Московский государственый университет
им. М. В. Ломоносова
Механико-математический факультет
Кафедра газовой и волновой динамики

Ракитин Виталий Павлович.

Интегрирование системы ОДУ методом Рунге-Кутта.

Москва, 2015 год.

Содержание

1 Постановка задачи

Задача 2.6: Решить задачу Коши и построить график траектории движения.

$$\begin{cases} \begin{cases} \frac{d^2x}{dt^2} = y(2 - x^2 - y^2); \\ \frac{d^2x}{dt^2} = -x(2 - x^2 - y^2); \end{cases} \\ 0 < t < 30; \\ t = 0 : x = 0, y = \alpha, \frac{dx}{dt} = \frac{dy}{dt} = 0; \\ \alpha \in \{1.0, 0.1\}. \end{cases}$$

Данную систему из 2х уравнений второго порядка можно свести к эквивалентной из 4х уравнений первого порядка. Сделаем замену $\frac{dx}{dt} = u$, $\frac{dy}{dt} = v$, получим

$$\begin{cases} \begin{cases} \frac{dx}{dt} = u; \\ \frac{dy}{dt} = v; \\ \frac{du}{dt} = y(2 - x^2 - y^2); \\ \frac{dv}{dt} = -x(2 - x^2 - y^2); \end{cases} \\ 0 < t < 30; \\ t = 0 : x = 0, y = \alpha, u = v = 0; \\ \alpha \in \{1.0, 0.1\}. \end{cases}$$

Так как данная задача не имеет аналитического решения, напишем компьютерную программу на языке программирования C для численного решения.

2 Метод решения

Для решения данной задачи применим метод Рунге - Кутты 5-го порядка с коэффициентами Дормана-Принса 5(4). Для этого зафиксируем следующие числа

α	β						
0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9071}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
p	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$\bar{p}$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Последовательно получим систему, где $y_n \in \{x_n, y_n, u_n, v_n\}$,

$$\begin{cases} k_1(h) = hf\left(t_n, y_n\right); \\ k_2(h) = hf\left(t_n + \alpha_2 h, y_n + \beta_{21} k_1(h)\right); \\ \dots \\ k_q(h) = hf\left(t_n + \alpha_q h, y_n + \sum\limits_{i=1}^{q-1} \beta_{qi} k_i(h)\right). \end{cases}$$

которая после подставления чисел α_i, p_i, β_i принимает вид

$$\begin{cases} k_1(h) = hf\left(t_n, y_n\right); \\ k_2(h) = hf\left(t_n + \frac{1}{5}h, y_n + \frac{1}{4}k_1\right); \\ k_3(h) = hf\left(t_n + \frac{3}{10}h, y_n + \frac{3}{40}k_1 + \frac{9}{40}k_2\right); \\ k_4(h) = hf\left(t_n + \frac{4}{5}h, y_n + \frac{44}{45}k_1 - \frac{56}{15}k_2 + \frac{32}{9}k_3\right); \\ k_5(h) = hf\left(t_n + \frac{8}{9}h, y_n + \frac{19372}{6561}k_1 - \frac{25360}{2187}k_2 + \frac{64448}{6561}k_3 - \frac{212}{729}k_4\right); \\ k_6(h) = hf\left(t_n + h, y_n + \frac{9017}{3168}k_1 - \frac{355}{33}k_2 + \frac{46732}{5247}k_3 + \frac{49}{176}k_4 - \frac{5103}{18656}k_5\right); \\ k_7(h) = hf\left(t_n + h, y_n + \frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6\right). \end{cases}$$

$$y_n(t+h) = y_n(t) + \sum_{i=1}^{q} p_i k_i = y_n(t) + \frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6;$$

$$\bar{y}_n(t+h) = y_n(t) + \sum_{i=1}^{q} \bar{p}_i k_i = y_n(t) + \frac{517}{57600}k_1 + \frac{7571}{16695}k_3 + \frac{393}{640}k_4 - \frac{92097}{339200}k_5 + \frac{187}{2100}k_6 + \frac{1}{40}k_7.$$

Таким образом величина погрешности имеет порядок точности $O(h^5)$ и её можно оценить двумя способами. Во-первых, как расстояние между двумя точками A и $\bar{A}$

$$Err = \sqrt{(x_n - \bar{x}_n)^2 + (y_n - \bar{y}_n)^2 + (u_n - \bar{u}_n)^2 + (v_n - \bar{v}_n)^2}.$$

Во-вторых, как максимум расстояний между соответствующими координатами, а именно

$$Err = \max\left(x_n - \bar{x}_n, y_n - \bar{y}_n, u_n - \bar{u}_n, v_n - \bar{v}_n\right).$$

Для удобства будем использовать второй вариант.

