Министерство образования и науки Российской Федерации   
Федеральное государственное образовательное учреждение высшего образования  
«Национальный исследовательский университет «МЭИ»   
Институт ИВТ  
Кафедра ПМИИ

Курсовая работа

Численные методы

Тема: “Изучение фазовых траекторий систем дифференциальных уравнений в зависимости от начальных условий”

Выполнил

Студент группы А-05-19

Желтиков Александр

Преподаватель Амосова О.А.

Москва 2021

Оглавление

[Введение. 3](#_Toc91780164)

[Постановка задачи 4](#_Toc91780165)

[Вывод расчетной формулы метода. Явного метода Адамса 4-го порядка точности. 4](#_Toc91780166)

[Обзор вспомогательного метода Рунге – Кутты 4го порядка. 4](#_Toc91780167)

[Описание работы функций программы 5](#_Toc91780168)

[Подготовка и проверка тестовых примеров. 5](#_Toc91780169)

[Изучение графиков зависимостей решения x(t), y(t), z(t), а также построение фазовых портретов. 6](#_Toc91780170)

[Изучение двух решений со слегка различающимися данными на одном графике. 10](#_Toc91780171)

[Код программы. 10](#_Toc91780172)

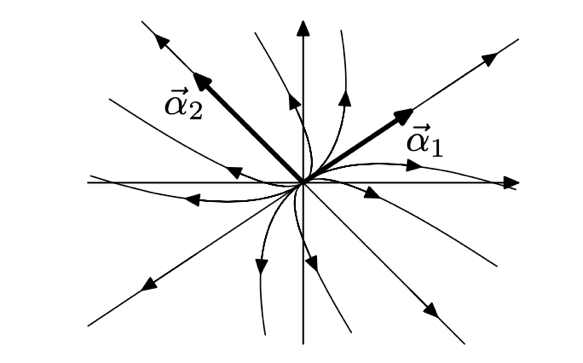
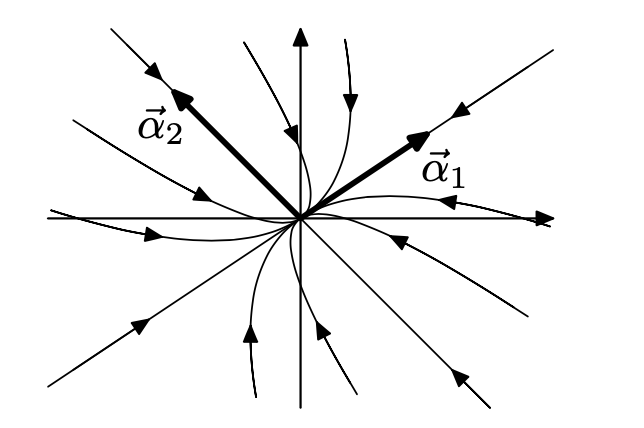
[Используемая литература. 13](#_Toc91780173)

# Введение.

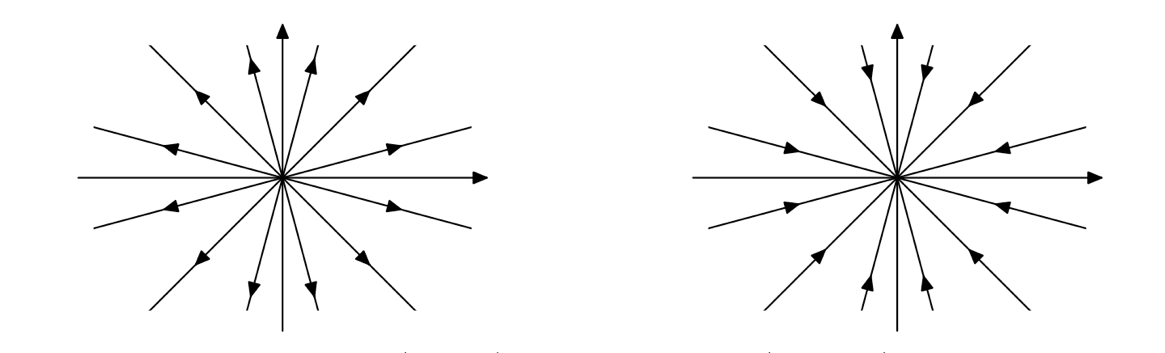
*Дифференциальное уравнение — уравнение, в которое входят производные функции и могут входить сама функция, независимая переменная и параметры. Порядок входящих в уравнение производных может быть различен. Производные, функции, независимые переменные и параметры могут входить в уравнение в различных комбинациях или могут отсутствовать вовсе, кроме хотя бы одной производной.*

*Классификация точек покоя по фазовым траекториям.*

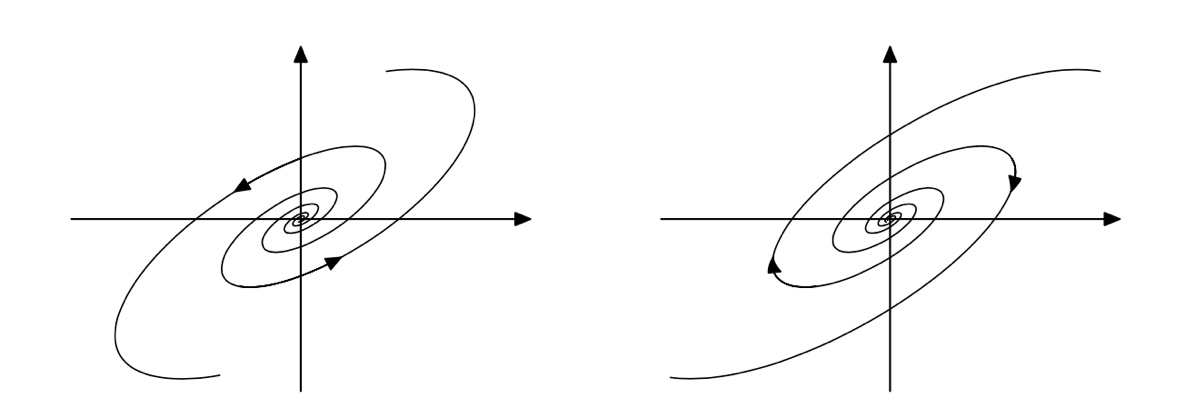
*На графиках изображено, что образует множество точек покоя:*

