Оглавление

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 алгоритмизуется 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

лачного сервиса[1], 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[2]. 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.

[1] При таком сценарии пользователю вообще не нужно устанавливать локально никакое программное обеспечение.

[2] 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.

ГЛАВА 1

Вращение жидкости

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

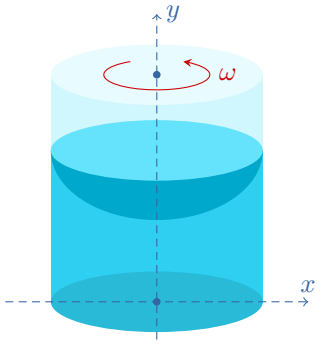

РИС. 1 Вращение сосуда с жидкостью

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 ν 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 α и разложим силу N на горизонтальную N_x

[1] «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 ω.

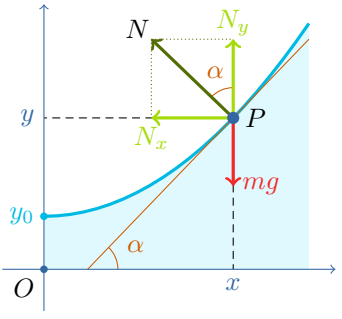

РИС. 2 Силы, действующие на малый объем жидкости ν массы m в точке $P(x, y)$

и вертикальную N_y составляющие (рис. 2):

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

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

$$N\cos\alpha - mg = 0. \tag{1}$$

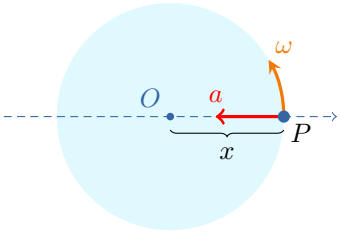

РИС. 3 Центростремительное ускорение для точки P (вид на сосуд сверху)

Единственная сила, действующая на объем ν 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):

$$a = \omega^2 x. \tag{2}$$

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

$$-N\sin\alpha = -ma = -m\omega^2 x, \tag{3}$$

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:

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

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:

$$y' = \frac{\omega^2 x}{g}. \tag{5}$$

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

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

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.

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

2 С помощью команды symbols создаем символы x, g, ω, 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 строго положительным, то при создании соответствующего символа укажем этот факт, используя опциональный аргумент positive[2]. 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[3].

```
x, omega, y0, C1 = symbols("x omega y0 C1")
g = symbols("g", positive = True)
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 переденных 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[4]. Создадим переменную ode_rhs, в которую и запишем правую часть решаемого уравнения.

[4] Как это мы будем делать в дальнейшем при решении более сложных дифференциальных уравнений.

```
ode_rhs = omega ** 2 * x / g
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[5], поэтому мы просто к результату работы функции integrate прибавим символ C1. Найденное *общее решение* дифференциального уравнения сохраним в переменной dsol.

[5] Если это необходимо.

```
dsol = integrate(ode_rhs, x) + C1
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`[6]. Созданное уравнение запоминаем в переменной `eq1`.

```
eq1 = Eq(dsol.subs(x, 0), y0)
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[7]. Найденное решение сохраним в переменной `sol1`.

```
sol1 = solveset(eq1, C1)
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`.

```
C2 = tuple(sol1)[0]
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`.

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

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

7 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 параболами[8]. Убедимся в этом, построив графики решения. Библиотека SymPy имеет несколько простых функций для построения графиков[9], 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:

$$y0 = 0,\ \omega = \pi/2,\ g = 9.8.$$

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 методу subs.

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

Результат выполнения[10] этого фрагмента кода показан на рис. 4.

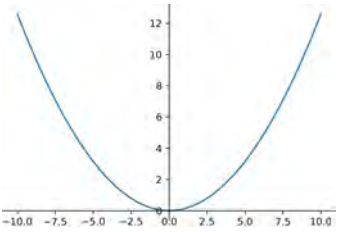

РИС. 4 График решения начальной задачи

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.

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

8 Таким образом, искомая поверхность будет парабалоидом вращения.

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

10 Переменная pi является предопределенной константой в SymPy.

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

Результат выполнения этого кода приведен на рис. 5.

11 При построении графиков решения выше мы *произвольно* задавали значение параметра `y0` 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 ω. Пусть объем жидкости равен V_0. Чтобы выразить величину y_0 через V_0, нам надо вычислить объем вращающейся жидкости, приравнять 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[11]:

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

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 φ:

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

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

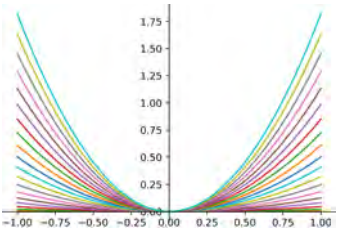

РИС. 5 График семейства решений начальной задачи для разных значений угловой скорости ω

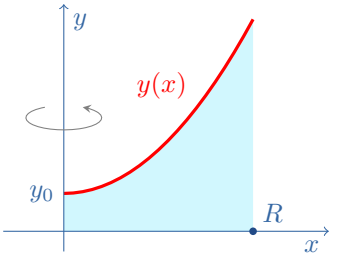

РИС. 6 Вычисление объема вращающейся жидкости

11 Наша поверхность является поверхностью вращения, поэтому функция $f(r)$ не зависит от полярного угла φ.

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

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

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

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

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

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

13 Решаем полученное уравнение относительно `y0`.

```
sol2 = solveset(eq2, y0)
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`.

