Доля П.Г.
Харьковский Национальный Университет
механико – математический факультет
2015 г.

Mathematica для математиков.

Представлен параграф 4.5 «Примеры исследования обыкновенных дифференциальных уравнений» главы 4 «Решение дифференциальных уравнений в пакете Mathematica» нашего пособия «Mathematica для математиков».

Оглавление

4.5 Примеры исследования ОДУ

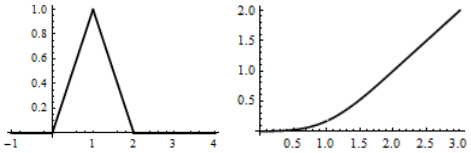
В этом параграфе мы приводим примеры решения прикладных задач из области физики, механики, электротехники и других областей науки, которые сводятся к решению ОДУ или их систем.

4.5.1 Движение материальной точки по прямой.

Дифференциальное уравнение движения имеет вид: $m\ddot{x} = f(t)$, где m – масса точки, $f(t)$ – внешняя сила.

Пример 1.1. Пусть внешняя сила на отрезке времени [*0,1*] задана в виде *f(t)=t*, на отрезке времени [*1,2*] равна *f(t)=2-t*, и равна 0 при *t>2*. Тогда решить задачу в *Mathematica* можно так (полагаем *m*=1).

```
f[t_] = Piecewise[{{0, t < 0}, {t, t < 1}, {2 - t, t < 2}}, 0];
Plot[f[t], {t, -1, 4}]
ds = DSolve[{x''[t] == f[t], x[0] == 0, x'[0] == 0}, x, t]
xs = x /. ds[[1]];
Plot[xs[t], {t, 0, 3}]
```



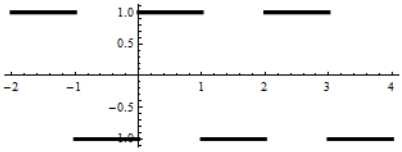
На левом рисунке приведен график внешней силы, а на правом – график функции *x*(*t*) – решения ОДУ.

Пример 1.2. Внешняя сила является периодической импульсивной функцией, которая на разных участках периода принимает только два значения ±1. Определим такую функцию (*Periodic Impuls Function = Pif*)

$$Pif(x,a,w) = 2\cdot\left(\left[\frac{x}{w}\right]-\left[\frac{x-a}{w}\right]\right)-1$$

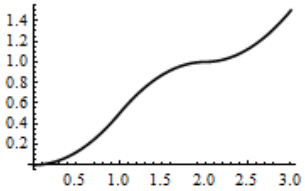
Это периодическая с периодом *w* функция. Для *a*<*w* на отрезке $0 \le x < a$ функция равна 1, на остальном куске периода $a \le x < w$ функция равна -1 (случай $a \ge w$ мы не рассматриваем). На следующем рисунке приведен график функции *Pif(x,1,2)*.

```
Pif[x_, a_, w_] = 2(Floor[x/w] - Floor[(x - a)/w]) - 1;
Plot[Pif[x, 1, 2], {x, -2, 4}, AspectRatio → Automatic]
```



Пусть внешняя сила имеет вид *f(t)=Pif(x,1,2)* и *m=1*. Тогда

```
nds = NDSolve[{x''[t] == Pif[t, 1, 2], x[0] == 0, x'[0] == 0}, x, {t, 0, 4}]
xs = x /. nds[[1]];
Plot[xs[t], {t, 0, 3}]
```



4.5.2 Движение упругого мяча

Упругий мячик имеет начальное положение и скорость. Сопротивление воздуха пренебрежимо мало, но энергия движения расходуется при отскоке мяча от земли. Пусть при отскоке от земли вертикальная скорость мячика составляет 90% от вертикальной скорости в момент падения.

