

УДК 517.9  
ББК 22.161.61  
Е80

**Ершов Н. М.**

Е80 Практическое введение в решение дифференциальных уравнений в Python. – М.: ДМК Пресс, 2022. – 176 с.: ил.

**ISBN 978-5-93700-147-4**

Книга посвящена вопросам практического применения символьных вычислений для решения различных прикладных задач, приводящих к дифференциальным уравнениям и их системам, с использованием библиотеки символьных вычислений SymPy языка Python.

Издание ориентировано на школьников старших классов, студентов, преподавателей и всех, интересующихся тематикой математического моделирования. Может служить дополнением к классическим учебникам по теории обыкновенных дифференциальных уравнений.

УДК 517.9  
ББК 22.161.61

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-5-93700-147-4

© Н. М. Ершов, 2022  
© Оформление, издание, ДМК Пресс, 2022

# Оглавление

	<b>Введение</b> .....	5
<b>Глава 1</b>	<b>Вращение жидкости</b> .....	7
	<i>Задача (7) Модель (9) Упражнения и замечания (18)</i>	
<b>Глава 2</b>	<b>Водяные часы</b> .....	21
	<i>Задача (21) Модель (22) Упражнения и замечания (29)</i>	
<b>Глава 3</b>	<b>Элементарные химические реакции</b> .....	33
	<i>Закон действующих масс (33) Задача (35) Модель (35) Упражнения и замечания (38)</i>	
<b>Глава 4</b>	<b>Задача о четырех жуках</b> .....	41
	<i>Кривые погони (41) Задача (41) Модель (43) Упражнения и замечания (47)</i>	
<b>Глава 5</b>	<b>Барометрическая формула</b> .....	51
	<i>Задача (51) Модель (53) Упражнения и замечания (58)</i>	
<b>Глава 6</b>	<b>Модели роста</b> .....	61
	<i>Модель естественного роста (61) Модель Ферхюльста (62) Модель (62) Упражнения и замечания (68)</i>	
<b>Глава 7</b>	<b>Табулирование функций</b> .....	73
	<i>Метод Эйлера (73) Задача (74) Модель (75) Упражнения и замечания (81)</i>	
<b>Глава 8</b>	<b>Ортогональные траектории</b> .....	85
	<i>Ортогональные семейства кривых (85) Задача (85) Модель (87) Упражнения и замечания (90)</i>	
<b>Глава 9</b>	<b>Математическая вышивка</b> .....	95
	<i>Задача (95) Уравнение Клеро (96) Модель (97) Упражнения и замечания (101)</i>	
<b>Глава 10</b>	<b>Криволинейные зеркала</b> .....	105
	<i>Задача (105) Модель (106) Упражнения и замечания (112)</i>	

Глава	<b>Из пушки на луну</b> .....	115
11	<i>Задача (115) Модель (116) Упражнения и замечания (120)</i>	
Глава	<b>Метроном</b> .....	123
12	<i>Задача (123) Модель (124) Упражнения и замечания (130)</i>	
Глава	<b>Пружинный маятник</b> .....	133
13	<i>Задача (133) Модель (134) Упражнения и замечания (140)</i>	
Глава	<b>Модель Лотки–Вольтерры</b> .....	143
14	<i>Задача (143) Модель (144) Упражнения и замечания (149)</i>	
Глава	<b>Системы реакций первого порядка</b> .....	153
15	<i>Задача (153) Преобразование Лапласа (154) Модель (156) Упражнения и замечания (160)</i>	
Глава	<b>Геодезические линии</b> .....	163
16	<i>Задача (163) Уравнение Эйлера–Лагранжа (165) Модель (166) Упражнения и замечания (171)</i>	

## Введение

Настоящая книга посвящена вопросам практического применения символьных вычислений для решения различных прикладных задач, приводящих к дифференциальным уравнениям и их системам. Как известно, решение дифференциальных уравнений — с одной стороны, процесс многогранный и местами даже творческий, с другой — связан с выполнением больших объемов и чисто рутинной работы: арифметика, алгебраические преобразования, вычисления производных и интегралов и т. д. Такая деятельность является по большому счету чисто механической, поэтому она относительно легко и эффективно алгоритмизируется и может быть реализована программно на любом современном языке программирования.

Программные системы, позволяющие пользователю работать с математическими формулами, выполняя над ними те или иные символьные преобразования, называются системами *символьных вычислений*, или системами *компьютерной алгебры*. Первые такие системы появились еще в 60-х годах прошлого века. В настоящее время пользователям доступны десятки подобного рода систем — от коммерческих (*Maple*, *Mathematica*) до систем с открытым исходным кодом (*Reduce*, *Maxima*), некоторые из которых способны работать уже и на мобильных устройствах.

Система *SymPy*, которой и посвящена данная книга, является по сути обычной *библиотекой* языка *Python*. Такой подход к построению систем символьных вычислений имеет ряд преимуществ. Во-первых, система оказывается открытой и доступной всем без исключения пользователям. Во-вторых, работа с библиотекой *SymPy* в среде *Jupyter Notebook* позволяет проводить символьные вычисления прямо в браузере либо локально, либо удаленно с помощью какого-нибудь об-



РИС. 1 *SymPy* — библиотека *Python* для символьных вычислений

<sup>1</sup> При таком сценарии пользователю вообще не нужно устанавливать локально никакое программное обеспечение.