2.1 Горизонтальная процедура автоматического выбора шага

Для того, чтобы избешать сильного отклонения от допустимого для корректных вычислений значения шага, напишем так же функцию, которая будет автоматически изменять наш шаг в течении всего времени расчётов.

Пусть ε — это заранее заданная точность нашей погрешности Err, а значения корректирующих констант заданы следующим образом (их можно менять по усмотрению программиста)

$$FAC = 0.9, \qquad FACMIN = 0.5, \qquad FACMAX = 2.0.$$

В таком случае формула для изменения шага будет представлена в следующем виде

$$h_{new} = h \cdot \min\left(FACMAX, \max\left(FACMIN, \left(\frac{\varepsilon}{Err}\right)^{\frac{1}{1+p}}\right)\right),$$

где $p = 5$ — порядок алгоритма. Таким образом, при расчётах будем производить коррекцию шага до тех пор, пока не добьёмся нужного порядка погрешности наших вычислений.

3 Оценка корректности результата

Прежде всего нам необходимо знать, можем ли мы доверять полученному результату. Для оценки корректности работы программы будем использовать нам необходимо понять, какова погрешность наших измерений. Для этого будем высчитывать два значения:

1. велечина отклонения на каждом шаге;

2. глобальная велечина отклонения.

3.1 Оценка локального отклонения (на шаге)

Оценивать велечину отклонения на каждом шаге не имеет никакого смысла из-за значительного увелечения времени работы программы. Поэтому, для этой оценки проведём все вышеизложенные вычисления для трёх значений точности велечины отклонения, а именно $-7, -9$ и -11. Далее зафиксируем 4 момента времени: $\frac{T}{4}, \frac{T}{2}, \frac{3T}{4}, T$, где T — полное время работы нашей программы. После этого вычислим все значение переменных в каждой из этих точек, и проверим следующее соотношение

$$variation = \frac{y_{-11} - y_{-9}}{y_{-9} - y_{-7}} \approx 100^{\frac{s}{s+1}} = 100^{\frac{5}{5+1}} \approx 46.42$$

После проведения всех расчётов для $\alpha = 1.0$ были получены следующие величины

При $t = 0.875$ для переменной x: $variation = 2.115109$

При $t = 0.875$ для переменной y: $variation = 2.337501$

При $t = 0.875$ для переменной u: $variation = 1.943925$

При $t = 0.875$ для переменной v: $variation = 2.170464$

При $t = 1.75$ для переменной x: $variation = 1.880996$

При $t = 1.75$ для переменной y: $variation = 1.996141$

При $t = 1.75$ для переменной u: $variation = 1.646851$

При $t = 1.75$ для переменной v: $variation = 1.792992$

При $t = 2.625$ для переменной x: $variation = 1.194104$

При $t = 2.625$ для переменной y: $variation = 1.057303$

При $t = 2.625$ для переменной u: $variation = 1.490886$

При $t = 2.625$ для переменной v: $variation = 1.573281$

При $t = 3.5$ для переменной x: $variation = 83.63506$

При $t = 3.5$ для переменной y: $variation = 1.837813$

При $t = 3.5$ для переменной u: $variation = 6.108553$

При $t = 3.5$ для переменной v: $variation = 1.147372$

3.2 Оценка глобальной погрешности.

Для вычисления глобальной погрешности введём множество переменных δ_i :

$$\delta_0 = 0, \qquad \delta_{k+1} = Err_k + \delta_k \cdot e^{\int\limits_{t_k}^{t_{k+1}} \mu(s)ds}$$

Интеграл в предыдущем выражении можно приблизить следующим образом

$$\int\limits_{t_k}^{t_{k+1}} \mu(s)ds = (t_{k+1} - t_k) \cdot Hmax\left(\frac{J + J^T}{2}\right),$$

где J — матрица Якоби исходной системы дифференциальных уравнений, а $Hmax$ — функция, возвращающая максимальное собственное значение полученной матрицы.

$$J = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -2xy & -x^2 - 3y^2 + 2 & 0 & 0 \\ 3x^2 + y^2 - 2 & 2xy & 0 & 0 \end{pmatrix}$$