Уравнение движения (без учета сопротивления воздуха) имеет вид

$$m\ddot{\mathbf{r}} = -m\mathbf{g},$$

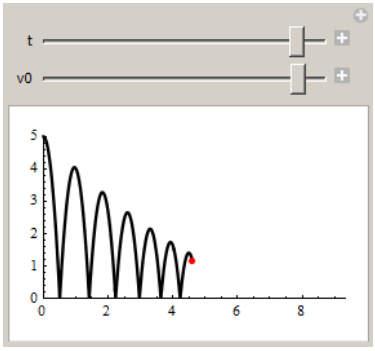
где $\mathbf{g}$ – вектор ускорения свободного падения, имеющий направление вертикально вниз. В покоординатной форме имеем $\ddot{x} = 0$, $\ddot{y} = -g$ и начальные условия $x(0) = x_0, y(0) = h, \dot{x}(0) = v_0^x$, $\dot{y}(0) = v_0^y$. Векторное уравнение распадается на два независимых скалярных уравнения.

$$\begin{cases} \ddot{x}(t) = 0 \\ \ddot{y}(t) = -g \end{cases},$$

первое из которых имеет решение $x(t) = x_0 + v_0^x \cdot t$. Это указывает на то, что в горизонтальном направлении движение происходит с постоянной скоростью v_0^x. В нашем примере положим $x_0 = 0$. Второе уравнение также интегрируется, если не учитывать скачкообразного изменения скорости в момент касания земли. Но мы хотим показать, как можно использовать функцию `WithEvent`, и будем решать уравнение численно. В момент отскока t_k вертикальная скорость $\dot{y}$ меняет знак, т.е. $\dot{y}_{\text{после отскока}}(t_k + \varepsilon) = -0.9 \cdot \dot{y}_{\text{до отскока}}(t_k - \varepsilon)$ и становится положительной.

Находим решение второго уравнения системы с учетом резкого изменения скорости мяча при отскоке. Затем строим параметрическое уравнение траектории в манипуляторе с параметрами времени движения и начальной скорости.

```
Manipulate[
  s = NDSolveValue[{y''[t] == -9.81, y[0] == 5, y'[0] == 0,
        WhenEvent[y[t] == 0, y'[t] -> -0.9 y'[t]]}, y, {t, 0, 10}];
  p1 = ParametricPlot[Evaluate[{x[t], s[t]}], {t, 0, tmax},
                                  PlotRange → {{0, 10 v0}, {0, 5}}];
  p2 = Graphics[{RGBColor[1, 0, 0], Disk[{x[tmax], s[tmax]}, 0.1]}];
  Show[p1, p2],
  {{tmax, 0.5, t}, 0.3, 10}, {{v0, 0.5, "v0"}, 0.1, 1},
  AutoAction → False, Initialization :> {x[t_] := v0 t;}]
```



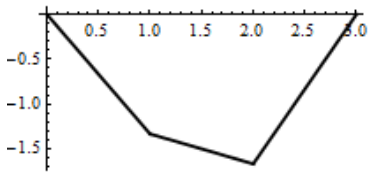
4.5.3 Равновесие струны под действием сосредоточенных сил.

Пусть струна закреплена в точках $x_0=0$ и $x_n=L$, а внешние сосредоточенные силы f_i приложены в точках x_i: $0 = x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n = L$. Уравнение равновесия струны (с малым отклонением) имеет вид: $u''_{xx} = -f(x)$, $f(x) = \dfrac{F(x)}{T}$, где T – натяжение струны, $F(x)$ – внешняя сила, рассчитанная на единицу длины, $u(x)$ – отклонение струны от положения равновесия. Функцию $f(x)$, создаваемую набором сосредоточенных сил во внутренних точках x_i отрезка $[0, L]$, можно представить следующим образом $f(x) = \sum_{i=1}^{n-1} f_i \cdot \delta(x - x_i)$, где $\delta(x - x_i)$ - функция Дирака, равная нулю везде, кроме точки x_i в которой она обращается в бесконечность так, что интеграл по отрезку, содержащему точку x_i, равен единице. В пакете *Mathematica* функция Дирака называется `DiracDelta[x]`. В примерах положим T=1.