**

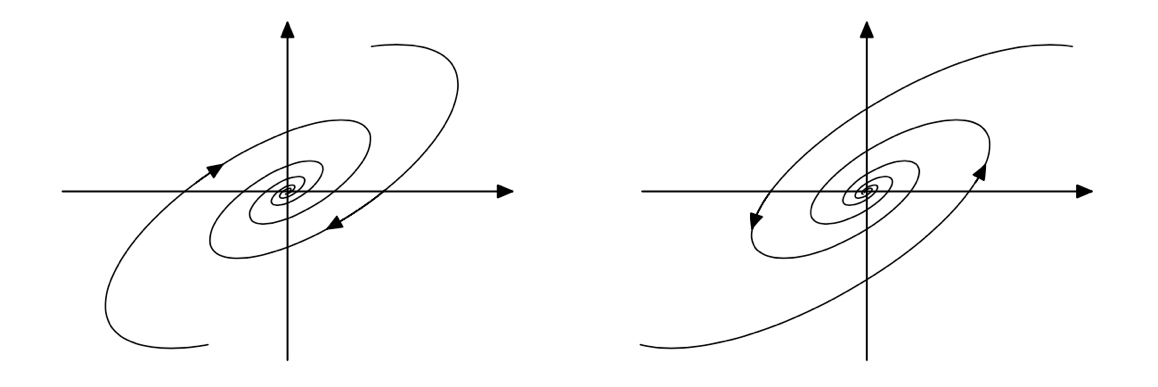
*Устойчивый узел Неустойчивый узел*

**

*Неустойчивый дикритический узел Устойчивый дикритический узел*

**

*Неустойчивые фокусы*

**

*Устойчивые фокусы*

# Постановка задачи

Для изучения фазовых траекторий нам дана система линейных дифференциальных уравнений первого порядка, с переменных коэффициентом , так же у данной системы начальные значения функций равны: x(0) = y(0) = z(0) = 0

Для того что бы изучить поведение фазовых траекторий нам необходимо найти решение данной ОДУ, чтобы это сделать, выведем и используем метод Адамса 4-го порядка точности.

# Вывод расчетной формулы метода. Явного метода Адамса 4-го порядка точности.

Для вывода формулы воспользуемся равенством:

*Заменим приближенно функцию интерполяционным многочленом степени принимающим значения в тех узлах где значения сеточной функции  уже найдены. Интегрирование этого многочлена дает приближенное равенство:*

*В результате от (1) приходим к формуле, соответствующей явному k-шаговому методу Адамса-Башфорта.*

*Далее выведем формула Адамса 4-го порядка точности:*

*Запишем функцию*

*Проинтегрируем данную функцию:*

*После интегрирования подставим результат в исходное равенство:*

*Для того чтобы начать вычисления по данному методу, необходимо знать решение в четырех точках. Это можно сделать, например, с помощью методов Рунге - Кутты. Кроме того, коэффициент при погрешности, например, для метода четвертого порядка точности Рунге - Кутты существенно меньше соответствующего коэффициента для метода Адамса.*

# Обзор вспомогательного метода Рунге – Кутты 4го порядка.

*Методы Рунге — Кутты — большой класс численных методов решения задачи Коши для обыкновенных дифференциальных уравнений и их систем. Первые методы данного класса были предложены около 1900 года немецкими математиками К. Рунге и М. В. Куттой.*

*Наиболее известный и часто используемый из методов Рунге-Кутты – это классический 4-этапный метод четвертого порядка точности.*

*Данный метод достаточно прост и эффективен на малых отрезках, а также когда нужна сравнительно не высокая точность.*

*Формула данного метода:*

# Описание работы функций программы

*Для того, чтобы производить меньше механических расчетов, все методы и функции постройки графиков реализованы на языке Python.*

*Сначала мы находим начальные значения для метода Адамса с помощью метода Рунге-Кутты [(1)](#Runge),*

*так как у нас дифференциальное уравнение зависит от нескольких переменных, не забываем учесть это. После этого, мы вызываем данную функцию [(1)](#Runge), при использовании функции, которая находит решение СЛДУ с помощью метода Адамса [(2)](#Adams). Все полученные значения записываем в переменные x\_data, y\_data, z\_data.*

*За построение графиков отвечают функции:*

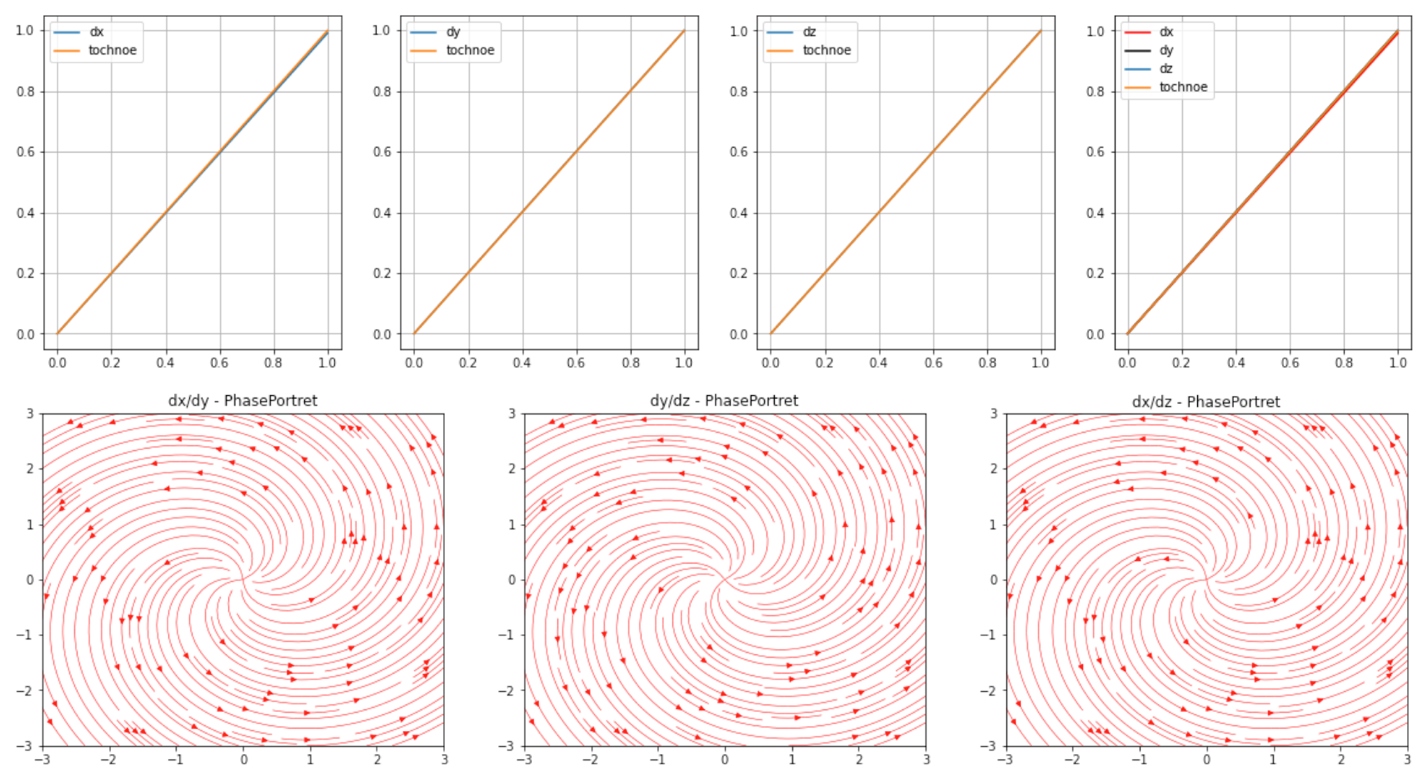
* *[Построение фазовых портретов](#plotPhasePortret)*
* *[Построение графиков решений](#PlotGraph)*
* *[Построение двух решений на одном графике](#plotSimilaGraph)*

# Подготовка и проверка тестовых примеров.

Пусть x(t) = y(t)= z(t) = t

Тогда наша СЛДУ принимает вид:

Для проверки нашего метода построим все на одном графике

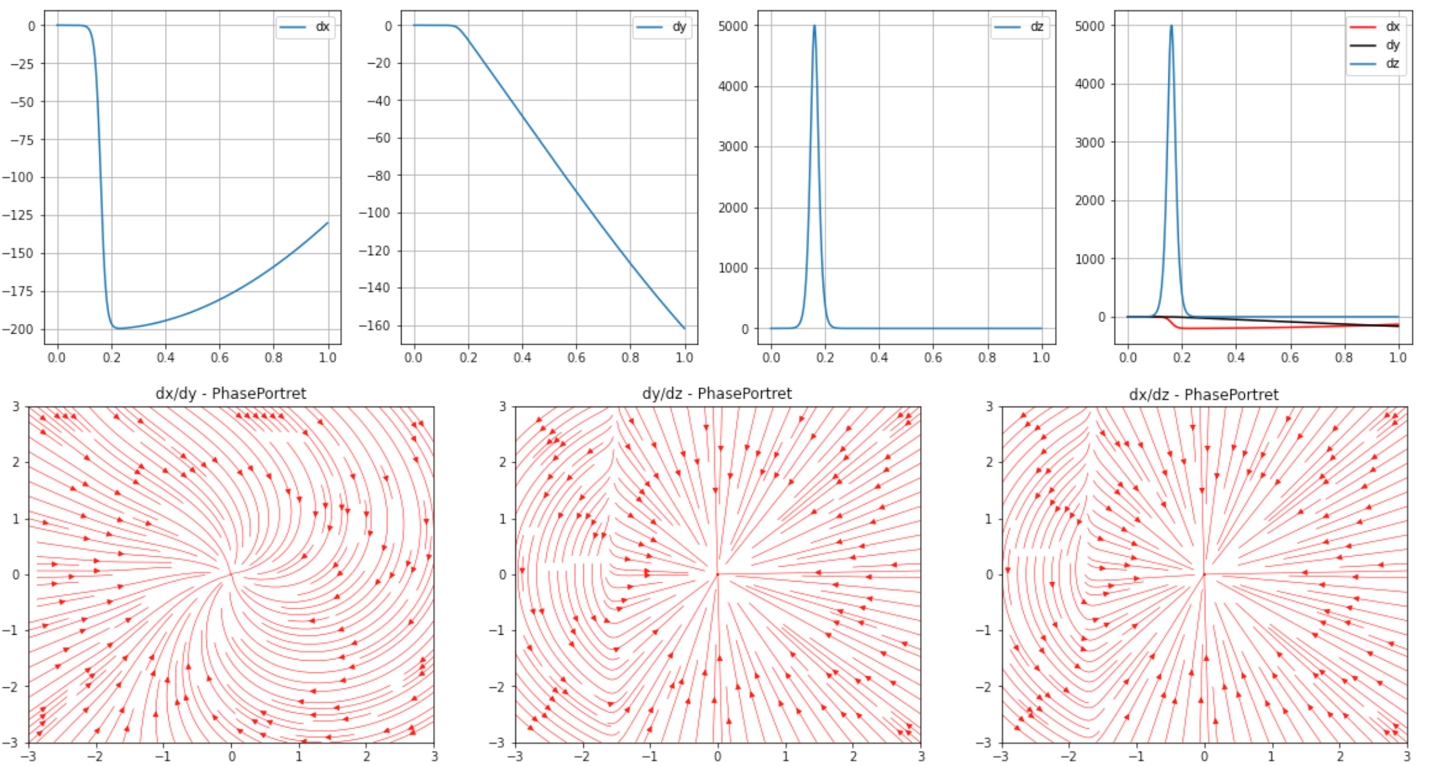


Как мы можем заметить, графики совпадают, следовательно запрограммированный метод работает.

# Изучение графиков зависимостей решения x(t), y(t), z(t), а также построение фазовых портретов.

Для изучения поведения фазовых траекторий, будет брать значения из массива [-100, -10, -1, 0, 1, 10, 100]

Рассмотрим значение



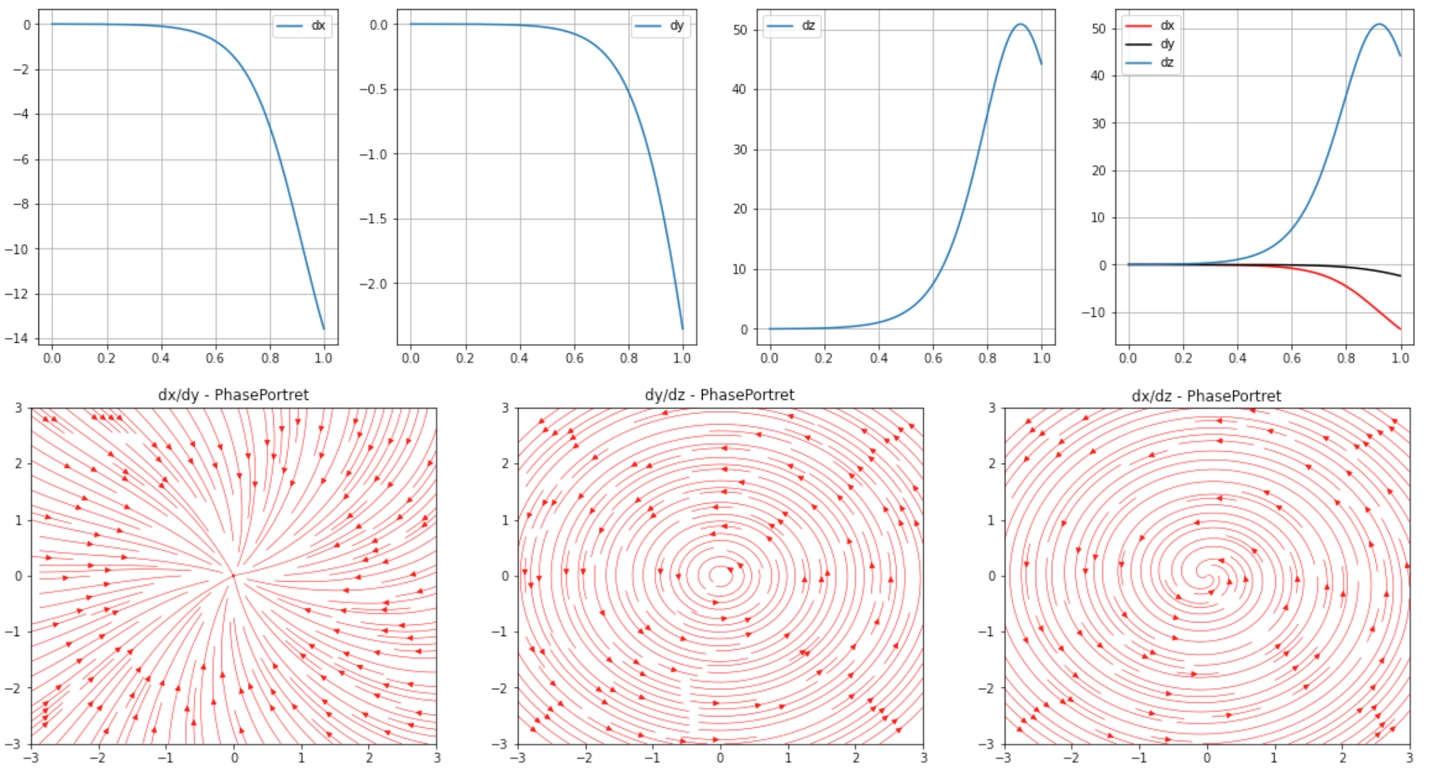
При данном значении коэффициента , можно заметить, что все фазовые траектории устойчивы, также при детальном рассмотрении, видим:

dx/dy – точки покоя образуют устойчивый фокус.

dy/dz – точки покоя образуют подобие устойчивого дикритического узла.

dx/dz – аналогичная ситуация как и на портрете dy/dz.

