**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**Московский государственный технический университет**

**Имени Н.Э. Баумана**

**(Национальный исследовательский университет)**

**(МГТУ им. Н.Э. Баумана)**

ooxWord://word/media/image3.png

***ОТЧЕТ ПО ЛЕТНЕЙ ПРАКТИКЕ***

***НА ТЕМУ:***

**«Оптимизация параметров клапанных пружин»**

**Выполнил: Рахимгалиев Т.**

**Э2-71Б**

**Проверил: к.т.н., доцент**

**кафедры Э2 Зенкин В. А.**

**Москва, 2020**

**Задание.**

1.Установка необходимого ПО: Python3, VS Code.

2. Освоение основ программирования в Python.

3. Почитать теоретическую часть про парето-оптимизацию и генетические алгоритмы.

4. Прочитать раздел "Газораспределительный механизм" из учебника по конструированию.

5. Собрать в интернете материалы по клапанным пружинам.

6. Изучить параграф “Расчет клапанных пружин”.

7. Реализовать в Python расчет клапанных пружин по учебнику.

8. Сформулировать задачу оптимизации.

9. Реализовать многокритериальную оптимизацию параметров клапанной пружины.

10. Отладить многокритериальную оптимизацию.

11. Оформить отчет.

**Содержание**

Введение………………………………………………………………………………………………………..………………….4

1. Фронт Парето…………………………………………….………………………………………………………………..5

2. Конструкция и расчет клапанных пружин…………………………………………………..……………..7

3. Формулировка задачи оптимизации…………………………………………………..……….……………9

4. Построение фронта Парето для клапанных пружины………………………………...……..…..10

Заключение………………………………………………………………………………………………………………………11

Список используемых источников………………………………………………………………………………….12

Приложения…………………………………………………………………………………………………………………….13

**Введение**

Целью учебной практики является реализация многокритериальной оптимизации параметров клапанной пружины.

Для достижения данной цели потребовалось решить следующие задачи:

* изучить газораспределительный механизм современных ДВС;
* произвести расчет клапанных пружин;
* изучить основы программирования в Python;
* построить фронт Парето для клапанных пружин.

В ходе прохождения учебной практики применялся язык программирования Python версии 3.8

Отчет по практике состоит из введения, 5 параграфов, заключения, списка используемой литературы и приложения.

**Фронт Парето**

Часто возникают ситуации, когда необходимо оптимизировать не одну функцию качества или приспособленности, а сразу множество функций. Например, представим, что инженерстроитель хочет построить идеальное здание. Т.е. он хочет построить недорогое, высотное, устойчивое к землетрясениям и энергоэффективное сооружение. Не правда ли отличная задумка? К сожалению, такое здание не может быть построено.

Каждая из таких оптимизируемых функций называется критерием. Иногда можно найти решение, которое оптимально по всем критериям. Однако гораздо чаще возникает обратная ситуация, когда критерии не согласуются друг с другом. В таких случаях решением является компромисс по различным критериям. Наш инженер-строитель знает, что нельзя построить совершенное здание: недорогое, высотное, устойчивое и «зеленое». Однако он может рассмотреть наилучшие возможные варианты. Существует множество способов определить набор «наилучших вариантов», но наиболее известным является множество Парето в пространстве возможных решений.

Предположим, что рассматриваются два здания, M и N. Говорят, что M доминирует N по Парето, если M не хуже N по всем критериям и хотя бы по одному критерию превосходит N. Если так оно и есть на самом деле, то, действительно, в выборе N нет никакого смысла. Ведь M по всем параметрам не уступает, а по каким-то и выигрывает N. Если рассматривать всего два критерия (Дешевле, Более энерго-эффективный), то на рис. 1 показана область пространства, доминируемая данным строительным решением А. Эта область «замкнута»: элементы на ее границе также доминируемы А.