Пример 3.1. Пусть $L = 3$, $x_1 = 1$, $x_2 = 2$, $f_1 = 1$, $f_2 = 2$. Тогда

```
L = 3;
f[x_] = DiracDelta[x - 1] + 2DiracDelta[x - 2];
ds = DSolve[{u''[x] == f[x], u[0] == 0, u[L] == 0}, u, x]
z[x_] = Simplify[(ds[[1, 1, 2, 2]])]
Plot[z[x], {x, 0, L}]
```

$$-\frac{4x}{3} + 2(-2 + x)\text{HeavisideTheta}[-2 + x] + (-1 + x)\text{HeavisideTheta}[-1 + x]$$



Мы изменили знак правой части специально, чтобы на графике прогиб струны был направлен вниз. Интересно отметить, что решение можно представить в более «приятном» виде: $u(x) = -\dfrac{5}{2} + \dfrac{x}{6} + \dfrac{1}{2}|x-1| + |x-2|$, который *Mathematica* не умеет получать с помощью функции `Simplify`. Для этого в полученном выражении для `z[x]` нужно сделать замену

```
(-k+x)HeavisideTheta[-k+x]->1/2(x-k-Abs[x-k]),
```

где k надо положить 1 и 2. Эта замена следует из формулы $x\,H(x)=\frac{1}{2}\left(x+\left|\,x\,\right|\right)$, где $H(x)$ – функция Хэвисайда (HeavisideTheta). Такая замена в более общем виде нам потребуется несколько раз, а именно $x^k\,H(x)=\frac{1}{2}\left(x^k+\left|\,x\,\right|^k\right)$ при нечетном k и $x^k\,H(x)=\frac{1}{2}\left(x^k+x^{k-1}\left|\,x\,\right|\right)$ при четном k. Напишем функцию, выполняющую такую замену

```
SimplifyHeavisideTheta = Function[{expr, xL, k, x},
   n = Length[xL];
   ch = Table[HeavisideTheta[-xL[[i]] + x], {i, n}];
   If[Mod[k, 2] == 0,
     cuh = Table[(-xL[[i]] + x)^k HeavisideTheta[-xL[[i]] + x] →
                 1/2((x - xL[[i]])^k + (x - xL[[i]])^(k-1) Abs[x - xL[[i]]]), {i, n}],
     cuh = Table[(-xL[[i]] + x)^k HeavisideTheta[-xL[[i]] + x] →
                 1/2((x - xL[[i]])^k + Abs[x - xL[[i]]]^k), {i, n}]
   ];
   cu = Simplify[Collect[expr, ch]];
   c2 = Simplify[(cu /. cuh)];
   cu2 = Simplify[Collect[c2, ch]];
   Simplify[(cu2 /. cuh)]
];
```

Эта функция упрощает выражение expr, содержащее элементы вида $(-i+x)^k\,HeavisideTheta[-i+x]$, xL – список целых значений i, входящих в заменяемые выражения, k - целая положительная степень k=1,2,...; x - имя переменной в выражении expr.

Тогда, используя эту функцию, имеем

```
SimplifyHeavisideTheta[z[x], {1, 2}, 1, x]//TraditionalForm
```

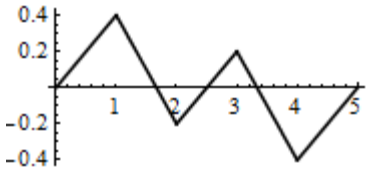
$\frac{1}{6}(6\,|\,x-2\,|\,+3\,|\,x-1\,|\,+x-15)$

Обратите внимание, что здесь решена краевая задача, а не задача Коши, как в предыдущем пункте.

Пример 3.2. Пусть L=5, сосредоточенные нагрузки $f_{list}=[1,-1,1,-1]$ приложены в точках $x_i=1,2,3,4$.

```
L = 5; xl = {1, 2, 3, 4}; fl = {1, -1, 1, -1};
f[x_] = Sum[fl[[i]] DiracDelta[x - xl[[i]]], {i, Length[xl]}];
nds = DSolve[{u''[x] == -f[x], u[0] == 0, u[L] == 0}, u, x];
z[x_] = (nds[[1, 1, 2, 2]]);
SimplifyHeavisideTheta[z[x], xl, 1, x]//TraditionalForm
Plot[z[x], {x, 0, L}, AspectRatio → 0.5]
```

$\frac{1}{10}(5\,|\,x-4\,|\,-5\,|\,x-3\,|\,+5\,|\,x-2\,|\,-5\,|\,x-1\,|\,+4x-10)$



Пример 3.3. Равновесие струны под действием кусочно – постоянной силы.