Соответственно,

$$A = \frac{J + J^T}{2} = \begin{pmatrix} 0 & 0 & \frac{1}{2}(1 - 2xy) & \frac{1}{2}\left(3x^2 + y^2 - 2\right) \\ 0 & 0 & \frac{1}{2}\left(-x^2 - 3y^2 + 2\right) & \frac{1}{2}(2xy + 1) \\ \frac{1}{2}(1 - 2xy) & \frac{1}{2}\left(-x^2 - 3y^2 + 2\right) & 0 & 0 \\ \frac{1}{2}\left(3x^2 + y^2 - 2\right) & \frac{1}{2}(2xy + 1) & 0 & 0 \end{pmatrix}$$

Тогда собственные значения матрицы A будут выражаться в следующем виде

$$\lambda_1 = -x^2 - y^2 - \sqrt{4x^4 + 8x^2y^2 - 8x^2 + 4y^4 - 8y^2 + 5}, \qquad \lambda_2 = x^2 + y^2 - \sqrt{4x^4 + 8x^2y^2 - 8x^2 + 4y^4 - 8y^2 + 5}$$

$$\lambda_3 = -x^2 - y^2 + \sqrt{4x^4 + 8x^2y^2 - 8x^2 + 4y^4 - 8y^2 + 5}, \qquad \lambda_4 = x^2 + y^2 + \sqrt{4x^4 + 8x^2y^2 - 8x^2 + 4y^4 - 8y^2 + 5}.$$

Несложно заметить, что максимальным собственным значением является именно λ_4, которое и будем использовать в дальнейших расчётах. Так же вспомним, что $t_{k+1} - t_k = h$. Значит окончательные формулы для расчёта погрешности принимают вид

$$\delta_0 = 0, \qquad \delta_{k+1} = Err_k + \delta_k \cdot e^{\lambda}, \qquad \lambda = \int\limits_{t_k}^{t_{k+1}} \mu(s)ds = h \cdot \left(x^2 + y^2 + \sqrt{4x^4 + 8x^2y^2 - 8x^2 + 4y^4 - 8y^2 + 5}\right).$$

4 Результаты

По вышеописанному алгоритму была написана компьютерная программа на языке программирования C. По результатам её работы было получено множество точек, составляющих траекторию движения системы, описанной данными уравениями. Однако, опытным путём было выяснено, что для $\alpha = 1.0$. при стремлении $t \to 3.6524$ значения скоростей стремяться к бесконечности. Поэтому расчёты производились до предельного значения точности шага, который был выбран следующим образом $H_{DEFAULT} = 10^{-15}$ и расчёты происходили до соответствующего момента времени $t = 3.5$. После этого по описанному выше алгоритму были посчитаны величины погрешности, которые удовляют всем поставленым ранее ограничениям и указывают на то, что полученная программа работает корректно.

Так же для проверки данная задача была решена с помощью пакета *Wolfram Mathematica 9* (см. прилож. 6.1). Отчётливо видно, что при запуске *Wolfram* выдаёт следующую ошибку:

"At t = 3.652401607212685, step size is effectively zero; singularity or stiff system suspected".
Таким образом, аналагично нашей программе, был произведён анализ до момента времени $t = 3.5$.

Аналогично при $\alpha = 0.1$ критическим значением оказалось $t = 5$.

4.1 Визуализация полученных результатов

Для более наглядного представления полученных результатов была написана компьютерная программа на языке программирования *Python*, с помощью библиотеки *Matplotlib*, которая строит графики зависимости наших координат от времени, а так же друг от друга, а именно x(t), y(t) и y(x) (см. рис. 1). Соответствующие графики для $\alpha = 1.0$ представлены на рис. (2), (3), (4) и $\alpha = 0.1$ на (5), (6), (7)

4.2 Анализ полученных результатов.

Проведём анализ полученных результатов можно провести в три стадии.

1. Несложно заметить, что полученное графики полностью совпадают с теми, что полученны с помощью пакета *Wolfram Mathematica 9* (см. прилож. 6.1 и рис. (2), (3) и (4)).

2. Из-за того, что максимальное собственное значение матрицы

$$A = \frac{J + J^T}{2}$$

быстро достигает больших значений, то экспонента, а соответственно и велечина глобальной погрешности, растут с крайне высокой скоростью, поэтому через какое-то время они переходят через точную верхнюю гран множества допустим значений соответствующего типа. Это явление делает невозможным подсчёт глобального отклонения.

3. Из-за того, что значение скорости быстро убегает в бесконечность, количество шагов метода Рунге-Кутта так же слишком мало для корректного подсчёта локального отклонения.

```
import pandas as pd
import numpy as np
import pylab as pl
%pylab inline
read_df = pd.read_csv("data.csv",sep = ",")

t = read_df[[0]].values
x = read_df[[1]].values
y = read_df[[2]].values

pl.plot(t, x)
pl.axis([0, 30 , -9 , 9])
pl.xlabel("t")
pl.ylabel("x")
savefig('t-x.pdf')

pl.show()
```

Рис. 1. Код на языке *Python* для построения графика $x(t)$.