<sup>2</sup> Заметим, что автором не ставилась цель дать полное описание библиотеки `SymPy`. Такая задача, с одной стороны, является просто неподъемной, библиотека `SymPy` состоит из большого числа модулей, только часть из которых посвящена решению дифференциальных уравнений. Кроме того, данная библиотека все еще активно развивается, ее функционал постоянно расширяется и модифицируется. Более полную и актуальную информацию по работе с библиотекой `SymPy` читателю рекомендуется находить либо на официальном сайте библиотеки <https://www.sympy.org>, либо на профильных ресурсах в сети Интернет, например на сайте <https://stackoverflow.com>.

лачного сервиса<sup>1</sup>, например `Google Colab`. В-третьих, символьные вычисления в рамках данной библиотеки оказываются интегрированными в обычную программу на языке `Python`, при этом пользователю системы оказывается доступным весь функционал языка `Python`, а также вся его инфраструктура в виде бесчисленного набора разнообразнейших пакетов и библиотек.

Предлагаемая читателю книга устроена достаточно просто. Каждая ее глава посвящена рассмотрению одной прикладной модели из физики, химии, биологии и т. д. После теоретического рассмотрения модели и вывода соответствующего ей дифференциального уравнения максимально детально описывается процесс формализации модели и решения возникающих в ней дифференциальных уравнений с помощью библиотеки `SymPy`<sup>2</sup>. Каждая глава сопровождается набором упражнений как технического характера — посчитать интеграл, решить уравнение, построить график, так и исследовательского — построить и исследовать по описанной схеме аналогичную модель.

Автор с благодарностью примет все конструктивные отзывы, комментарии и замечания относительно структуры и содержания настоящего издания, которые читатель может оставить или в телеграм-чате <https://t.me/odesinsympy>, или прислать автору на электронную почту по адресу [ershovnm@gmail.com](mailto:ershovnm@gmail.com).

# ГЛАВА 1

## Вращение жидкости

**Задача** • В сосуд, имеющий форму прямого кругового цилиндра, налита жидкость, например вода. Сосуд вращается с постоянной угловой скоростью  $\omega$  относительно оси цилиндра (рис. 1). Требуется определить, какую форму примет поверхность жидкости, если вращение продолжается достаточно долго<sup>1</sup>. При построении модели мы будем предполагать, что сосуд достаточно широкий и глубокий, это позволит пренебречь разными поверхностными эффектами около боковых стенок сосуда.

Очевидно, что искомая поверхность должна быть поверхностью вращения. Поэтому для нахождения ее формы достаточно рассмотреть осевое сечение нашего сосуда и найти форму соответствующей *кривой*, из которой потом мы сможем сформировать и саму поверхность.

Введем в модель систему координат, как это показано на рис. 1. Ось  $Oy$  направим по оси цилиндра, ось  $Ox$  — перпендикулярно оси  $Oy$  вдоль основания цилиндра, начало координат, таким образом, оказывается в центре основания цилиндра.

Рассмотрим малый объем  $\nu$  жидкости на ее поверхности (точка  $P$  на рис. 2). Пусть его масса равна  $m$ . На этот объем действуют две силы: сила тяжести  $F = mg$  и сила реакции опоры  $N$  со стороны всей остальной жидкости. Сила тяжести, как обычно, действует вертикально вниз, а реакция опоры — по нормали к нашей искомой кривой, т. е. перпендикулярно касательной к этой кривой в точке  $P$ .

Обозначим угол наклона данной касательной к оси  $Ox$  через  $\alpha$  и разложим силу  $N$  на горизонтальную  $N_x$

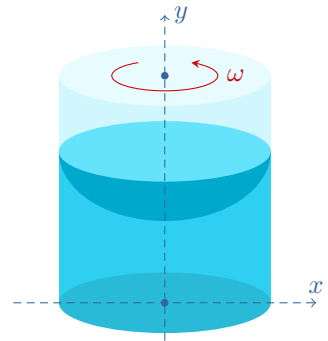


РИС. 1 Вращение сосуда с жидкостью

<sup>1</sup> «Достаточно долго» понимается здесь в том смысле, что вся жидкость в сосуде должна прийти в стационарное состояние относительно самого сосуда, т. е. каждый элементарный ее объем будет совершать только общее вращательное движение с заданной угловой скоростью  $\omega$ .

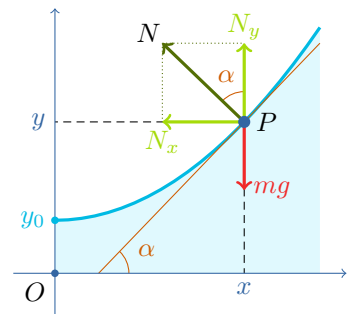


РИС. 2 Силы, действующие на малый объем жидкости  $\nu$  массы  $m$  в точке  $P(x, y)$