$$\boldsymbol{L = 3;}$$

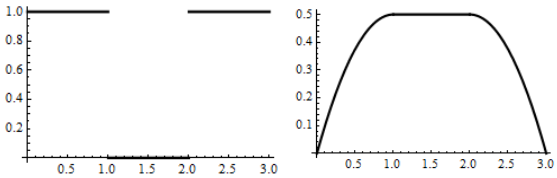
$$\boldsymbol{f[x\_] = \text{HeavisideTheta}[x] - \text{HeavisideTheta}[x-1] + \text{HeavisideTheta}[x-2];}$$

$$\boldsymbol{\text{ds} = \text{DSolve}[\{u''[x] == -f[x], u[0] == 0, u[L] == 0\}, u, x];}$$

$$\boldsymbol{\text{SimplifyHeavisideTheta}[\text{ds}[[1,1,2,2]], \{0,1,2\}, 2, x]//\text{TraditionalForm}}$$

$$\boldsymbol{\text{GraphicsRow}[\{\text{Plot}[f[x], \{x, 0, L\}], \text{Plot}[\text{nds}[[1,1,2,2]], \{x, 0, L\}]\}]}$$

$$\frac{1}{4}(-x\,|\,x\,| - (x-2)\,|\,x-2\,| + (x-1)\,|\,x-1\,| - x^2 + 6x - 3)$$



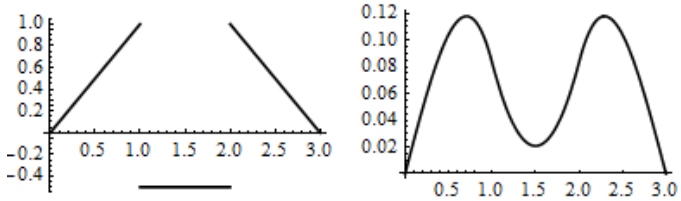
Слева показан график внешней силы, справа профиль струны.

Пример 3.4. Равновесие нити под действием кусочно - линейной внешней силы

$$\boldsymbol{L = 3; f[\text{x\_}] = \text{Piecewise}[\{\{x, x < 1\}, \{-1/2, x < 2\}\}, 3 - x];}$$

$$\boldsymbol{\text{ds} = \text{DSolve}[\{u''[x] == -f[x], u[0] == 0, u[L] == 0\}, u, x];}$$

$$\boldsymbol{\text{GraphicsRow}[\{\text{Plot}[f[x], \{x, 0, L\}], \text{Plot}[\text{ds}[[1,1,2,2]], \{x, 0, L\}]\}]}$$



Слева показан график внешней силы, справа профиль нити. Заметим, что когда исходная функция f[x] задана как кусочная (Piecewise), то и результат получается в виде Piecewise функции. Если ту же функцию f[x] задать с использованием функции Хэвисайда, то результат можно будет привести к выражению, содержащему функции абсолютного значения.

$$\boldsymbol{f[\text{x\_}] = x(1 - \text{HeavisideTheta}[x-1]) - (\text{HeavisideTheta}[x-1] - \text{HeavisideTheta}[x-2])/2 + (3-x)\text{HeavisideTheta}[x-2];}$$

$$\boldsymbol{\text{ds} = \text{DSolve}[\{u''[x] == -f[x], u[0] == 0, u[L] == 0\}, u, x];}$$

$$\boldsymbol{\text{ss} = \text{SimplifyHeavisideTheta}[\text{ds}[[1,1,2,2]], \{1,2\}, 2, x]}$$

$$\boldsymbol{\text{ss}//\text{TraditionalForm}}$$

$$\frac{1}{24}((x-2)(2x-13)|x-2| + (x-1)(2x+7)|x-1| - 9(2(x-3)x+5))$$

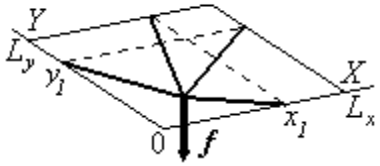
или

$$\boldsymbol{\text{Collect}[\text{Expand}[\text{ss}], \{\text{Abs}[-1+x], \text{Abs}[-2+x]\}]//\text{TraditionalForm}}$$

$$\left(\frac{x^2}{12} - \frac{17x}{24} + \frac{13}{12}\right)|x-2| + \left(\frac{x^2}{12} + \frac{5x}{24} - \frac{7}{24}\right)|x-1| - \frac{3x^2}{4} + \frac{9x}{4} - \frac{15}{8}$$