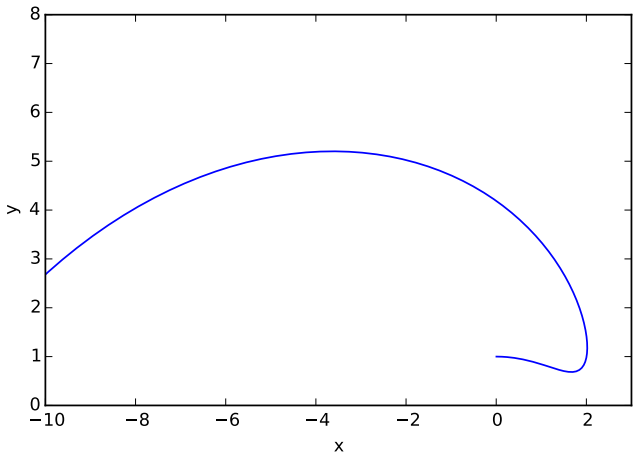


Рис. 2. Решение задачи 2.6 при $\alpha = 1.0$. График зависимости $y(x)$

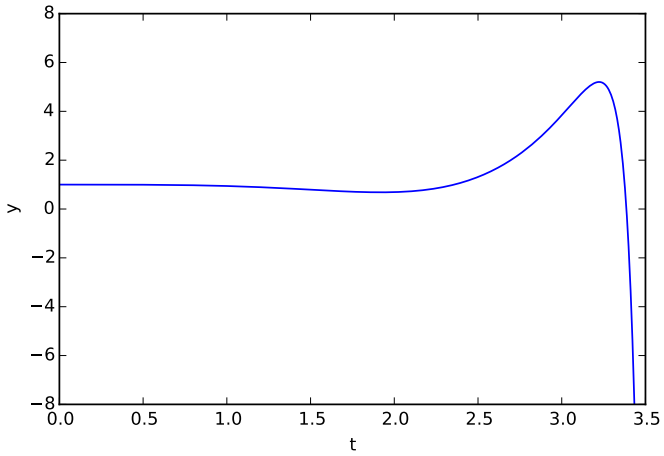


Рис. 3. Решение задачи 2.6 при $\alpha = 1.0$. График зависимости $y(t)$

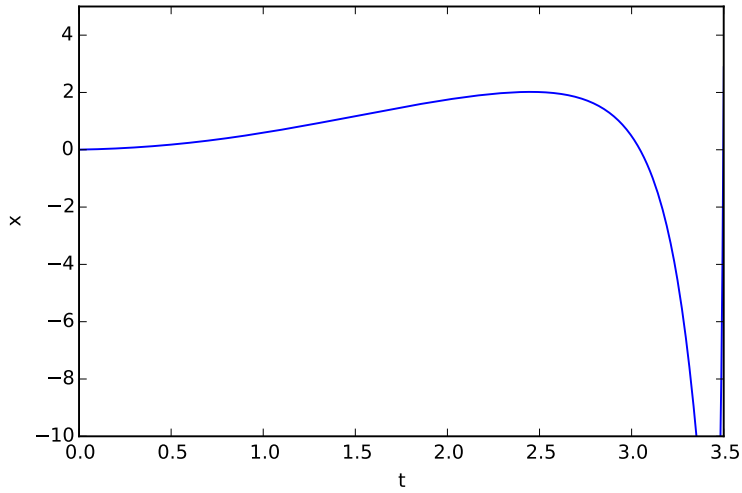


Рис. 4. Решение задачи 2.6 при $\alpha = 1.0$. График зависимости $x(t)$

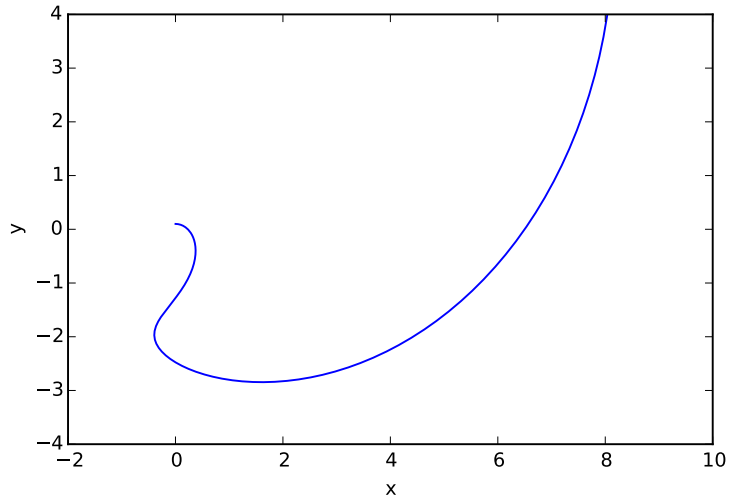


Рис. 5. Решение задачи 2.6 при $\alpha = 0.1$. График зависимости $y(x)$

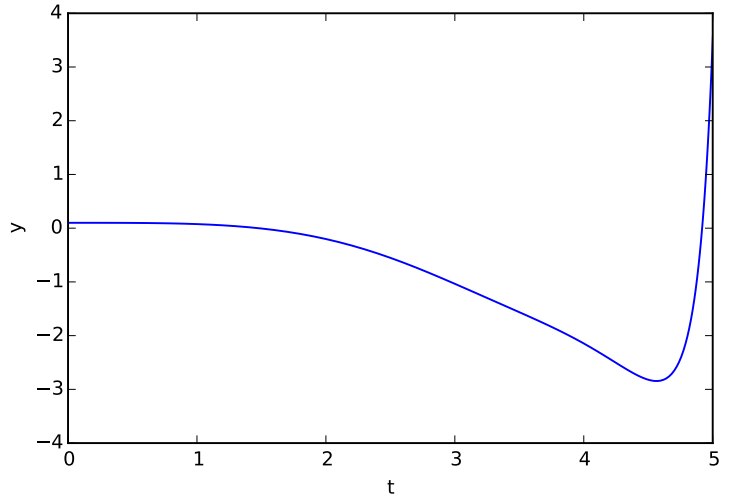


Рис. 6. Решение задачи 2.6 при $\alpha = 0.1$. График зависимости $y(t)$

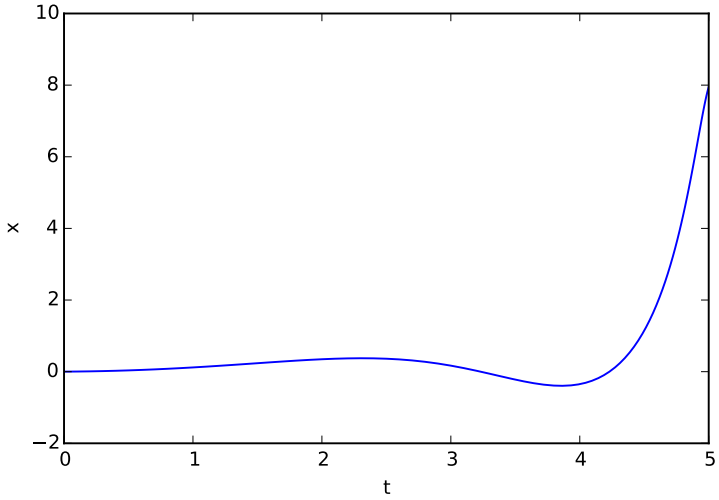


Рис. 7. Решение задачи 2.6 при $\alpha = 0.1$. График зависимости $x(t)$

5 Гармонический осциллятор

Дабы отбросить все сомнения в полученных результатах, проверим работоспособность программы на более простом случае с заранее известным решением, а именно на гармоническом осцилляторе

$$\begin{cases} \frac{dx}{dt} = y; \\ \frac{dy}{dt} = -x; \\ 0 < t < 30; \\ t = 0 : x = 0, y = 8. \end{cases}$$

Визуализация численного решения данной задачи с помощью нашей программы представлена в виде графиков на рис.(10), (9) и (8). Для удобства проверки дополнительно решим нашу задачу с помощью пакета *Wolfram Mathematica 9* (см. рис. 6.2). Сравнивая полученные результаты можно заметить, что полученные решения абсолютно идентичны, на основе чего можно сделать вывод о корректности работы программы. Однако, для большей достоверности проверим так же численные оценки наших погрешностей.

1. Для вычисления **глобальной погрешности** найдём матрицу Якоби нашей системы

$$J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \qquad A = \frac{J + J^T}{2} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\lambda_{1,2} = 0, \qquad \delta_0 = 0, \qquad \delta_{k+1} = Err_k + \delta_k.$$

Таким образом были получены следующие значения глобальной погрешности:

для точности погрешности -7-го порядка $\delta_k = 2.433796 \cdot 10^{-6}$

для точности погрешности -9-го порядка $\delta_k = 3.964618 \cdot 10^{-7}$

для точности погрешности -11-го порядка $\delta_k = 8.135620 \cdot 10^{-9}$

2. А оценка **локального отклонения на шаге** для каждой точки $t = \{50, 100, 150, 200\}$ получилась равна в районе 100 ± 2.