Рассмотрим значение



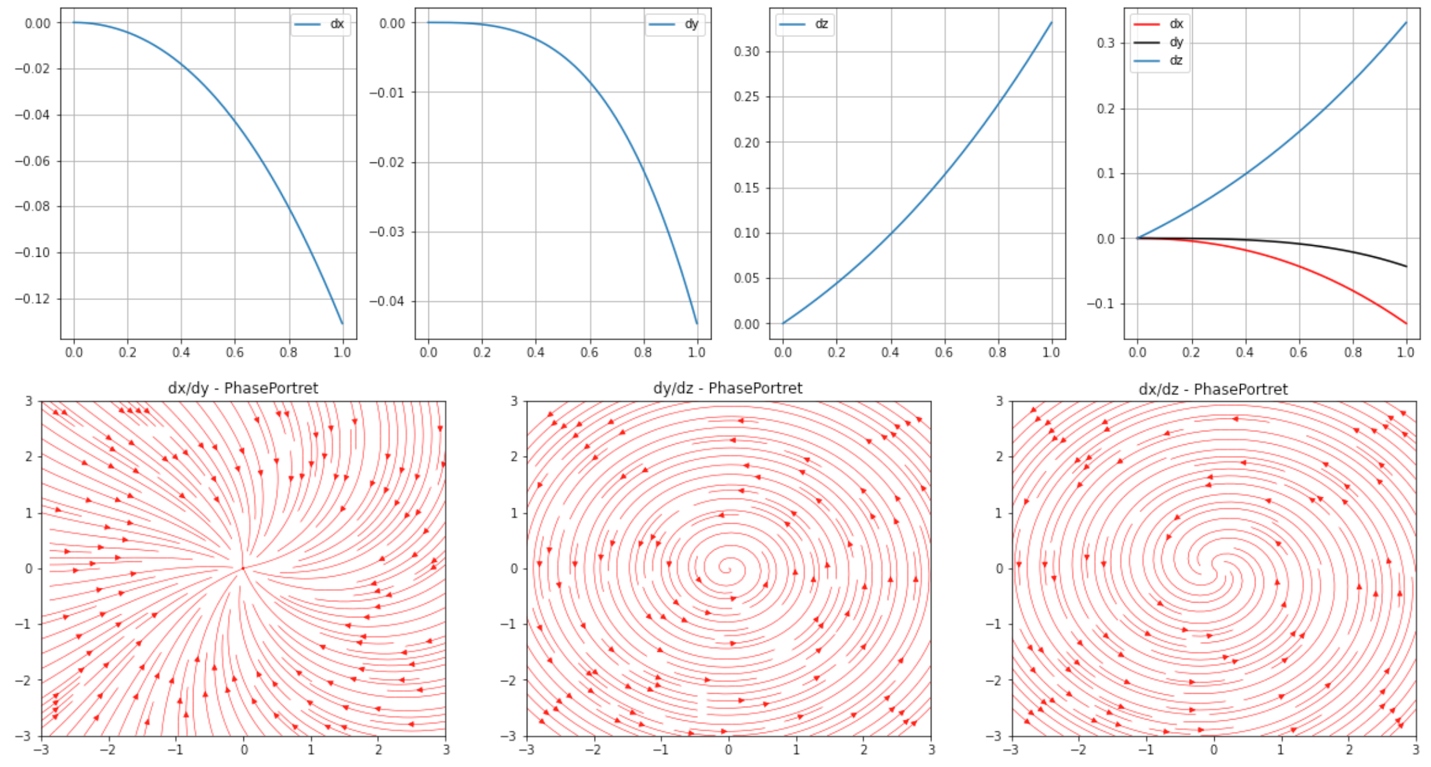
Далее мы уменьшили значение коэффициента в десять раз, и сразу можно увидеть, что фазовые траектории сильно изменились, но все остались устойчивыми.  
  
dx/dy – точки покоя образуют устойчивый фокус.

dy/dz – здесь тип точек покоя изменился, теперь это устойчивый центр.

dx/dz – здесь также тип точек покоя изменился, теперь это устойчивый фокус

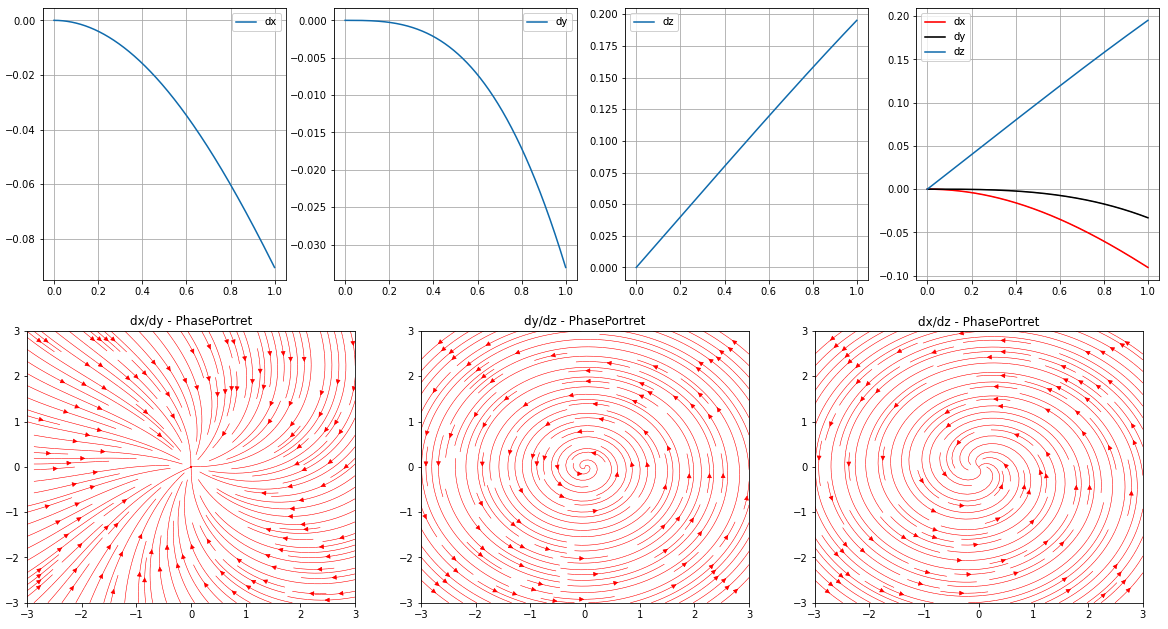
Уменьши значение коэффициента еще в 10 раз.