4.5.4 Статический прогиб сети нитей

Пример 4.1 Рассмотрим механическую систему, состоящую из двух пересекающихся под прямым углом струн/нитей L_x и L_y длиной L. Нити в точке пересечения (x_1, y_1) соединены и в этой точке приложена сосредоточенная сила f_0. Под действием силы f_0 обе нити примут вид ломаной (см. следующий рисунок)



Требуется составить уравнения этих ломаных.

Очевидно, что действие силы f_0 частично уравновешивается струной L_x и частично струной L_y. Т.е. на каждую струну действует своя сосредоточенная сила. Это значит, что на нить L_x действует только часть силы f_0. Обозначим ее через a. Часть силы f_0, действующую на нить L_y, обозначим через b. Очевидно, что $a+b=f_0$. Если знать силы a и b, то можно, решив соответствующее ОДУ, определить прогиб каждой из струн.

Имея ввиду впоследствии рассмотреть сеть нитей, т.е. множество нитей с прогибами $u_1(x),\ldots$ и $v_1(y),\ldots$ обозначим функцию прогибов нити L_x через $u[1][x]$, а нити L_y – через $v[1][y]$, а параметры через a[1] и b[1]. Составим и решим ОДУ прогиба первой и второй нитей при $x_1=1,\ y_1=2$

```
L = 3; f0 = 3;
dsX = DSolve[{u[1]''[x] == -a[1]DiracDelta[x - 1],
              u[1][0] == 0, u[1][L] == 0}, {u[1][x]}, x];
uu[x_] = SimplifyHeavisideTheta[dsX[[1, 1, 2]], {1}, 1, x]
dsY = DSolve[{v[1]''[y] == -b[1]DiracDelta[y - 2],
              v[1][0] == 0, v[1][L] == 0}, {v[1][y]}, y];
vv[y_] = SimplifyHeavisideTheta[dsY[[1, 1, 2]], {2}, 1, y]
```

$$\frac{1}{6}a[1](3+x-3\text{Abs}[-1+x])$$

$$-\frac{1}{6}(-6+y+3\text{Abs}[-2+y])b[1]$$

В решении присутствуют неопределенные пока параметры $a[1]$ и $b[1]$. Но у нас есть еще два условия: прогиб нитей в точке пересечения одинаков, и сумма этих параметров равна приложенной в узле (x_1, y_1) силе f0. Составляем алгебраическую систему уравнений и решаем ее

```
ss = Solve[{uu[1] == vv[2], a[1] + b[1] == f0}, {a[1], b[1]}]
```

$$\{\{a[1]\to\frac{3}{2}, b[1]\to\frac{3}{2}\}\}$$

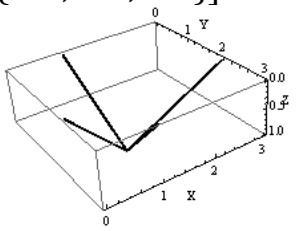
Зная параметры $a[1]$ и $b[1]$, построим функции прогибов нитей и нарисуем их графики

```
U1[x_] = uu[x]/.ss[[1]]
V1[y_] = vv[y]/.ss[[1]]
```

$\frac{1}{4}(3 + x - 3\text{Abs}[-1 + x])$

$\frac{1}{4}(6 - y - 3\text{Abs}[-2 + y])$

```
su1 = ParametricPlot3D[{t, 2, U1[t]}, {t, 0, L}];
sv1 = ParametricPlot3D[{1, t, V1[t]}, {t, 0, L}];
Show[su1, sv1, AxesLabel → {"X", "Y", "Z"}]
```