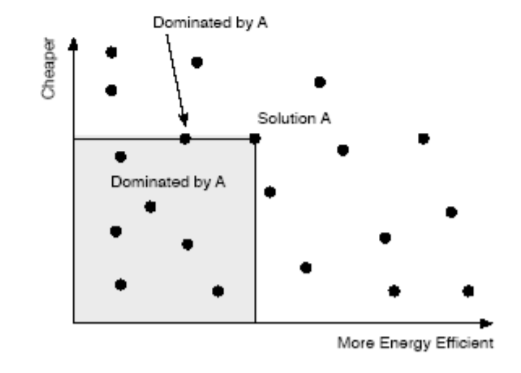


Рис. 1. Область решений, доминируемых по Парето решением А, включая решение на границе. Учтите, что это не изображение пространство фенотипов, а просто результат сравнения по двум критериям

С другой стороны, ни M, ни N не доминируют друг друга, если они равны по всем критериям, либо если N лучше в чем-то одном, а M – в другом. В таких случаях оба решенияM и N представляют интерес для нашего инженера. Поэтому одним из способов определения набора «наилучших вариантов» является набор зданий, которые ничем не доминируемы. Такие строения называются недоминируемыми. Этот набор решений представляет фронт Парето (границу Парето) в пространстве решений. На рис. 47 показана граница Парето для возможных решений в нашем двухкритериальном пространстве. В двумерном случае фронт Парето представляет собой кривую, определяющую нечто вроде внешней границы. В трехмерном случае – это будет что-то, напоминающее оболочку. Если имеется решение, которое превосходит все остальные (эдакий супермен), то фронт сожмется до этого одного решения.

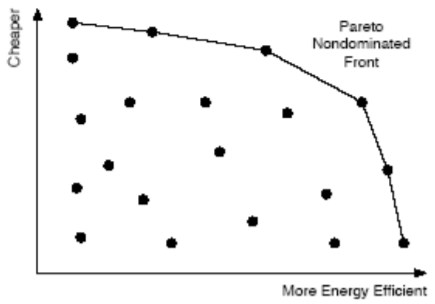
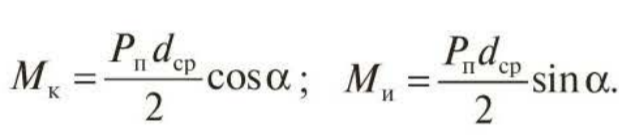


Рис. 2. Граница Парето для недоминируемых решений

**Конструкция и расчет клапанных пружин**

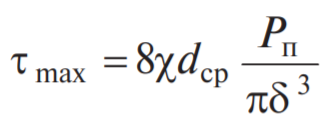
В поршневых двигателях применяются пружины из круглой проволоки. Круглое сечение хорошо работает на кручение, которому подвергаются витки при сжатии пружины. К основным параметрам цилиндрической пружины относятся диаметр проволоки δ, индекс пружины с = dср / δ (dср - средний диаметр), шаг витков, число ее рабочих витков iп = Lп / t (где Lп – длина рабочей части пружины).

Изменяя индекс и сохраняя жесткость пружины, можно изменять ее длину и диаметр. Силовые факторы. в поперечных сечениях клапанной пружины сводятся к моменту М = Рп\*dср/2 и силе, действующей вдоль оси пружины. Момент М раскладывают на крутящий и изгибающий моменты:



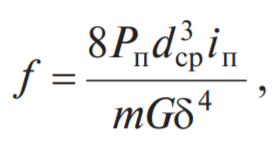
Обычно угол подъема витков α = 10 ... 12° и расчет пружины можно вести только на кручение от момента Mk‘= Рп\*dср/2.

Наибольшее напряжение кручения возникает на внутренних во окнах:



где χ = 1 + 1,45/с - коэффициент, учитывающий кривизну витков. При расчете пружин искомым является диаметр 8 проволоки . Задаваясь в соответствии с выбранной маркой стали допускаемыми напряжениями [τ], определяют δ. Полученное значение δ округляют в большую сторону до ближайшего стандартного диаметра проволоки и повторяют расчет пружины.

Осевое упругое сжатие пружины



где m - коэффициент, который учитывает влияние перерезывающих сил, зависящих от с и который с достаточной точностью может быть принят равным 0,95 ... 1,00; G - модуль сдвига.

Обычно f = (1,5-3,5)h.