и вертикальную  $N_y$  составляющие (рис. 2):

$$N_x = N \sin \alpha, \quad N_y = N \cos \alpha.$$

Запишем теперь 2-й закон Ньютона отдельно для каждой координаты. Вдоль вертикальной оси наш объем  $\nu$  находится в состоянии равновесия, поэтому суммарная сила, действующая на него в вертикальном направлении, должна быть равна нулю:

$$N \cos \alpha - mg = 0. \quad (1)$$

Единственная сила, действующая на объем  $\nu$  вдоль горизонтального направления, — это проекция реакции опоры  $N_x$ . Так как данный объем совершает круговое движение с радиусом вращения  $x$ , то он испытывает центростремительное ускорение  $a$ , которое направлено горизонтально к оси вращения (рис. 3). Величина этого ускорения вычисляется по формуле (квадрат угловой скорости на радиус вращения):

$$a = \omega^2 x. \quad (2)$$

Таким образом, вдоль оси  $Ox$  второй закон Ньютона записывается в следующем виде:

$$-N \sin \alpha = -ma = -m\omega^2 x, \quad (3)$$

обе части взяты с отрицательным знаком, потому что и сила  $N_x$ , и ускорение  $a$  направлены против положительного направления оси  $Ox$ .

Запишем формулы (1) и (3) в виде системы двух уравнений:

$$\begin{aligned} Ox : N \sin \alpha &= m\omega^2 x, \\ Oy : N \cos \alpha &= mg. \end{aligned} \quad (4)$$

Поделим первое уравнение системы на второе:

$$\operatorname{tg} \alpha = \frac{\omega^2 x}{g}.$$

Теперь учтем тот факт, что согласно геометрическому смыслу производной функции  $y(x)$  величина производной в точке  $x_0$  равна тангенсу угла наклона касательной к соответствующей кривой  $y(x)$  в заданной точке  $x_0$ . То есть мы можем заменить  $\operatorname{tg} \alpha$  на  $y'$ , что и даст нам искомое дифференциальное уравнение:

$$y' = \frac{\omega^2 x}{g}. \quad (5)$$

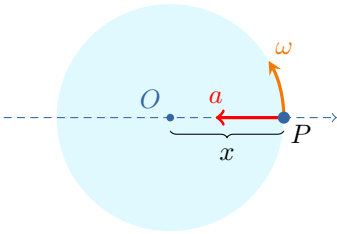


РИС. 3 Центростремительное ускорение для точки  $P$  (вид на сосуд сверху)

Если дополнительно обозначить высоту жидкости в точке  $x = 0$  (на оси вращения) через  $y_0$ , то можно поставить для полученного дифференциального уравнения задачу Коши:

$$\begin{aligned} y' &= \frac{\omega^2 x}{g}, \\ y(0) &= y_0. \end{aligned} \quad (6)$$

**Модель** • Рассмотрим теперь процесс решения поставленной начальной задачи с помощью библиотеки `SymPy`. Сначала мы построим общее решение дифференциального уравнения (5), затем, используя начальное условие  $y(0) = y_0$ , решим начальную задачу (6). Следующим шагом определим параметр  $y_0$  из условия постоянства объема вращаемой жидкости, рассмотрев при этом два случая — подкритический, когда поверхность вращаемой жидкости не касается дна цилиндра, и надкритический, когда такое касание происходит.

**1** Построение модели начинаем с подключения библиотеки `SymPy` и функции `display`, определенной в модуле `IPython.display`.

```
1 from sympy import *
2 from IPython.display import display
```

**2** С помощью команды `symbols` создаем символы  $x$ ,  $g$ ,  $\omega$ ,  $y_0$  и  $C_1$ , которые нам потребуются для задания и решения дифференциального уравнения. Аргументом этой команды является строка, содержащая вводимые символы и определяющая, как эти символы будут выглядеть при печати. Возвращает эта команда кортеж с объектами `SymPy`, которые нужно запомнить в соответствующих переменных для последующего использования. Так как параметр  $g$ , представляющий собой ускорение свободного падения, является всегда строю положительным, то при создании соответствующего символа укажем этот факт, используя опциональный аргумент `positive`<sup>2</sup>. Используем команду `display`, чтобы посмотреть, как выглядят определенные нами символы. Например, видно, что `SymPy` корректно отображает греческие буквы, а цифры после символов преобразует в нижние индексы этих символов<sup>3</sup>.

```
1 x, omega, y0, C1 = symbols("x omega y0 C1")
2 g = symbols("g", positive = True)
3 display(x, g, omega, y0, C1)
```

**2** Информация о положительности того или иного символа может быть использована библиотекой `SymPy` для упрощения формул, в которые входит этот символ, например, при извлечении квадратных корней.

**3** Для краткости вывод команды `display` в данном случае мы поместили в одну строку, на самом деле эта команда выводит каждое из переданных ей выражений в отдельной строке.



$x$   $g$   $\omega$   $y_0$   $C_1$

3 Наше дифференциальное уравнение

$$y' = \frac{\omega^2 x}{g}$$

является простейшим, это значит, что оно решается непосредственным интегрированием его правой части. Следовательно, для решения этого уравнения с помощью библиотеки `SymPy` нам достаточно определить только его правую часть, а не все уравнение целиком<sup>4</sup>. Создадим переменную `ode_rhs`, в которую и запишем правую часть решаемого уравнения.