Рассмотрим значение



Сравнивая данные портреты с предыдущими, можно заметить, что они особо и не различаются. Есть различия только в графиках решений, так же портрет dy/dz немного начинает перетекать из устойчивого центра в устойчивый фокус.

Теперь рассмотрим значение



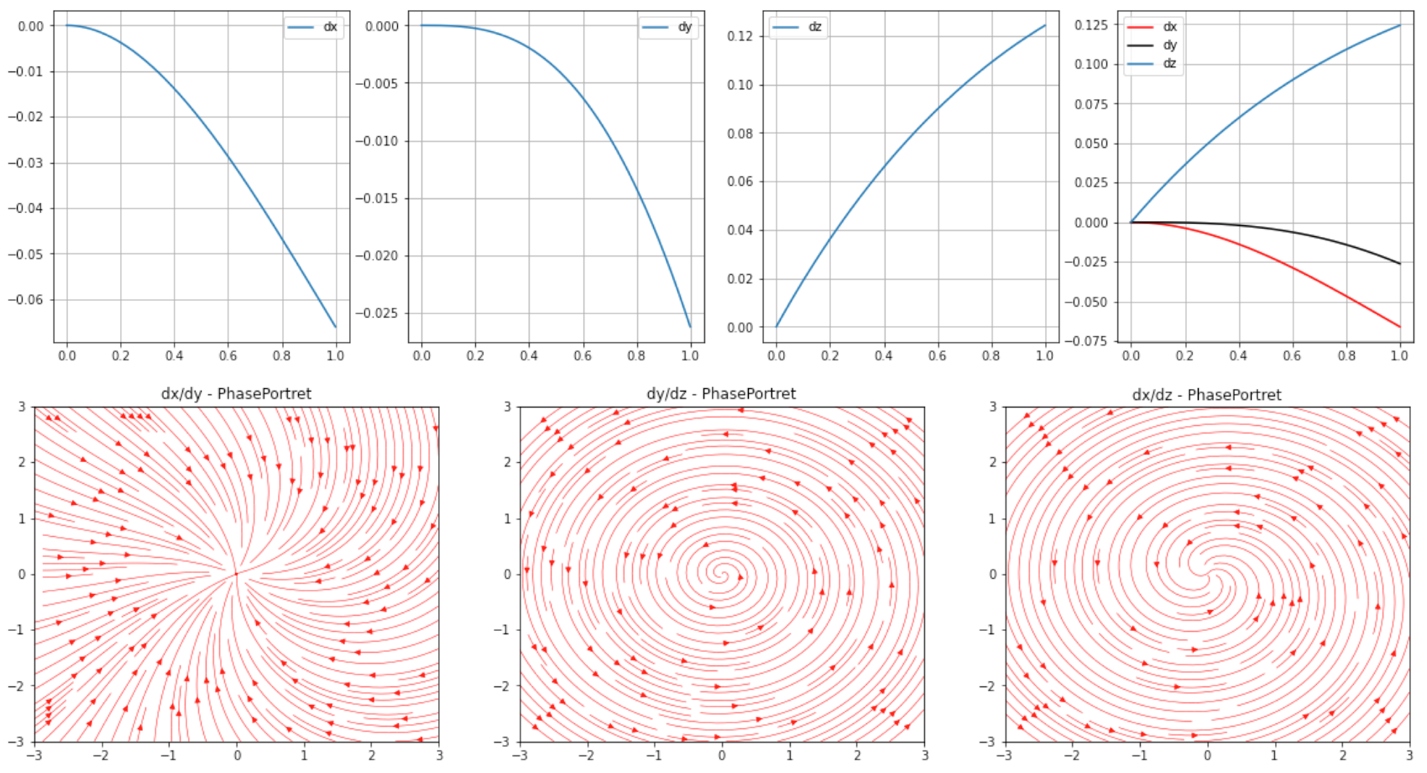
Рассмотрим, что происходит на фазовых портретах:

dx/dy фазовые траектории, как и раньше, образуют устойчивый фокус.

dy/dz здесь мы можем заметить, что теперь, точки покоя ОДУ образуют не устойчивый центр, а устойчивый фокус

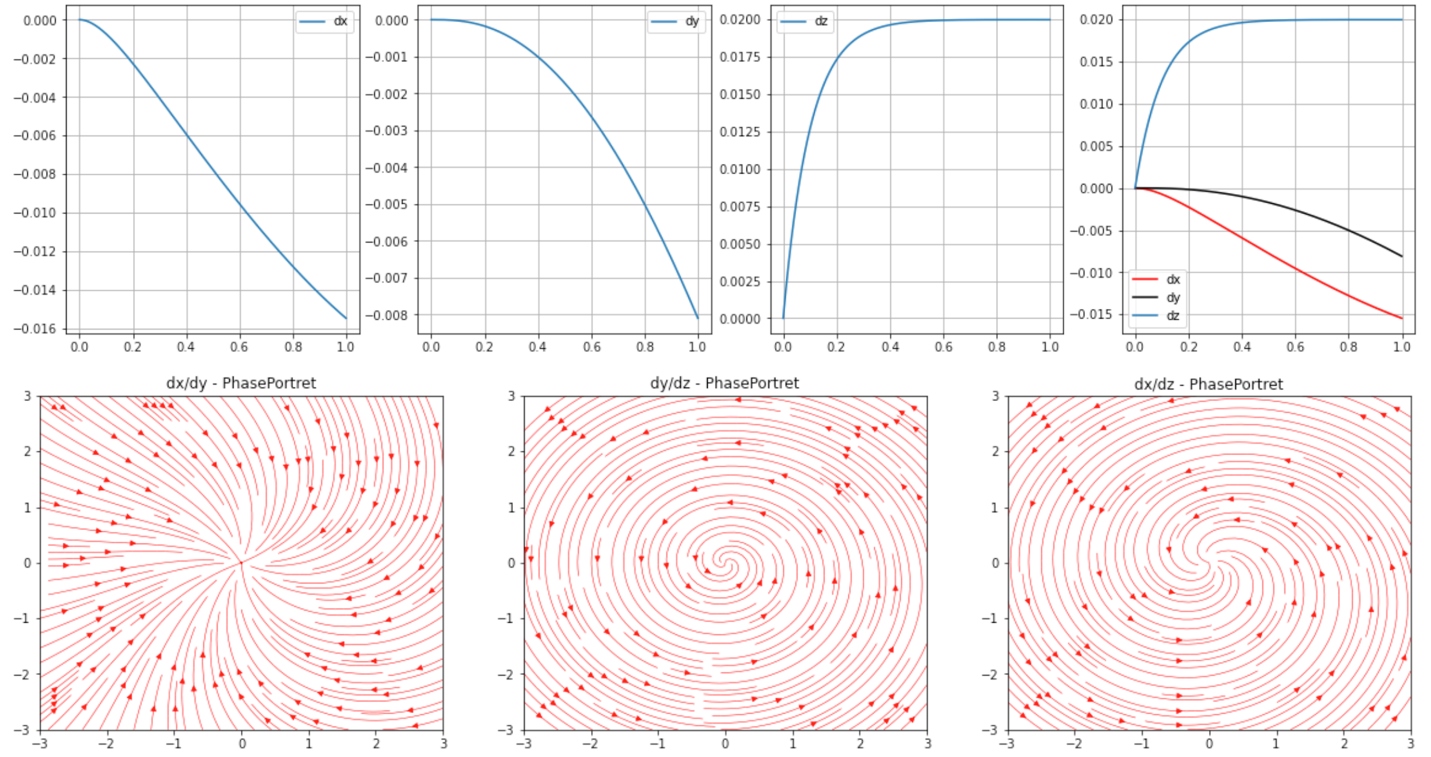
dx/dz ситуация не особо изменилась, при таком значении сохранился устойчивый фокус.