По результатам расчета строят характеристику пружины (рис. 3) и устанавливают значение усилия пружины по при закрытом клапане:

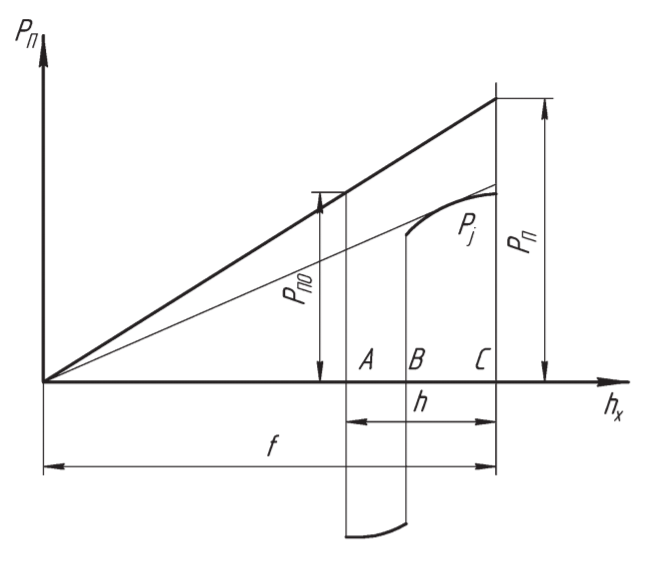
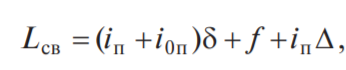


Рис. 3. Изменение силы пружины (J) и сил инерции (2) по подъему клапана

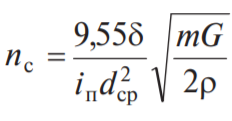
Сила пружины выпускного клапана Рп должна быть достаточной для удержания клапана в закрытом состоянии во время такта впуска с учетом разрежения Δр в цилиндре.

Длина пружины в свободном состоянии



где п = 2-3 - число опорных витков; Δ = 0,2 ... 0,3 мм - величина зазора между витками пружины при открытом клапане, зависящая от размеров пружины.

После определения размеров пружины ее проверяют на резонанс. При резонансе перемещение среднего витка пружины резко возрастает. Обычно определяют число пс собственных колебаний пружины в минуту и отношение пс к частоте вращения пр распределительного вала. Число собственных колебаний пружины



где G - модуль упругости II рода материала пружины; ρ - плотность материала пружины.

Опасность резонанса считается незначительной, если отношение nc / np > 8; значение nc не должно быть кратным np.

**Формулировка задачи оптимизации**

В качестве исходных данных программа будет принимать

- P­p – сила, необходимая для удержания клапана в открытом состоянии

- h – ход кулачка

- f – максимальное сжатие пружины

Кроме того, нужно определить константы, описывающие материал из которого будет изготовлена пружина.

Выше перечисленные данные записываются в файл config.py.

В файле data.py хранится массив из возможных геометрических параметров клапанных пружин.

Целевыми функциями будут являться коэффициент запаса, а также длина пружины в сжатом состоянии.

Искомыми параметры в этой задаче будут:

* + - средний диаметр пружины dcp
    - диаметр проволоки δ

**Построение фронта Парето для клапанных пружин**

Как было сказано в предыдущей главе, целевыми функциями будут являться коэффициент запаса, а также длина пружины в сжатом состоянии.

Определим термин доминирования по Парето одной пружины над другой.

Критерии доминирования:

1. Больший коэффициент запаса.
2. Меньшая длина в сжатом состоянии.

Тогда пружина А доминирует по Парето Б, если А не хуже Б по данным двум критерия и хотя бы по одному из них превосходит Б.

Следовательно, в данной задаче нужно минимизировать функцию Lсжат и максимизировать nτ.

Так как критериев доминирования два, то фронт Парето будет выглядеть как ломанная линия в двухмерном пространстве решений (см. Рис. 4)

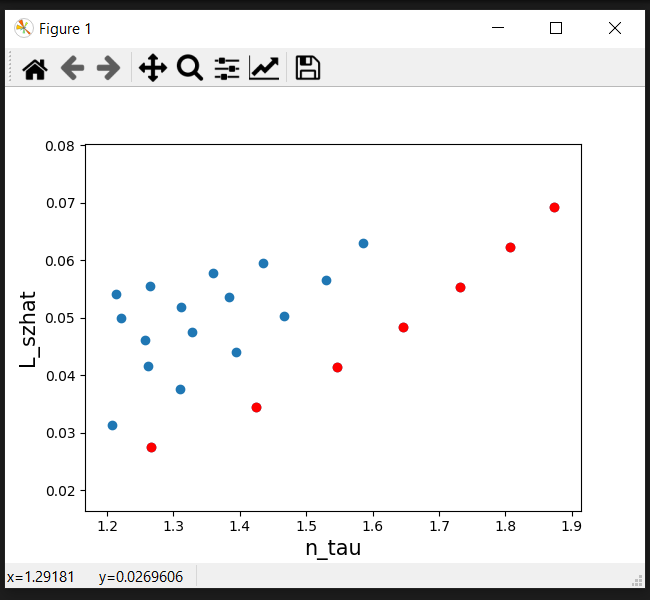


Рис 4: Пример исполнения программы (красным обозначен фронт)