<sup>4</sup> Как это мы будем делать в дальнейшем при решении более сложных дифференциальных уравнений.

```
1 ode_rhs = omega ** 2 * x / g
2 display(ode_rhs)
```

$$\frac{\omega^2 x}{g}$$

4 Общее решение простейшего дифференциального уравнения  $y' = f(x)$  — это неопределенный интеграл от правой части  $f(x)$  этого уравнения. В библиотеке `SymPy` неопределенные интегралы вычисляются с помощью команды `integrate(f, x)`, где `f` — подынтегральное выражение, `x` — переменная интегрирования. Константа интегрирования  $C_1$  при этом автоматически командой `integrate` к найденному интегралу не прибавляется, это надо делать вручную<sup>5</sup>, поэтому мы просто к результату работы функции `integrate` прибавим символ `C1`. Найденное *общее решение* дифференциального уравнения сохраним в переменной `dsol`.

<sup>5</sup> Если это необходимо.

```
1 dsol = integrate(ode_rhs, x) + C1
2 display(dsol)
```

$$C_1 + \frac{\omega^2 x^2}{2g}$$

5 Стандартный способ определения константы интегрирования  $C_1$  в общем решении дифференциального уравнения из начального условия заключается в подстановке в это решение величин из начального условия, что приводит к алгебраическому уравнению для переменной  $C_1$ . Решив это уравнение и подставив

найденное решение вместо  $C_1$  обратно в формулу общего решения, мы найдем решение соответствующей начальной задачи для заданного дифференциального уравнения. В нашем случае начальное условие имеет вид  $y(0) = y_0$ , поэтому мы сначала подставляем в общее решение `dsol` вместо `x` значение `0`, это делается с помощью метода `subs(f, g)`, где `f` — выражение, которое мы хотим заменить на выражение `g`. Полученное в результате подстановки выражение приравниваем к величине `y0`, используя команду `Eq(lhs, rhs)`, которая создает равенство вида `lhs = rhs`<sup>6</sup>. Созданное уравнение запоминаем в переменной `eq1`.

```
1 eq1 = Eq(dsol.subs(x, 0), y0)
2 display(eq1)
```

$$C_1 = y_0$$

**6** Следующим шагом решаем построенное уравнение с помощью команды `solveset(eq, x)`, где первый аргумент `eq` — это само уравнение, второй аргумент `x` — переменная, относительно которой данное уравнение должно решаться<sup>7</sup>. Найденное решение сохраним в переменной `sol1`.

```
1 sol1 = solveset(eq1, C1)
2 display(sol1)
```

$$\{y_0\}$$

**7** Как мы видим, команда `solveset` выдает решения уравнения в виде множества, чтобы выбрать какое-то одно решение, нужно преобразовать это множество в кортеж (tuple) или список (list) и выбрать соответствующий элемент, указав его индекс. В нашем случае множество содержит всего одно решение, поэтому для его извлечения используем нулевой индекс. Запоминаем найденное значение в переменной `C2`.

```
1 C2 = tuple(sol1)[0]
2 display(C2)
```

$$y_0$$

**8** Последним действием подставляем найденное значение `C2` вместо символа `C1` в общее решение `dsol`, что и дает нам искомое решение начальной задачи. Сохраним это решение в переменной `dsol_y0`.

```
1 dsol_y0 = dsol.subs(C1, C2)
2 display(dsol_y0)
```

<sup>6</sup> Python не позволяет переопределить команду присваивания `=`, поэтому для создания уравнений приходится использовать специальную функцию `Eq`.

<sup>7</sup> В рассматриваемом случае уравнение тривиально, но ровно такие же действия нужно будет выполнять и в более сложных случаях.

$$y_0 + \frac{\omega^2 x^2}{2g}$$

9 Как видно из последней формулы, найденная зависимость высоты  $y$  поверхности жидкости от расстояния до оси вращения  $x$  оказывается квадратичной, т. е. интегральные кривые (графики решений дифференциального уравнения) должны быть парабололами<sup>8</sup>. Убедимся в этом, построив графики решения. Библиотека `SymPy` имеет несколько простых функций для построения графиков<sup>9</sup>, простейшей из которых является функция `plot`. Построим с помощью этой функции график найденного нами решения начальной задачи при следующих значениях входящих в это решение констант:

$$y_0 = 0, \omega = \pi/2, g = 9.8.$$

Чтобы сделать в некоторой формуле сразу несколько подстановок, можно создать из них словарь, ключами которого будут заменяемые символы или выражения, и подать этот словарь на вход метода `subs`.

```
1 ds = dsol_y0.subs({y0: 0, omega: pi/2, g: 9.8})
2 p1 = plot(ds)
```

Результат выполнения<sup>10</sup> этого фрагмента кода показан на рис. 4.

10 Построим теперь графики семейства решений начальной задачи для различных значений угловой скорости  $\omega \in [0, 2\pi]$  с шагом  $\pi/10$ . Для формирования списка значений переменной `omega` будем использовать функцию `arange` из библиотеки `NumPy`. Для построения нескольких кривых на одном графике нужно сначала создать пустой график и запомнить его в некоторой переменной, в нашем случае это будет переменная `p2`. Для каждой кривой следует создавать свой график (переменная `p`) и затем добавлять его к общему графику `p2` с помощью метода `extend`. По умолчанию любой создаваемый график в среде `Jupyter` сразу же визуализируется, чтобы избежать этого, нужно в команде `plot` использовать опцию `show = False`. Построенный график можно визуализировать с помощью вызова метода `show`.

```
1 from numpy import arange
2 p2 = plot(show = False)
3 for om in arange(0, 2 * pi, pi / 10):
```

8 Таким образом, искомая поверхность будет параболоидом вращения.