Далее рассмотрим



При увеличении значения коэффициента на 1, каких-либо критических изменений не произошло.

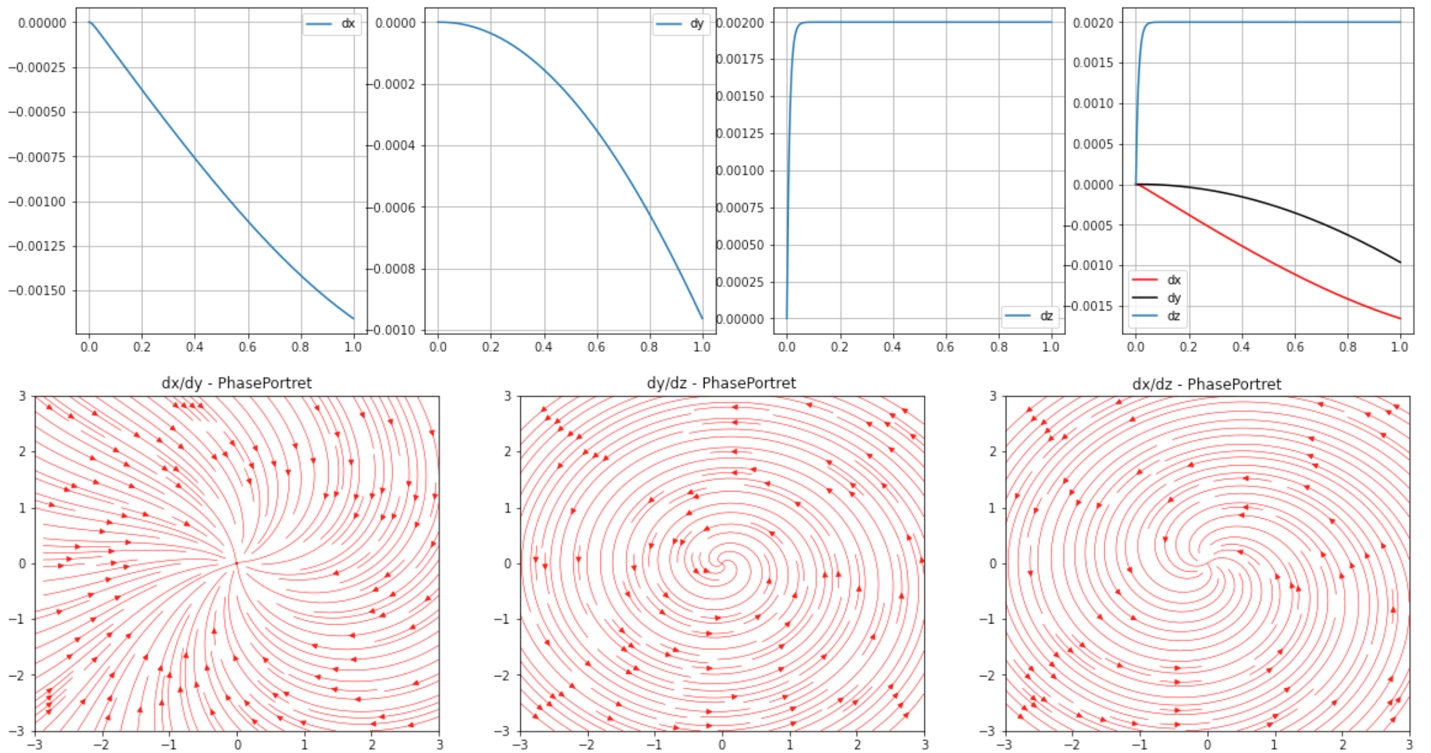
Увеличим значение в десять раз и рассмотрим значение



Ситуация практически аналогична предыдущему значению, за исключением одного момента, на портрете dx/dz фазовые траектории стали иметь чуть большую кривизну, тем самым более наглядно можно понять, что тип точек покоя – это устойчивый фокус.

Пробуем добиться более наглядных результатов, увеличив значение еще в десять раз.

Рассмотрим значение

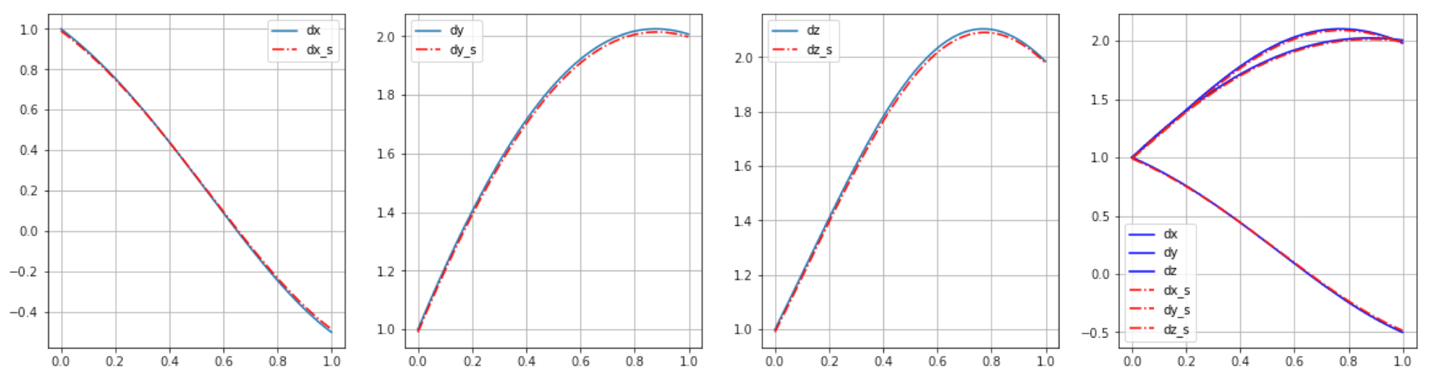


Несмотря на то, что мы увеличили значение коэффициента в 10 раз, ситуация с траекториями не сильно поменялась.