Пример 4.2 Рассмотрим сеть, составленную из двух пар пересекающихся нитей/струн одинаковой длины и одинакового натяжения, имеющие симметричное расположение в плоскости гипотетической квадратной мембраны размера L. Пусть струны в точках пересечения соединены и в узлах приложены одинаковые вертикальные сосредоточенные силы *f*. В каждом узле (i, j) такая сила уравновешивается частично натяжением нити одного семейства и частично натяжением нити второго семейства. Можно сказать, что на эти нити действуют по отдельности сосредоточенные силы, сумма которых равна внешней силе приложенной в этом узле. Пусть уравнение ломаных первого семейства будут $u_1(x)$ и $u_2(x)$, а уравнения ломаных второго семейства $v_1(y)$ и $v_2(y)$. Обозначим узлы сети через (i, j), где $i, j=1,2$, сосредоточенные силы, приложенные к нитям первого семейства, через a_{ij}, и сосредоточенные силы, приложенные к нитям второго семейства – через b_{ij}. Очевидно, что $a_{ij}+b_{ij}=f$ для любых i и j. Рассматривая a_{ij} и b_{ij} как параметры, составим и решим соответствующие ОДУ прогиба нитей

```
Remove[a, b, u, v, UU, VV, U, V]
L = 3; XL = Table[i, {i, L - 1}]; YL = Table[j, {j, L - 1}];
m = Length[XL]; n = Length[YL];
f = Table[1, {i, m}, {j, n}];
dsX = DSolve[
        Flatten[Table[
            {u[j]''[x] == -Sum[a[j, k]DiracDelta[x - XL[[k]]], {k, m}],
                    u[j][0] == 0, u[j][L] == 0}, {j, n}]],
          Table[u[j][x], {j, n}], x];
sx = SimplifyHeavisideTheta[FullSimplify[dsX[[1]]], XL, 1, x];
Table[UU[j][x_] = sx[[j, 2]], {j, n}];
dsY = DSolve[Flatten[Table[
            {v[i]''[y] == -Sum[b[i, k]DiracDelta[y - XL[[k]]], {k, n}],
                    v[i][0] == 0, v[i][L] == 0}, {i, m}]],
            Table[v[i][y], {i, m}], y];
sy = SimplifyHeavisideTheta[FullSimplify[dsY[[1]]], YL, 1, y];
Table[VV[i][y_] = sy[[i, 2]], {i, m}];
```

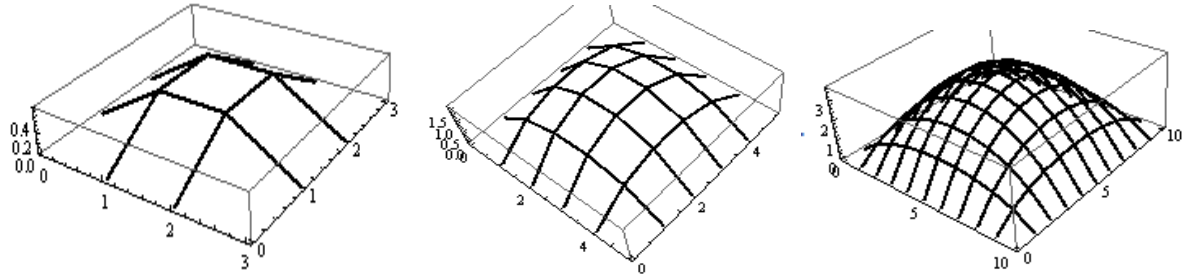
Здесь функции UU[j][x] и VV[i][y] содержат неопределенные пока константы a[j,k] и b[i,k]. Составим систему линейных алгебраических уравнений относительно этих величин и решим ее.

```
tt = Join[
      Flatten[Table[UU[j][XL[[i]]] == VV[i][YL[[j]]], {i, m}, {j, n}]],
      Flatten[Table[a[i, j] + b[j, i] == f[[i, j]], {i, m}, {j, n}]]];
tv = Flatten[Join[Table[a[i, j], {i, m}, {j, n}], Table[b[i, j], {i, m}, {j, n}]]];
ss = Solve[tt, tv];
```

$$\{\{a[1,1] \to \frac{1}{2}, a[1,2] \to \frac{1}{2}, a[2,1] \to \frac{1}{2}, a[2,2] \to \frac{1}{2},$$
$$b[1,1] \to \frac{1}{2}, b[1,2] \to \frac{1}{2}, b[2,1] \to \frac{1}{2}, b[2,2] \to \frac{1}{2}\}\}$$