9 Являющихся, по сути, надстройкой над существенно более мощной графической библиотекой `Matplotlib`.

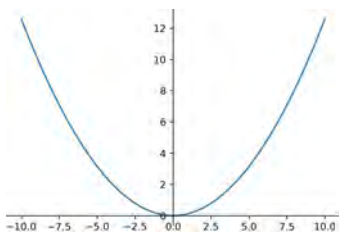


РИС. 4 График решения начальной задачи

10 Переменная `pi` является предопределенной константой в `SymPy`.

```

4 ds = dsol_y0.subs({y0: 0, omega: om, g: 9.8})
5 p = plot(ds, (x, -1, 1), show = False)
6 p2.extend(p)
7 p2.show()

```

Результат выполнения этого кода приведен на рис. 5.

**11** При построении графиков решения выше мы *произвольно* задавали значение параметра  $y_0$  равным нулю. Более естественный подход к определению этого параметра заключается в вычислении объема вращающейся жидкости и учете очевидного условия, что этот объем является постоянной величиной, не зависящей от скорости вращения  $\omega$ . Пусть объем жидкости равен  $V_0$ . Чтобы выразить величину  $y_0$  через  $V_0$ , нам надо вычислить объем вращающейся жидкости, приравняв его к  $V_0$  и решить полученное уравнение относительно  $y_0$ . Заметим, что интересующий нас объем получается вращением участка кривой  $y(x)$  на отрезке  $[0, R]$  вокруг оси  $Oy$ , где  $R$  — радиус основания цилиндра (рис. 6). Следовательно, для его вычисления можно воспользоваться двойным интегралом в полярных координатах<sup>11</sup>:

$$V = \int_0^{2\pi} \int_0^R f(r) r dr d\varphi.$$

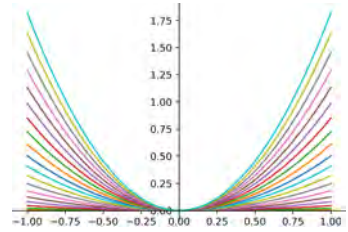
Роль радиуса  $r$  в нашем случае играет переменная  $x$ , а функция  $f(r)$  представлена решением начальной задачи  $y(x)$ , следовательно,

$$V = \int_0^{2\pi} \int_0^R y(x) x dx d\varphi.$$

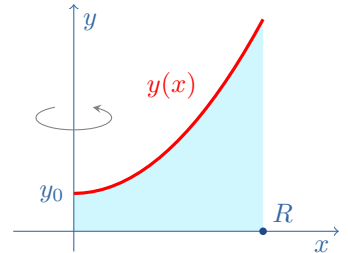
Данный двойной интеграл можно упростить с учетом того, что подынтегральное выражение не зависит от полярного угла  $\varphi$ :

$$V = 2\pi \int_0^R y(x) x dx. \tag{7}$$

Введем два дополнительных символа для радиуса цилиндра и объема содержащейся в нем жидкости (оба параметра являются строго положительными величинами), вычислим *определенный* интеграл (7) и запомним результат в переменной `vol`. Определенные интегралы в библиотеке `SymPy` вычисляются с помощью той



**РИС. 5** График семейства решений начальной задачи для разных значений угловой скорости  $\omega$



**РИС. 6** Вычисление объема вращающейся жидкости

<sup>11</sup> Наша поверхность является поверхностью вращения, поэтому функция  $f(r)$  не зависит от полярного угла  $\varphi$ .

же функции `integrate`, но вторым ее аргументом для этого нужно указать кортеж из трех элементов: переменная интегрирования, нижний и верхний пределы интегрирования.

```
1 V0, R = symbols("V0 R", positive = True)
2 vol = 2 * pi * integrate(dsol_y0 * x, (x, 0, R))
3 display(vol)
```

$$2\pi \left( \frac{R^4 \omega^2}{8g} + \frac{R^2 y_0}{2} \right)$$

**12** Приравниваем найденный объем к `V0` и получаем уравнение относительно переменной `y0`.

```
1 eq2 = Eq(vol, V0)
2 display(eq2)
```

$$2\pi \left( \frac{R^4 \omega^2}{8g} + \frac{R^2 y_0}{2} \right) = V_0$$

**13** Решаем полученное уравнение относительно `y0`.

```
1 sol2 = solveset(eq2, y0)
2 display(sol2)
```

$$\left\{ -\frac{\frac{R^4 \omega^2}{g} - \frac{4V_0}{\pi}}{4R^2} \right\}$$

**14** Уравнение `eq2` линейное, поэтому решение у него единственное, для его извлечения используем нулевой индекс. Заодно упростим полученную формулу, используя команду `simplify`.

```
1 y1 = simplify(tuple(sol2)[0])
2 display(y1)
```

$$-\frac{R^2 \omega^2}{4g} + \frac{V_0}{\pi R^2}$$

**15** Подставляем найденное выражение `y1` вместо символа `y0` в решение `dsol_y0` начальной задачи. Получаем еще одно выражение для решения этой же начальной задачи, но теперь выраженное через заданный объем жидкости `V0`.

```
1 dsol_V0 = dsol_y0.subs(y0, y1)
2 display(dsol_V0)
```

$$-\frac{R^2\omega^2}{4g} + \frac{\omega^2x^2}{2g} + \frac{V_0}{\pi R^2}$$

16 Построим еще раз семейство итегральных кривых нашей задачи для разных значений скорости вращения  $\omega$  для случая  $V_0 = 2$  (рис. 7).

```
1 p3 = plot(show = False)
2 for om in arange(0, 2 * pi, pi / 10):
3     ds = dsol_V0.subs({V0:2, R:1, omega:om, g:9.8})
4     p = plot(ds, (x, -1, 1), show = False)
5     p3.extend(p)
6 p3.show()
```

