy[1]) / 2
98.     # Найдём их радиус
99.     r = math.sqrt(((xy[0] - x1) ** 2) + (xy[1] - y1) ** 2) / 2
100.
101.     return (r, (u1, v1), (u2, v2))
102.
103.# Пример использования
104.vertices = [(0, 0), (13, 0), (4, 9)]
105.
106.# Проверка условий теоремы 7 (величина углов из отрезка)
107.corner = size_corners(vertices)
108.for ex in corner:
109.   if ex >= 30 and ex <= 90:
110.     sharp_corner = 1
111.   elif ex > 90:
112.     sharp_corner = 2
113.     break
114.
115.
116.# Вывод
117.print('Углы треугольника: ')
118.for el in size_cornes(vertices):
119.   print(el, '\t')
120.print("Радиус и координаты центров шаров имеют вид:")
121.if sharp_corner == 1:
122.     print('Наилучшая 3-сеть: ', theorem7(vertices))
123.elif sharp_corner == 2:
124.     print('Наилучшая 2-сеть: ', theorem6(vertices))
```

Пример 1(теорема 7).

```
vertices = [(0, 0), (13, 0), (4, 9)]
Вывод:
Углы треугольника:
68.96248897457819
66.03751102542182
45.0
Радиус и координаты центров шаров имеют вид:
Наилучшая 3-сеть:
(3.48209706929603, (3.25, 1.25), (9.75, 1.25), (5.25, 5.75))
```

Пример 2(теорема 6).

```
vertices = [(0, 0), (20, 0), (4, 7)]
Вывод
Углы треугольника:
96.11550356628543
60.25511870305776
23.62937773065681
Радиус и координаты центров шаров имеют вид:
Наилучшая 2-сеть:  (5.457577873929057, (5.0, 2.1875), (15.0, 2.1875))
```

Комментарии к программе: здесь, я буду обозначать через x. строку, которой буду давать комментарий (x - номер строки).

1. Здесь я портировал библиотеку numpy, NumPy это open-source модуль для python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций.

4-26. Заводим функцию, которая вычисляет углы треугольника в радианах. Для этого используется теорема косинусов (блок "углы в радианах"), результат вернем списком углов в градусах. В блоке "Длины сторон"вычисляем длины (евклидовое расстояние) сторон треугольника.

29-53. Заводим функцию theorem7(vertices), которую вызовем, если углы нашего треугольника удовлетворят условиям теоремы 1.7. Согласно теореме нам нужно найти центр описанной около треугольника окружности, в блоке

"Центр описанной окружности"используется стандартная формула центра в декартовых координатах. Далее согласно теореме нужно найти центры радиусов описанной около треугольника окружности, проведенные к вершинам треугольника. Делаем это в блоке "Центры искомых кругов"с помощью формулы деления отрезка в отношении $1:1$. На конечном шаге вычисляем радиус описанной окружности (евклидовое расстояние) с помощью стандартной формулы и окончательно радиус искомых кругов.

55-101. Тело функции theorem6(vertices), которую вызовем, если выполнятся условия теоремы 1.6. Согласно теореме, нужно найти координаты пересечения серединного перпендикуляра (далее серпер) большей стороны с другой стороной (важно: не с продолжением стороны, а именно с ней самой). При этом учитываем, длины сторон нашего треугольника идут в указанном порядке, иначе можно было бы сделать перевыбор сторон и идти дальше: здесь этот случай не реализован. Рассмотрим подробно.

68-69. Находим центр наибольшей стороны. Теперь находим пересечения серпера с другой стороной как точку пересечения прямых стороны треугольника и проведенного к ней серпера с другой стороны - СЛАУ с 2 неизвестными. При этом здесь учитываются разные случаи.

70-80. Случай, когда серпер является вертикальной в Oxy прямой. Тогда его уравнение очевидно имеет вид $x = x_M$.

82-93. Случай, когда серпер не является вертикальной в Oxy прямой. При этом условие система будет не тривиальной. Находим коэффициент в системе перед x (тангенс угла наклона стороны $[(x1, y1), (x2, y2)]$) - строка 83, тогда коэффициент перед y будет равен -1, а перед x соответственно $-k_M$. Теперь коэффициенты у другой прямой: по уравнению прямой по двум точкам несложно вывести, что коэффициенты перед x и y у прямой будут равны соответственно $y_2 - y_3$ и $x_3 - x_2$, а свободные члены в первом и втором уравнениях соответственно $x_3(y_2 - y_3) - y_3(x_2 - x_3)$ и $-k_M x_M - y_M$

96-97. Находим центры искомых кругов как центр отрезка с концами в одной из вершин треугольника и другой в точке пересечения серпера большей стороны со стороной треугольника.

99. Радиусы находим как половина длины таких отрезков.

В остальных стороках идет стандартная проверка выполнения условий теорем.

106-113. Величины углов из отрезка для теоремы 1.7.

Дальше в зависимости от того, какой результат, используются теоремы. Однако, здесь важно заметить, что программа является достаточной, то есть учитывает и теорему 1.6 и теорему 1.7 в условиях, что поданный на вход треугольник уже либо удовлетворяет теореме 1.6 или теореме 1.7. Это сделанно из соображений компактности кода.

Заключение. В ходе данной работы были предложены методы построения набора кругов на плоскости, которые наилучшим образом в некотором смысле аппроксимируют многоугольники. Сформулированы две теоремы, позволяющие получить точное решение задачи такой аппроксимации для некоторых видов многоугольников. Полученно решение задачи о наилучшем приближении выпуклого тела шаром произвольной нормы с фиксированным радиусом при условии что тело и шар - многогранники. Показано, как свести в этом случае исходную задачу к задаче линейного программирования. Разработа программа на языке python, позволяющая получить точное решение задачи для некоторых типов треугольников при $n = 2$ и $n = 3$. Проведено несколько численных экспериментов, отображающих результаты исследования.