На основе выше сказанного можно сделать вывод, что полученная программа работает корректно.

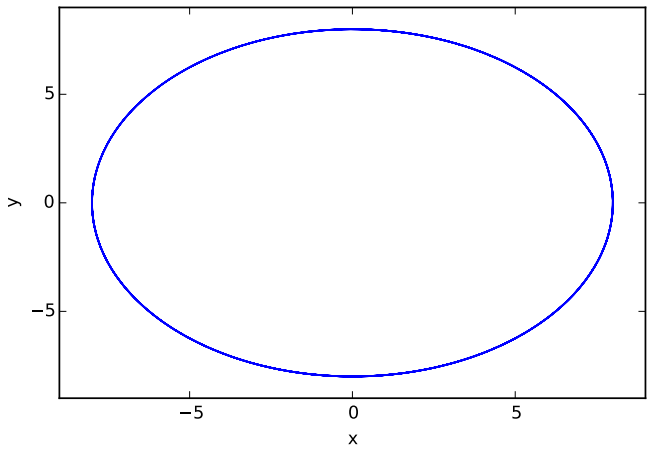


Рис. 8. Решение задачи о математическом осцилляторе. График зависимости $y(x)$

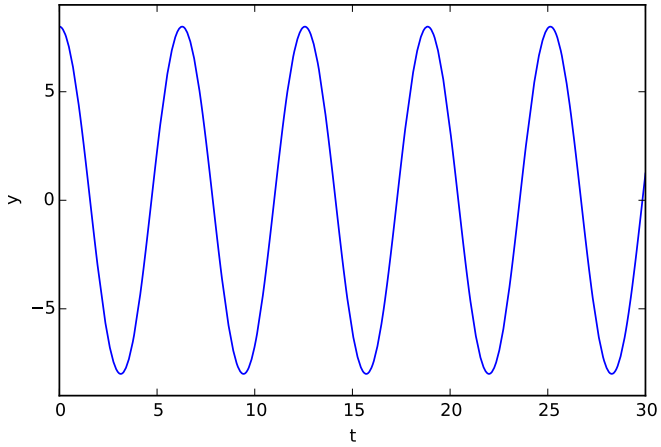


Рис. 9. Решение задачи о математическом осцилляторе. График зависимости $y(t)$

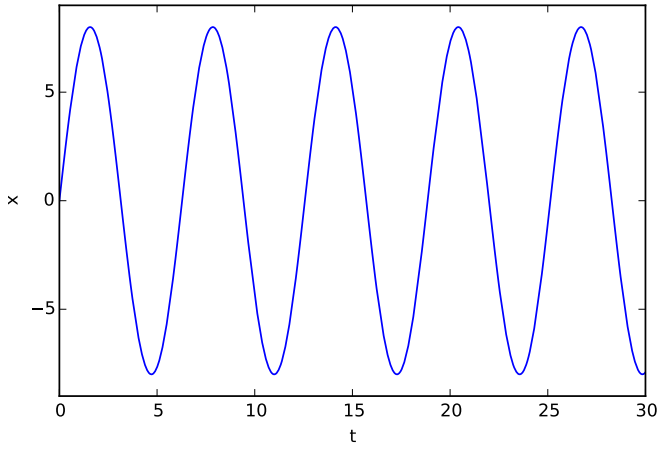


Рис. 10. Решение задачи о математическом осцилляторе. График зависимости $x(t)$

6 Приложения

6.1 Решение задачи 2.6 Wolfram Mathematica 9.

In[2]:=
```
NDSolve[{ x''[t] ==  y[t] * (2 - x[t]^2 - y[t]^2),
          y''[t] == -x[t] * (2 - x[t]^2 - y[t]^2),
          x[0] == 0, y[0] == 1., x'[0] == 0, y'[0] == 0},
         {x, y}, {t, 0, 30}]
```

NDSolve::ndsz :

At t == 3.652401607212685`, step size is effectively zero; singularity or stiff system suspected. ≫

Out[2]=
```
{{x → InterpolatingFunction[{{0., 3.6524}}, <>],
  y → InterpolatingFunction[{{0., 3.6524}}, <>]}}
```

```
solve = NDSolve[{ x''[t] ==  y[t] * (2 - x[t]^2 - y[t]^2),
                  y''[t] == -x[t] * (2 - x[t]^2 - y[t]^2),
                  x[0] == 0, y[0] == 1., x'[0] == 0, y'[0] == 0},
                 {x, y}, {t, 0, 3.5}]
ParametricPlot[Evaluate[{x[t], y[t]} /. solve], {t, 0, 3.5}]
ParametricPlot[Evaluate[{t, x[t]   } /. solve], {t, 0, 3.5}]
ParametricPlot[Evaluate[{t, y[t]   } /. solve], {t, 0, 3.5}]
```