17 Видно, что поведение интегральных кривых теперь стало более естественным — нижняя точка параболоида постепенно опускается при увеличении скорости вращения. Однако при достижении дна цилиндра ( $y = 0$ ) это поведение становится нереалистичным — поверхность жидкости в некоторой области оказывается ниже дна цилиндра, а интегрирование по этой области дает отрицательный объем. То есть наши выкладки справедливы только при условии, что кривая  $y(x)$  располагается строго над осью  $Ox$ . Найдем критическую скорость  $\omega_0$ , при которой нижняя точка поверхности жидкости (при  $x = 0$ ) касается дна цилиндра. Для этого подставим в решение  $x = 0$ , приравняем его к нулю и решим полученное уравнение относительно скорости вращения  $\omega$ .

```
1 eq3 = Eq(dsol_V0.subs(x, 0), 0)
2 sol3 = solveset(eq3, omega)
3 display(sol3)
```

$$\left\{ -\frac{2\sqrt{V_0}\sqrt{g}}{\sqrt{\pi R^2}}, \frac{2\sqrt{V_0}\sqrt{g}}{\sqrt{\pi R^2}} \right\}$$

18 Команда `solve` в данном случае выдает два решения, отличающихся только знаком (т. е. направлением вращения), выберем из них положительное (с индексом 1) и сохраним его в переменной `omega0`. Построим графики интегральных кривых для случая  $\omega < \omega_0$  (рис. 8). Этот случай назовем *подкритическим*. Для удобства заранее создадим словарь `par` для выполнения дальнейших подстановок. Опции `xlim` и `ylim` команды `plot` определяют пределы изменения переменных по горизонтальной и вертикальной осям графика. Кривые рисуем синим цветом, используя опцию `line_color` команды `plot`.

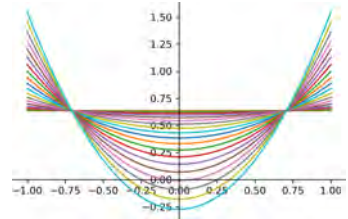


РИС. 7 График семейства решений начальной задачи для разных значений угловой скорости  $\omega$  при фиксированном объеме  $V_0$

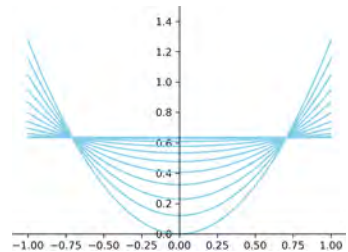


РИС. 8 График семейства решений начальной задачи в подкритическом случае

```

1 omega0 = tuple(sol3)[1]
2 display(omega0)
3 p4 = plot(ylim = (0, 1.5), show = False)
4 par = {V0: 2, omega: 0, R: 1, g: 9.8}
5 om0 = omega0.subs(par)
6 for om in arange(0, 1.1 * om0, 0.1 * om0):
7     par[omega] = om
8     ds = dsol_V0.subs(par)
9     p = plot(ds, (x, -1, 1), line_color = "skyblue",
10            show = False)
11     p4.extend(p)
12 p4.show()

```

$$\left| \frac{2\sqrt{V_0}\sqrt{g}}{\sqrt{\pi}R^2} \right|$$

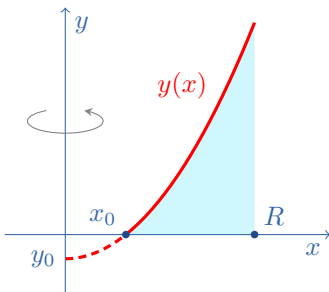
**19** Рассмотрим теперь по аналогичной схеме *надкритический* случай, когда скорость вращения превышает критическую, что приводит к образованию в центре дна цилиндра области, не покрытой жидкостью. Первым шагом найдем радиус  $x_0$  этой области. Для этого приравняем решение `dsol_y0` к нулю и решим полученное уравнение относительно  $x$ .

```

1 eq4 = Eq(dsol_y0, 0)
2 sol4 = solveset(eq4, x)
3 display(sol4)

```

$$\left| \left\{ -\frac{\sqrt{2}\sqrt{g}\sqrt{-y_0}}{\omega}, \frac{\sqrt{2}\sqrt{g}\sqrt{-y_0}}{\omega} \right\} \right|$$



**РИС. 9** Вычисление объема вращающейся жидкости в надкритическом случае

**20** Из двух найденных решений выберем решение со знаком плюс. И еще раз вычислим объем, занимаемый жидкостью. Единственное отличие от предыдущего случая состоит в том, что интегрирование теперь выполняется от  $x_0$  до  $R$ , т. е. часть ниже оси  $Ox$  мы не интегрируем (см. рис. 9):

$$V = 2\pi \int_{x_0}^R y(x) x dx.$$

```

1 x0 = tuple(sol4)[0]
2 display(x0)
3 vol1 = 2 * pi * integrate(dsol_y0 * x, (x, x0, R))
4 display(vol1)