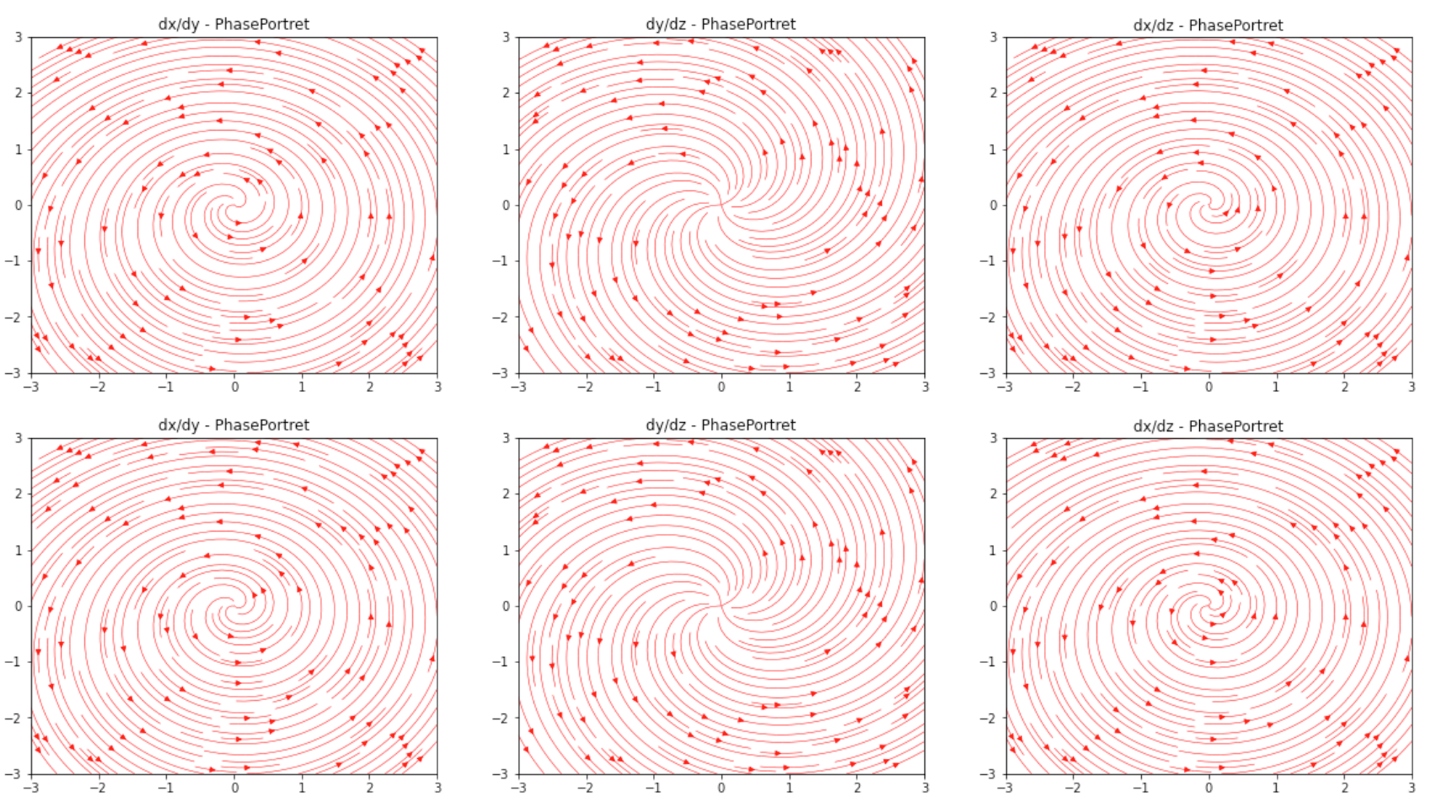
При исследовании траекторий данного ОДУ на диапазоне коэффициента [-100 ; 100], можно сделать вывод о том, что при увеличении значений фазовые траектории не будут сильно изменяться, а тип точки покоя сохранится.

# Изучение двух решений со слегка различающимися данными на одном графике.

Рассмотрим начальные значения x(0) = y(0) = z(0) = 1; , и слегка различающиеся значения з x(0) = y(0) = z(0) = 0.99 ;



Как можно заметить, графики функций немного различаются (Но это логично), рассмотрим поведение фазовых портретов:



Сверху наше исходное, снизу со слегка различающимися значениями. Их точки покоя — это фокусы, причем в обоих случаях неустойчивые.

# Код программы.

1. **import** numpy **as** np
2. **from** matplotlib **import** pyplot **as** plt
4. **def** PlotGraph(T , t0 , h , dx , dy , dz, C1 , C2 , C3 , m):
5. x\_data , y\_data , z\_data = Adams(T , t0 , h , dx , dy , dz, C1 , C2 , C3 , m)
6. t\_data =[0.001\*i **for** i **in** range(round(1 / h))]
7. fig, axs = plt.subplots(1, 4, figsize = (20, 5))
8. axs[0].plot(t\_data, x\_data, label = 'dx')
9. axs[1].plot(t\_data, y\_data, label = 'dy')
10. axs[2].plot(t\_data, z\_data, label = 'dz')
11. axs[3].plot(t\_data, x\_data, label = 'dx' , color = 'red')
12. axs[3].plot(t\_data, y\_data, label = 'dy' , color = 'black')
13. axs[3].plot(t\_data, z\_data, label = 'dz')
14. axs[0].legend()
15. axs[0].grid()
16. axs[1].legend()
17. axs[1].grid()
18. axs[2].legend()
19. axs[2].grid()
20. axs[3].legend()
21. axs[3].grid()
23. **def** plotSimilaGraph(T , t0 , h , dx , dy , dz, C1 , C2 , C3 , m , C1\_s , C2\_s , C3\_s , m\_s):
24. x\_data , y\_data , z\_data = Adams(T , t0 , h , dx , dy , dz, C1 , C2 , C3 , m)
25. x\_data\_s , y\_data\_s , z\_data\_s = Adams(T , t0 , h , dx , dy , dz, C1\_s , C2\_s , C3\_s , m\_s)
26. t\_data =[0.001\*i **for** i **in** range(round(1 / h))]
27. fig, axs = plt.subplots(1, 4, figsize = (20, 5))
28. axs[0].plot(t\_data, x\_data, label = 'dx')
29. axs[0].plot(t\_data, x\_data\_s, label = 'dx\_s' , color = 'red', linestyle="-.")
31. axs[1].plot(t\_data, y\_data, label = 'dy')
32. axs[1].plot(t\_data, y\_data\_s, label = 'dy\_s' , color = 'red', linestyle="-.")
34. axs[2].plot(t\_data, z\_data, label = 'dz')
35. axs[2].plot(t\_data, z\_data\_s, label = 'dz\_s' , color = 'red', linestyle="-.")
37. axs[3].plot(t\_data, x\_data, label = 'dx' , color = 'blue')
38. axs[3].plot(t\_data, y\_data, label = 'dy' , color = 'blue')
39. axs[3].plot(t\_data, z\_data, label = 'dz' , color = 'blue')
40. axs[3].plot(t\_data, x\_data\_s, label = 'dx\_s' , color = 'red', linestyle="-.")
41. axs[3].plot(t\_data, y\_data\_s, label = 'dy\_s' , color = 'red', linestyle="-.")
42. axs[3].plot(t\_data, z\_data\_s, label = 'dz\_s' , color = 'red', linestyle="-.")
44. axs[0].legend()
45. axs[0].grid()
46. axs[1].legend()
47. axs[1].grid()
48. axs[2].legend()
49. axs[2].grid()
50. axs[3].legend()
51. axs[3].grid()