Out[28]=
```
{{x → InterpolatingFunction[{{0., 3.5}}, <>],
  y → InterpolatingFunction[{{0., 3.5}}, <>]}}
```

График y (x)

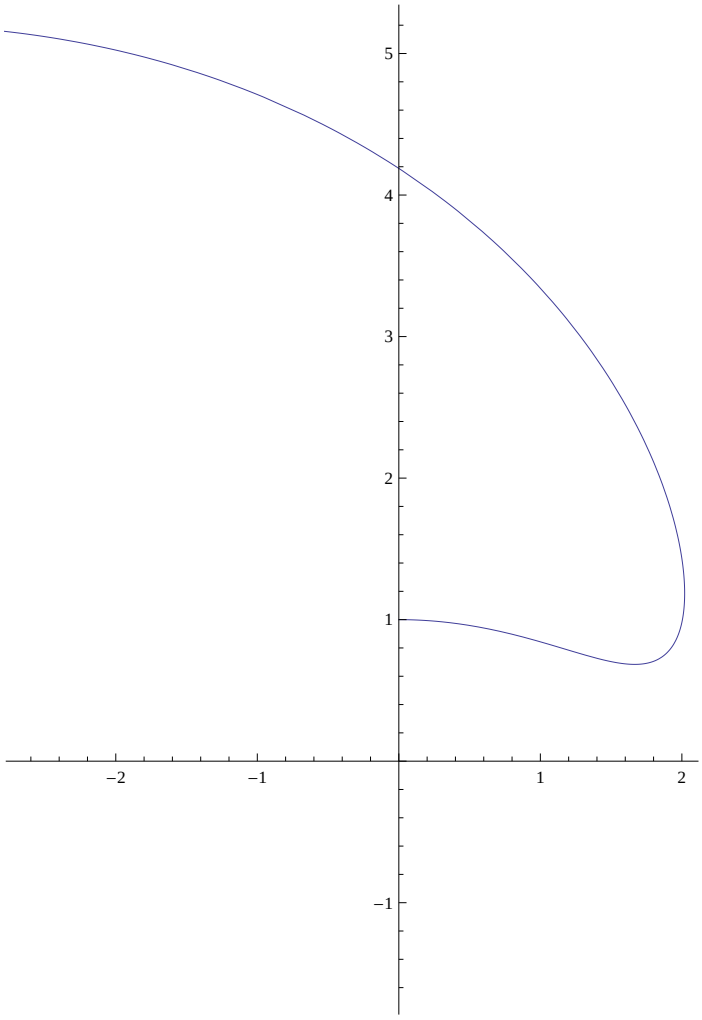


График x (t)

Out[30]=

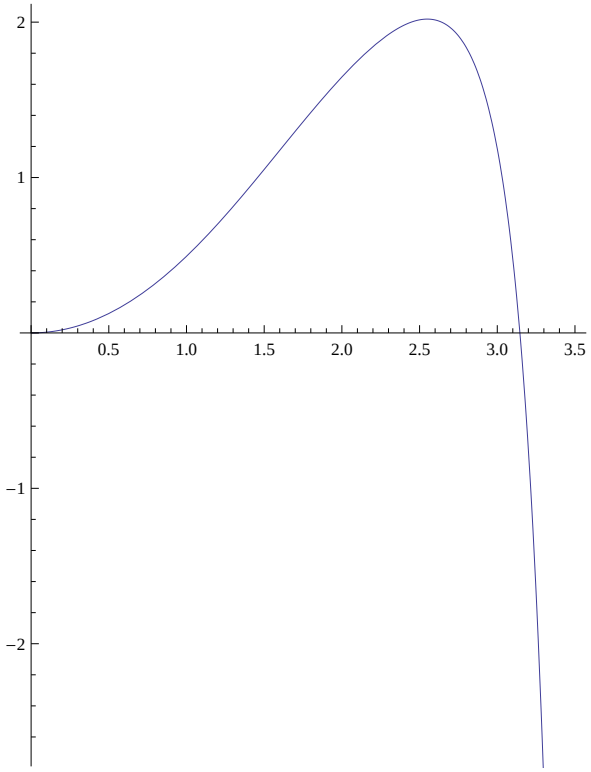
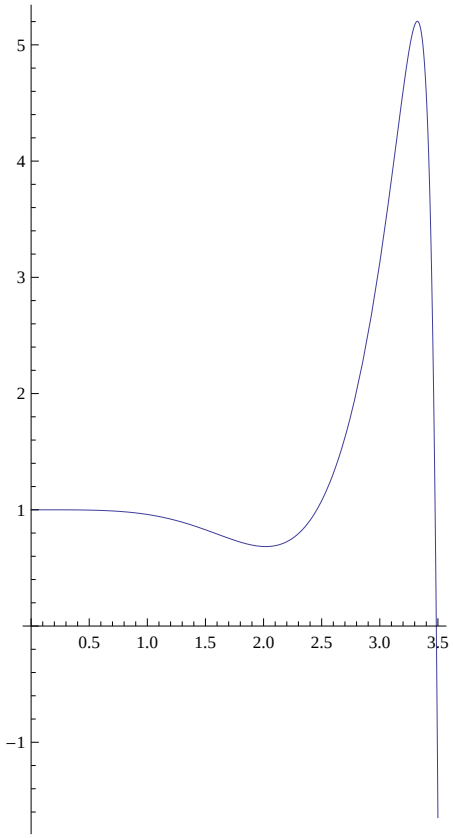