```

$$\frac{\sqrt{2}\sqrt{g}\sqrt{-y_0}}{\omega} \left( \frac{R^4\omega^2}{8g} + \frac{R^2y_0}{2} + \frac{gy_0^2}{2\omega^2} \right)$$

21 Приравниваем вычисленное выражение к  $V_0$  и решаем полученное уравнение относительно  $y_0$ .

```
1 eq5 = Eq(vol1, V0)
2 sol5 = solveset(eq5, y0)
3 display(sol5)
```

$$\left\{ -\frac{R^2\omega^2}{2g} - \frac{\sqrt{V_0}\omega}{\sqrt{\pi}\sqrt{g}}, -\frac{R^2\omega^2}{2g} + \frac{\sqrt{V_0}\omega}{\sqrt{\pi}\sqrt{g}} \right\}$$

22 Получили два возможных значения параметра  $y_0$ . Проверим, какое из них оказывается равным нулю при подстановке вместо  $\omega$  критического значения  $\omega_0$ .

```
1 for s in tuple(sol5):
2     display(s.subs(omega, omega0))
```

$$0$$

$$-\frac{4V_0}{\pi R^2}$$

23 Выбираем решение с индексом 0 и подставляем его в решение начальной задачи `dsol_y0` вместо символа  $y_0$ , что дает нам уравнение поверхности для надкритического случая.

```
1 y2 = tuple(sol5)[0]
2 dsol_sup = dsol_y0.subs(y0, y2)
3 display(dsol_sup)
```

$$-\frac{R^2\omega^2}{2g} + \frac{\sqrt{V_0}\omega}{\sqrt{\pi}\sqrt{g}} + \frac{\omega^2x^2}{2g}$$

24 По аналогичной схеме строим семейство кривых (красного цвета) найденного решения для скорости  $\omega \in [1.2\omega_0, 3\omega_0]$  с шагом  $0.2\omega_0$  (рис. 10).

```
1 p5 = plot(ylim = (0, 1.5), show = False)
2 for om in arange(1.2 * om0, 3 * om0, 0.2 * om0):
3     par[omega] = om
4     ds = dsol_sup.subs(par)
5     p = plot(ds, (x, -1, 1), line_color = "red",
6             show = False)
7     p5.extend(p)
8 p5.show()
```

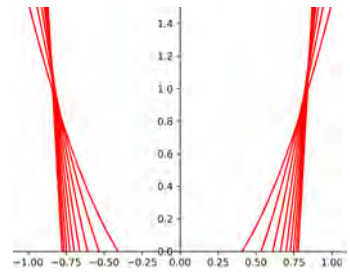
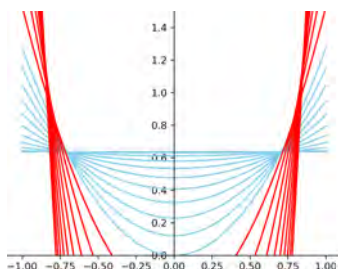


РИС. 10 График семейства решений начальной задачи в надкритическом случае





**РИС. 11** График семейства решений начальной задачи в подкритическом (синие кривые) и надкритическом (красные кривые) случаях

**ТАБЛ. 1** Стандартные математические функции в `SymPy`

Функция	<code>SymPy</code>
$\sqrt{x}$	<code>sqrt(x)</code>
$e^x$	<code>exp(x)</code>
$\ln x$	<code>log(x)</code>
$\sin x$	<code>sin(x)</code>
$\cos x$	<code>cos(x)</code>
$\operatorname{tg} x$	<code>tan(x)</code>
$\operatorname{ctg} x$	<code>cot(x)</code>
$\arcsin x$	<code>asin(x)</code>
$\arccos x$	<code>acos(x)</code>
$\operatorname{arctg} x$	<code>atan(x)</code>
$\operatorname{arcctg} x$	<code>acot(x)</code>

**12** Библиотека `SymPy` частично поддерживает работу с символом бесконечности  $\infty$ , который определен в ней под именем `oo`.

**25** Наконец, объединим графики двух рассмотренных нами случаев: подкритического (переменная `p4`) и надкритического (переменная `p5`).

```
1 p6 = plot(ylim = (0, 1.5), show = False)
2 p6.extend(p4)
3 p6.extend(p5)
4 p6.show()
```

Итоговый график, включающий оба случая, показан на рис. 11.

## УПРАЖНЕНИЯ И ЗАМЕЧАНИЯ

**1** Решите с помощью `SymPy` следующие квадратные уравнения:

- 1)  $x^2 - 4x + 3 = 0$ ;                      3)  $64x^2 + 16x + 1 = 0$ ;  
 2)  $x^2 - 5x + 1 = 0$ ;                      4)  $2x^2 - 3x + 2 = 0$ .

**2** Решите с помощью `SymPy` следующие кубические уравнения, выведите на экран найденные *положительные* корни:

- 1)  $x^3 - 3x^2 + 3x - 1 = 0$ ;                3)  $x^3 - 2x^2 - 7x - 4 = 0$ ;  
 2)  $x^3 - 7x + 6 = 0$ ;                      4)  $x^3 + 7x^2 + 14x + 8 = 0$ .