**Заключение**

Я достиг всех целей учебной практики, а именно:

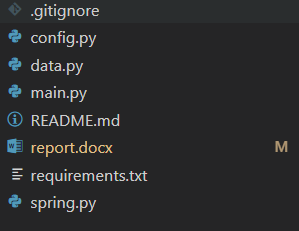
* изучил газораспределительный механизм современных ДВС;
* произвел расчет клапанных пружин;
* изучить основы программирования в Python;
* построил фронт Парето для клапанных пружин.

**Список используемых источников**

1. Д-р техн. наук Н. Д. Чайнов, д-р техн. наук Н. А. Иващенко, канд. техн. наук. А. Н. Краснокутский, канд. техн. наук Л. Л. Мягков: Конструирование двигателей внутреннего сгорания. — 2008 — с. 504.
2. С. Люк: Основы метаэвристики — 2009 — с. 204.
3. http://www.mopartsracing.com/parts/Comp/CompCamsSprings.pdf

**Приложения**

1. Структура проекта:



1. main.py
2. # -\*- coding: utf-8 -\*-
3. import math
4. import random
5. import matplotlib.pyplot as plt
6. from config import Config
7. from spring import Spring
8. from data import array\_of\_delta\_sp, array\_of\_i\_p\_sp
9. def print\_smt(value, description="DATA"):
10. print()
11. print("------------------------" + description + "------------------------")
12. print(value)
13. print("------------------------" + description + "------------------------")
14. print()
15. def show\_plot(all\_springs, front):
16. plt.xlabel("n\_tau", fontsize=15)
17. plt.ylabel("L\_szhat", fontsize=15)
18. all\_springs\_x = [i.n\_tau for i in all\_springs]
19. all\_springs\_y = [j.L\_szhat for j in all\_springs]
20. plt.scatter(all\_springs\_x, all\_springs\_y)
21. front\_springs\_x = [i.n\_tau for i in front]
22. front\_springs\_y = [j.L\_szhat for j in front]
23. plt.scatter(front\_springs\_x, front\_springs\_y, c="r")
24. plt.show()
25. if \_\_name\_\_ == "\_\_main\_\_":
26. # Main program starts here
27. pop\_size = 20
28. # max\_gen = 921
29. max\_gen = 921
30. # Initialization
31. min\_x = -55
32. max\_x = 55
33. solution = [min\_x + (max\_x - min\_x) \* random.random() for i in range(0, pop\_size)]
34. gen\_no = 0
35. config = Config()
36. springs = []
37. for delta\_sp in array\_of\_delta\_sp:
38. for i\_p\_sp in array\_of\_i\_p\_sp:
39. spring = Spring(config, delta\_sp, i\_p\_sp)
40. if spring.is\_spring\_ok():
41. springs.append(spring)
42. front = []
43. if len(springs) == 0:
44. print("SPRING DO NOT FIT LIMITS")
45. exit(0)
46. # generate Pareto-front
47. for spring in springs:
48. not\_dominating = True
49. for front\_elem in front:
50. if spring.dominates\_by\_pareto(front\_elem):
51. print(front.index(front\_elem))
52. print(len(front))
53. front = [x for x in front if not x.equals(front\_elem)]
54. print(len(front))
55. if front\_elem.dominates\_by\_pareto(spring):
56. not\_dominating = False
57. if not\_dominating and not spring.on\_array(front):
58. front.append(spring)
59. for spring in front:
60. print(spring)
61. print("ALL SPRINGS ARR LEN: ", len(springs))
62. print("FRONT SPRINGS ARR LEN: ", len(front))
63. show\_plot(springs, front)
64. print(front[0])
65. print(springs[0])
66. spring.py
67. import math