```
y1 = simplify(tuple(sol2)[0])
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`.

```
dsol_V0 = dsol_y0.subs(y0, y1)
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 ω для случая $V_0 = 2$ (рис. 7).

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

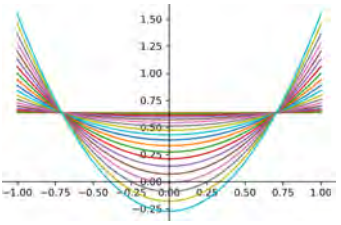

РИС. 7 График семейства решений начальной задачи для разных значений угловой скорости ω при фиксированном объеме V_0

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 ω_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 ω.

```
eq3 = Eq(dsol_V0.subs(x, 0), 0)
sol3 = solveset(eq3, omega)
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`.

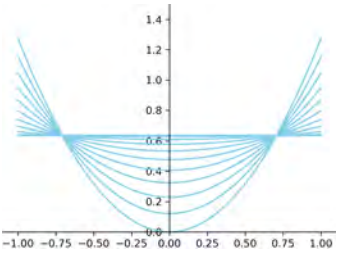

РИС. 8 График семейства решений начальной задачи в подкритическом случае

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

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

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.

```
eq4 = Eq(dsol_y0, 0)
sol4 = solveset(eq4, x)
display(sol4)
```

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

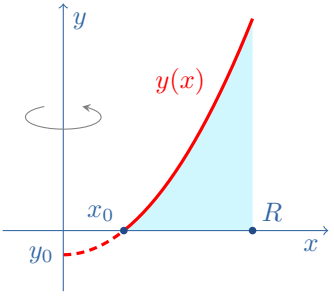

РИС. 9 Вычисление объема вращающейся жидкости в надкритическом случае

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):

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

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

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

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

```
eq5 = Eq(vol1, V0)
sol5 = solveset(eq5, y0)
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 ω критического значения ω_0.

```
for s in tuple(sol5):
    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 `y0`, что дает нам уравнение поверхности для надкритического случая.

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

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

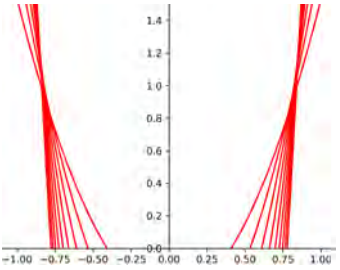

РИС. 10 График семейства решений начальной задачи в надкритическом случае

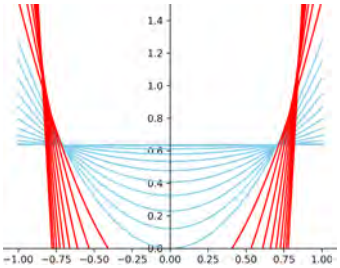

РИС. 11 График семейства решений начальной задачи в подкритическом (синие кривые) и надкритическом (красные кривые) случаях

ТАБЛ. 1 Стандартные математические функции в SymPy

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

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

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

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

УПРАЖНЕНИЯ И ЗАМЕЧАНИЯ

1 Решите с помощью SymPy следующие квадратные уравнения:

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

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

1) $x^3 - 3x^2 + 3x - 1 = 0$;
2) $x^3 - 7x + 6 = 0$;
3) $x^3 - 2x^2 - 7x - 4 = 0$;
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) $5x^4 + x^2 - 4$;
2) $ax^2 + bx + c$;
3) $e^x \cdot \sqrt{4x + 2}$;
4) $\arccos(a^2 - x^2)$.

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

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

5 Вычислите с помощью команды `integrate` следующие *определенные* интегралы[12]:

1) $\int_{-1}^{1} (x^3 - 1)dx$;
2) $\int_{a}^{b} \frac{y-1}{y+1} dy$;
3) $\int_{0}^{2\pi} \sin 2\varphi \, d\varphi$;
4) $\int_{1}^{\infty} e^{-3t} dt$.

[12] Библиотека SymPy частично поддерживает работу с символом бесконечности ∞, который определен в ней под именем `oo`.

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

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

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 δ 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

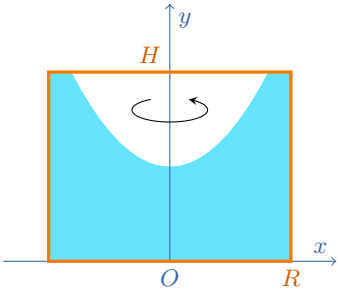

РИС. 12 Вращение жидкости в закрытом сверху цилиндре

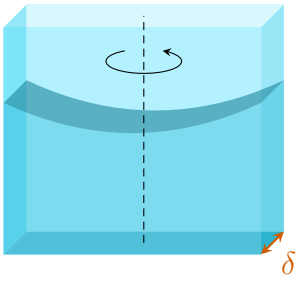

РИС. 13 Вращение тонкого параллелепипеда

более простой формуле:

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

где R — половина длинной стороны основания параллелепипеда. Так как объем V постоянен и δ 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).

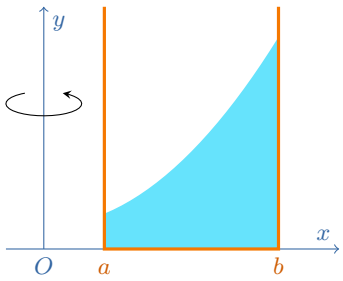

РИС. 14 Модель со смещенной осью вращения