**3** Дифференцирование выражений в `SymPy` выполняется командой `diff(f, x)`, где `f` — выражение, `x` — переменная дифференцирования. Вычислите с помощью этой функции производные по `x` следующих выражений (см. табл. 1):

- 1)  $5x^4 + x^2 - 4$ ;                          3)  $e^x \cdot \sqrt{4x + 2}$ ;  
 2)  $ax^2 + bx + c$ ;                          4)  $\arccos(a^2 - x^2)$ .

**4** Вычислите с помощью команды `integrate` следующие *неопределенные* интегралы:

- 1)  $\int (x^2 - 2x + 3) dx$ ;                      3)  $\int se^s ds$ ;  
 2)  $\int y \sin(y^2) dy$ ;                        4)  $\int e^{2t} \cos t dt$ .

**5** Вычислите с помощью команды `integrate` следующие *определенные* интегралы<sup>12</sup>:

- 1)  $\int_{-1}^1 (x^3 - 1) dx$ ;                        3)  $\int_0^{2\pi} \sin 2\varphi d\varphi$ ;  
 2)  $\int_a^b \frac{y-1}{y+1} dy$ ;                        4)  $\int_1^{\infty} e^{-3t} dt$ .

6 Найдите с помощью `SymPy` общее решение простейшего дифференциального уравнения и решите соответствующую начальную задачу:

- 1)  $y' = \frac{1}{x}, y(1) = -1;$
- 2)  $y' = 2 \cos^2 x, y(\pi) = 0;$
- 3)  $y' = \frac{1 + 2x^2}{\sqrt{x}}, y(1) = -\frac{1}{5};$
- 4)  $y' = x \ln x, y(\sqrt{e}) = 1.$

Числовые дроби вида  $n/m$  с целыми  $n$  и  $m$  в `SymPy` задаются командой `Rational(n, m)`.

7 Постройте семейство кривых решения начальной задачи `dsol_y0` для значений параметра  $y_0$  из интервала  $[0, 2]$  с шагом 0.1 при фиксированной скорости вращения  $\omega = 1$ .

8 Постройте семейство кривых решения начальной задачи для разных значений ускорения свободного падения  $g$ , список этих значений сформируйте из величин ускорения свободного падения на поверхности различных космических тел Солнечной системы (Солнце, планеты, Луна).

9 Для подкритического случая найдите положение неподвижной точки  $x_1$ , в которой пересекаются все интегральные кривые рассматриваемого дифференциального уравнения. Может ли эта точка находиться вне цилиндра?

10 Напишите функцию для построения графика решения начальной задачи, автоматически определяющую, к какому случаю относится решение при заданных значениях параметров  $V_0, R, g$  и  $\omega$ .

11 Постройте модель вращения жидкости в сосуде, имеющем форму параболоида вращения  $y = kx^2, k > 0$ .

12 Рассмотрите модель вращения жидкости в круговом цилиндре высоты  $H$ , который закрыт сверху крышкой, в предположении, что поверхность жидкости касается верхнего основания цилиндра при угловой скорости  $\omega_1 < \omega_0$  (т. е. в подкритическом случае, см. рис. 12).

13 Рассмотрите случай вращения сосуда в форме тонкого прямоугольного параллелепипеда, толщина которого  $\delta$  настолько мала, что искривлением поверхности вдоль соответствующей стороны можно пренебречь (рис. 13). Вращение сосуда выполняется вокруг оси, проходящей через его центр перпендикулярно основанию. Объем жидкости во вращающемся сосуде в данном случае вычисляется по

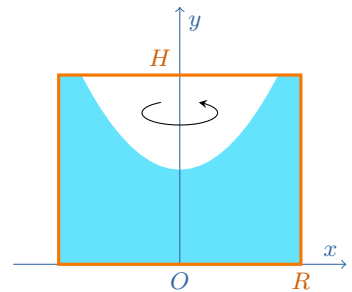


РИС. 12 Вращение жидкости в закрытом сверху цилиндре

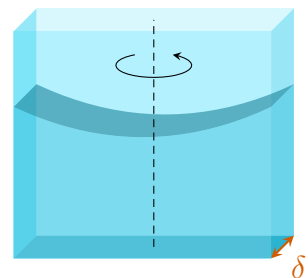


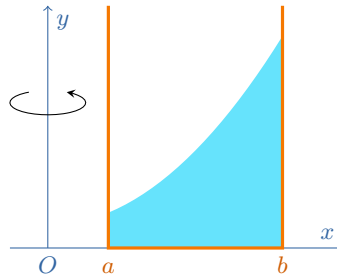
РИС. 13 Вращение тонкого параллелепипеда

более простой формуле:

$$V = 2\delta \int_0^R y(x) dx, \quad (8)$$

где  $R$  — половина длинной стороны основания параллелепипеда. Так как объем  $V$  постоянен и  $\delta$  является константой, то постоянным должен быть интеграл в формуле (8). Найдите величину  $y_0$  в предположении, что объем жидкости равен  $V_0$ . Рассмотрите по аналогии с построенной нами моделью подкритический и надкритический случаи.

**14** Исследуйте модель вращения жидкости в тонком прямоугольном параллелепипеде (см. предыдущее упражнение) для случая, когда ось вращения смещена относительно центра параллелепипеда (см. рис. 14).



**РИС. 14** Модель со смещенной осью вращения