68. import random
69. class Spring:
70. def \_\_init\_\_(self, config, delta\_sp, i\_p\_sp):
71. self.config = config
72. self.h\_sp = config.h\_sp
73. self.k\_sp = config.k\_sp
74. self.G\_sp = config.G\_sp
75. self.n\_p = config.n\_p
76. self.m\_sp = config.m\_sp
77. self.ro\_sp = config.ro\_sp
78. self.gap\_sp = config.gap\_sp
79. self.i\_0p\_sp = config.i\_0p\_sp
80. self.P\_p = config.P\_p
81. self.f\_sp = self.P\_p / self.k\_sp
82. self.P\_p0 = self.P\_p \* (self.f\_sp - self.h\_sp) / self.f\_sp
83. self.delta\_sp = delta\_sp
84. self.i\_p\_sp = i\_p\_sp
85. self.d\_sr\_sp = self.\_get\_d\_sr\_sp()
86. # c - индекс пружины
87. self.c\_sp = self.d\_sr\_sp / self.delta\_sp
88. #
89. self.n\_c\_sp = self.\_get\_n\_c\_sp()
90. # высота пружины в свободном состоянии
91. self.L\_sv\_sp = self.\_get\_L\_sv\_sp()
92. # Высота пружины в сжатом состоянии
93. self.L\_szhat = self.L\_sv\_sp - self.f\_sp
94. # Наибольшее напряжение кручения
95. self.t\_max = self.\_get\_t\_max()
96. # Коэффициент запаса
97. self.n\_tau = self.\_get\_n\_tau()
98. # Возвращает число рабочих витков пружины (8.55)
99. def \_get\_d\_sr\_sp(self):
100. return (
101. self.f\_sp
102. \* self.m\_sp
103. \* self.G\_sp
104. \* self.delta\_sp \*\* 4
105. / (8 \* self.P\_p \* self.i\_p\_sp)
106. ) \*\* (1 / 3)
107. # Возвращает число собственных колебаний пружины (8.59)
108. def \_get\_n\_c\_sp(self):
109. return (
110. 9.55
111. \* self.delta\_sp
112. / (self.i\_p\_sp \* self.d\_sr\_sp \*\* 2)
113. \* (self.m\_sp \* self.G\_sp / (2 \* self.ro\_sp)) \*\* (1 / 2)
114. )
115. # Высота (длина) пружины в свободном состоянии (стр 333)
116. def \_get\_L\_sv\_sp(self):
117. return (
118. (self.i\_p\_sp + self.i\_0p\_sp) \* self.delta\_sp
119. + self.f\_sp
120. + self.i\_p\_sp \* self.gap\_sp
121. )
122. # t\_max - Наибольшее напряжение кручения
123. def \_get\_t\_max(self):
124. hi = 1 + 1.45 / self.c\_sp
125. return 8 \* hi \* self.d\_sr\_sp \* self.P\_p / (math.pi \* self.delta\_sp \*\* 3)
126. # n\_tau - коэффициент запаса
127. def \_get\_n\_tau(self):
128. k\_tau\_sp = self.config.k\_tau\_sp
129. e\_p\_tau\_sp = self.config.e\_p\_tau\_sp
130. \_lambda = 0.02e-3
131. e\_m\_tau\_sp = 0.5 + (1 - 0.5) \* math.exp(-\_lambda \* self.delta\_sp)
132. return self.config.t\_1\_sp / (k\_tau\_sp / e\_m\_tau\_sp / e\_p\_tau\_sp \* self.t\_max)
133. def is\_spring\_ok(self):
134. if self.\_is\_resonance\_possible() or self.n\_tau < 1.2:
135. return False
136. return True
137. # Возвращает true, если возможен резонанс (8.59)
138. def \_is\_resonance\_possible(self):
139. if (
140. self.n\_c\_sp / self.n\_p == int(self.n\_c\_sp / self.n\_p)
141. or self.n\_c\_sp / self.n\_p < 8
142. ):
143. return True
144. return False
145. def dominates\_by\_pareto(self, other):
146. if self.L\_szhat < other.L\_szhat and self.n\_tau > other.n\_tau:
147. return True
148. return False
149. def equals(self, other):
150. if (