График y (t)

Out[31]=

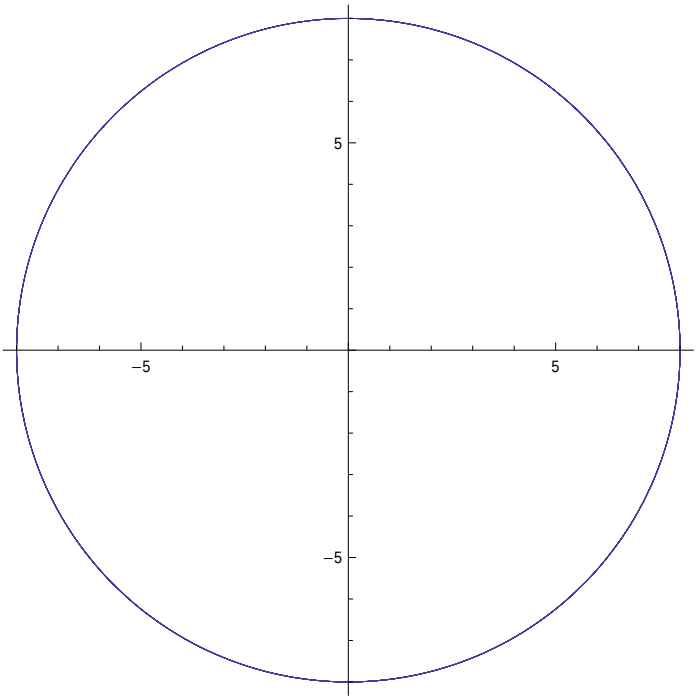


6.2 Решение задачи о математическом осцилляторе в Wolfram Mathematica 9.

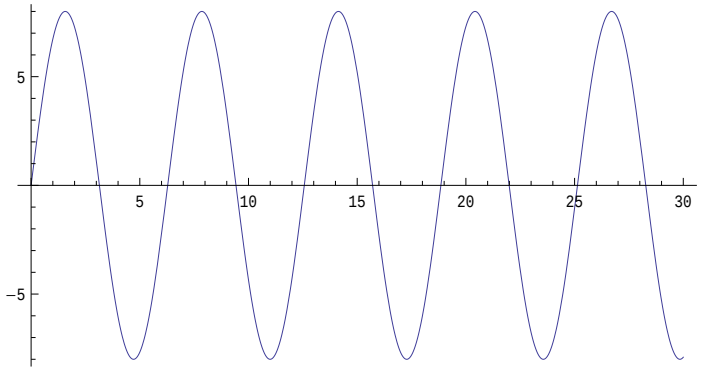
In[44]:=
```
solve = NDSolve[
  {x'[t] == y[t], y'[t] == -x[t], x[0] == 0, y[0] == 8}, {x, y}, {t, 0, 30}]
ParametricPlot[Evaluate[{x[t], y[t]} /. solve], {t, 0, 30}]
ParametricPlot[Evaluate[{t, x[t]   } /. solve], {t, 0, 30}]
ParametricPlot[Evaluate[{t, y[t]   } /. solve], {t, 0, 30}]
```

Out[44]=
```
{{x → InterpolatingFunction[{{0., 30.}}, <>],
  y → InterpolatingFunction[{{0., 30.}}, <>]}}
```

Out[45]=



Out[46]=



Out[47]=