54. **def** plotPhasePortret(T , t0 , h , dx , dy , dz, C1 , C2 , C3 , m):
55. fig, axs = plt.subplots(1, 3, figsize = (20, 5))
56. X, Y = np.meshgrid(np.linspace(-3.0, 3.0, 1000), np.linspace(-3.0, 3.0, 1000))
57. R, Theta = (X\*\*2 + Y\*\*2)\*\*0.5, np.arctan2(Y, X)
58. x\_data , y\_data , z\_data = Adams(T , t0 , h , dx , dy , dz, C1 , C2 , C3 , m)
59. C, S = np.cos(Theta), np.sin(Theta)
61. U, V = x\_data\*C - R\*S\*y\_data, x\_data\*S+R\*C\*y\_data
62. axs[0].streamplot(X, Y, U, V, color='r', linewidth=0.5, density=1.6)
64. U, V = y\_data\*C - R\*S\*z\_data, y\_data\*S+R\*C\*z\_data
65. axs[1].streamplot(X, Y, U, V, color='r', linewidth=0.5, density=1.6)
67. U, V = x\_data\*C - R\*S\*z\_data, x\_data\*S+R\*C\*z\_data
68. axs[2].streamplot(X, Y, U, V, color='r', linewidth=0.5, density=1.6)
70. axs[0].set\_title('dx/dy - PhasePortret')
71. axs[1].set\_title('dy/dz - PhasePortret')
72. axs[2].set\_title('dx/dz - PhasePortret')

75. **def** dx(t,x,y,z):
76. **return** -y - z
78. **def** dy(t,x,y,z):
79. **return** x + (1./5.) \* y
81. **def** dz(t,x,y,z,m):
82. **return** 1./5. + z \* (x - m)
84. *# Через данный метод , я найду начальные значения для метода Адамса*
85. *# его можно использовать , тк их порядки совпадают*
87. **def** runge\_kutta\_4(T , t0 , h , dx , dy , dz, C1 , C2 , C3 , m):
88. n = round((T-t0)/h)
89. xn = np.zeros(n)
90. yn = np.zeros(n)
91. zn = np.zeros(n)
92. xn[0] = C1
93. yn[0] = C2
94. zn[0] = C3
95. t = t0
96. **for** i **in** range(1, 4):
97. k1\_x = dx(t, xn[i-1] , yn[i-1] , zn[i-1])
98. k2\_x = dx(t + h/2 ,xn[i-1] + (h\*k1\_x)/2 , yn[i-1] + (h\*k1\_x)/2, zn[i-1] + (h\*k1\_x)/2 )
99. k3\_x = dx(t + h/2 ,xn[i-1] + (h\*k2\_x)/2 , yn[i-1] + (h\*k2\_x)/2, zn[i-1] + (h\*k2\_x)/2 )
100. k4\_x = dx(t + h , xn[i-1] + h \* k3\_x , yn[i-1] + h \* k3\_x , zn[i-1] + h \* k3\_x)
102. xn[i] = xn[i-1] + h/6. \* (k1\_x + 2 \* k2\_x + 2 \* k3\_x + k4\_x)
104. k1\_y = dy(t, xn[i-1] , yn[i-1] , zn[i-1])
105. k2\_y = dy(t + h/2 ,xn[i-1] + (h\*k1\_y)/2 , yn[i-1] + (h\*k1\_y)/2, zn[i-1] + (h\*k1\_y)/2 )
106. k3\_y = dy(t + h/2 ,xn[i-1] + (h\*k2\_y)/2 , yn[i-1] + (h\*k2\_y)/2, zn[i-1] + (h\*k2\_y)/2 )
107. k4\_y = dy(t + h , xn[i-1] + h \* k3\_y , yn[i-1] + h \* k3\_y , zn[i-1] + h \* k3\_y)
109. yn[i] = yn[i-1] + h/6. \* (k1\_y + 2 \* k2\_y + 2 \* k3\_y + k4\_y)
111. k1\_z = dz(t, xn[i-1] , yn[i-1] , zn[i-1] , m)
112. k2\_z = dz(t + h/2 ,xn[i-1] + (h\*k1\_z)/2 , yn[i-1] + (h\*k1\_z)/2, zn[i-1] + (h\*k1\_z)/2 , m)
113. k3\_z = dz(t + h/2 ,xn[i-1] + (h\*k2\_z)/2 , yn[i-1] + (h\*k2\_z)/2, zn[i-1] + (h\*k2\_z)/2 , m)
114. k4\_z = dz(t + h , xn[i-1] + h \* k3\_z , yn[i-1] + h \* k3\_z , zn[i-1] + h \* k3\_z , m)
116. zn[i] = zn[i-1] + h/6. \* (k1\_z + 2 \* k2\_z + 2 \* k3\_z + k4\_z)
117. t+=h
118. **return** xn , yn , zn
120. **def** Adams(T , t0 , h , dx , dy , dz, C1 , C2 , C3 , m):
121. n = round((T - t0) / h)
122. xn = np.zeros(n)
123. yn = np.zeros(n)
124. zn = np.zeros(n)
125. xn\_help , yn\_help , zn\_help = runge\_kutta\_4(T , t0 , h , dx , dy , dz, C1 , C2 , C3 , m)
126. xn[0] = xn\_help[0] ; xn[1] = xn\_help[1] ; xn[2] = xn\_help[2] ; xn[3] = xn\_help[3]
127. yn[0] = yn\_help[0] ; yn[1] = yn\_help[1] ; yn[2] = yn\_help[2] ; yn[3] = yn\_help[3]
128. zn[0] = zn\_help[0] ; zn[1] = zn\_help[1] ; zn[2] = zn\_help[2] ; zn[3] = zn\_help[3]
129. t = t0
130. **for** i **in** range(4, n):
131. xn[i] = xn[i-1] + h/24. \* (55 \* dx(t , xn[i-1] , yn[i-1] , zn[i-1]) - 59 \*
132. dx(t , xn[i-2] , yn[i-2] , zn[i-2]) + 37 \*
133. dx(t , xn[i-3] , yn[i-3] , zn[i-3]) - 9  \*
134. dx(t , xn[i-4] , yn[i-4] , zn[i-4]))
136. yn[i] = yn[i-1] + h/24. \* (55 \* dy(t , xn[i-1] , yn[i-1] , zn[i-1]) - 59 \*
137. dy(t , xn[i-2] , yn[i-2] , zn[i-2]) + 37 \*
138. dy(t , xn[i-3] , yn[i-3] , zn[i-3]) - 9  \*
139. dy(t , xn[i-4] , yn[i-4] , zn[i-4]))
141. zn[i] = zn[i-1] + h/24. \* (55 \* dz(t , xn[i-1] , yn[i-1] , zn[i-1] , m) - 59 \*
142. dz(t , xn[i-2] , yn[i-2] , zn[i-2] , m) + 37 \*
143. dz(t , xn[i-3] , yn[i-3] , zn[i-3] , m) - 9  \*
144. dz(t , xn[i-4] , yn[i-4] , zn[i-4] , m))
145. t += h
146. **return** xn , yn , zn

# Используемая литература.

* А.А. Амосов, Ю.А. Дубинский , Н.В. Копченова: “Вычислительные методы для инженеров”
* И.Г. Петровский: “Лекции по теории обыкновенных дифференциальных уравнений”
* Классификация точек покоя двумерных линейных однородных систем дифференциальных уравнений первого порядка   
  / Учебное пособие. М.: Физический факультет МГУ, 2012. 29 с.