151. self.d\_sr\_sp == other.d\_sr\_sp
152. and self.delta\_sp == other.delta\_sp
153. and self.i\_p\_sp == other.i\_p\_sp
154. ):
155. return True
156. return False
157. def on\_array(self, springs):
158. for s in springs:
159. if self.equals(s):
160. return True
161. return False
162. def \_\_str\_\_(self):
163. return (
164. "Spring:\n\tP\_p0 = {P\_p0} H\n\tP\_p = {P\_p} H\n\tk\_sp = {k\_sp} H/m\n\t"
165. "G\_sp = {G\_sp} Pa\n\tn\_p = {n\_p} rpm\n\td\_sr = {d\_sr\_sp} mm\n\ti\_p = {i\_p\_sp}"
166. "\n\tdelta = {delta\_sp} mm\n\tf\_sp = {f\_sp} mm\n\th\_sp = {h\_sp} mm\n\tro = {ro\_sp} kg/m^3"
167. "\n\tn\_c\_sp = {n\_c\_sp} rpm\n\tL\_szhat = {L\_szhat}\n\tn\_tau = {n\_tau} mm".format(
168. P\_p0=self.P\_p0,
169. P\_p=self.P\_p,
170. k\_sp=self.k\_sp,
171. G\_sp=self.G\_sp,
172. n\_p=self.n\_p \* 60,
173. d\_sr\_sp=self.d\_sr\_sp \* 1e3,
174. i\_p\_sp=self.i\_p\_sp,
175. delta\_sp=self.delta\_sp \* 1e3,
176. f\_sp=self.f\_sp \* 1e3,
177. h\_sp=self.h\_sp \* 1e3,
178. ro\_sp=self.ro\_sp,
179. n\_c\_sp=self.n\_c\_sp \* 60,
180. L\_szhat=self.L\_szhat \* 1e3,
181. n\_tau=self.n\_tau,
182. )
183. )
184. config.py
185. class Config:
186. def \_\_init\_\_(self):
187. # размер в СИ
188. # P\_p - значение усилия пружины при открытом клапане
189. self.P\_p = 400
190. # --------------------SPRING--------------------
191. # Максимальный подъем клапана по кулачку
192. self.h\_sp = 10E-3
193. # k\_sp - жесткость пружины
194. self.k\_sp = 30E3
195. # G\_sp - модуль сдвига
196. self.G\_sp = 80E9
197. # m\_sp - коэффициент, учитывающий влияние перерезывающих сил,зависящий от с,
198. # который с достаточной точностью может быть принят равным 0,95–1,00
199. self.m\_sp = 0.95
200. # n\_p - частота вращения коленвала
201. self.n\_p = 7300 /2 / 60
202. # ro\_sp - плотность материала пружины
203. self.ro\_sp = 7860
204. # gap\_sp - зазор между витками пружины при открытом клапане
205. self.gap\_sp = 0.2E-3
206. # i\_0p - число опорных витков
207. self.i\_0p\_sp = 2
208. # t\_0\_sp - Допускаемое статическое напряжение (тау), Па
209. self.t\_0\_sp = 600E6
210. # t\_0\_sp - Допускаемое динамическое напряжение (тау минус один), ПА
211. self.t\_1\_sp = 400E6
212. # эффективный коэффициент концентрации напряжений
213. self.k\_tau\_sp = 1.2
214. # Это оценивается коэффициентом влияния состояния поверхностного слоя
215. self.e\_p\_tau\_sp = 0.99
216. # --------------------END\_SPRING--------------------
217. data.py
218. array\_of\_delta\_sp\_inch = [
219. 0.363 / 2,
220. 0.256 / 2,
221. 0.355 / 2,
222. 0.344 / 2,
223. 0.284 / 2,
224. 0.414 / 2,
225. 0.332 / 2,
226. 0.360 / 2,
227. 0.382 / 2,
228. 0.312 / 2,
229. 0.532 / 2,
230. 0.442 / 2,
231. 0.483 / 2,
232. 0.424 / 2,
233. 0.456 / 2,
234. ]
235. array\_of\_delta\_sp = []
236. for delta\_sp in array\_of\_delta\_sp\_inch:
237. array\_of\_delta\_sp.append(delta\_sp \* 25.4 \* 10\*\*(-3))
238. # число опорных витков
239. array\_of\_i\_p\_sp = [2, 3, 4, 5, 6, 7, 8]