Зная параметры a[j,k] и b[i,k], выполним их подстановку в функции UU[j][x] и VV[i][y], и построим кривые прогибов нитей U[j][x] и V[i][y].

```
Table[U[j][x_] = UU[j][x] /. ss[[1]], {j, n}];
Table[V[i][y_] = VV[i][y] /. ss[[1]], {i, m}];
p = Table[ParametricPlot3D[{t, YL[[j]], U[j][t]}, {t, 0, L},
                  DisplayFunction → Identity], {j, n}];
q = Table[ParametricPlot3D[{XL[[i]], t, V[i][t]}, {t, 0, L},
                  DisplayFunction → Identity], {i, m}];
Show[p, q]
```



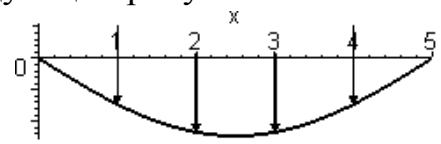
На предыдущем рисунке слева показана сеть, составленная из двух пар ортогонально пересекающихся упругих нитей. Если положить в предыдущем коде (сценарии) L=5, то получится сеть, показанная на предыдущем рисунке в середине. Положив L=10, мы получим деформированную сеть, показанную справа.

4.5.5 Прогиб балки

Уравнение изгибающего момента свободно опертой балки имеет тот же вид, что и уравнение равновесия струны $\frac{d^2M}{dx^2} = -q$, где $q(x)$ внешняя сила, а $M(x)$ изгибающий момент в балке, x – координата, направленная вдоль оси балки. Это значит, если балка находится под действием сосредоточенных или кусочных внешних сил, то мы можем применить прежний код для определения изгибающего момента $M(x)$. Для определения прогиба балки нужно решать уравнение $E\,J\,\frac{d^2u(x)}{d\,x^2} = -M(x)$, $J(x)$ – момент инерции поперечного сечения балки, E – модуль упругости материала балки, $u(x)$ – отклонение нейтральной

оси балки от положения равновесия. Также можно сразу решать ОДУ относительно функции прогиба $\frac{d^2}{dx^2}\left(EJ\frac{d^2u}{dx^2}\right)=q(x)$. Для свободно опертой на краях балки граничные условия должны иметь вид `u[0]=0,u[L]=0, u''[0]=0, u''[L]=0`. При ином закреплении граничные условия будут другими.

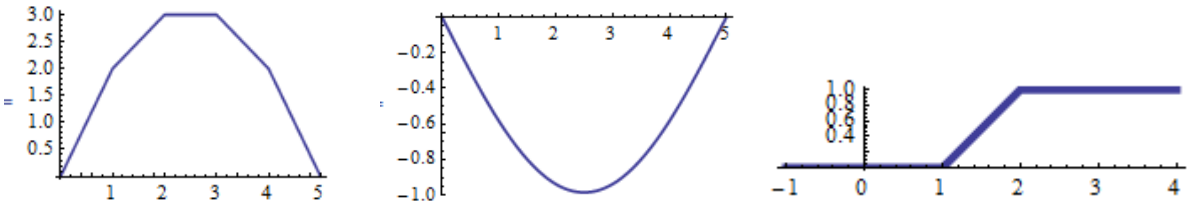
Пример 5.1 Пусть балка свободно оперта по краям и находится под действием четырех одинаковых сосредоточенных нагрузок. Нагрузки приложены на одинаковом расстоянии друг от друга и от концов балки. Решим модельную задачу, в которой положим L=5 (длина балки), E=8, J=1, h=1(высота балки). Единичные сосредоточенные нагрузки приложены в точках $x_i=1,2,3,4$. Схема нагрузок показана на следующем рисунке.



Найдем изгибающий момент $M(x)$ балки. Его график показан на следующем рисунке слева.

```
Remove[M, f, u, x, U, W]; L = 5;
f[x_] = Sum[DiracDelta[x - i], {i, 4}];
sM = DSolve[{M''[x] == -f[x], M[0] == 0, M[L] == 0}, M, x];
M = sM[[1, 1, 2]];
SimplifyHeavisideTheta[M[x], {1, 2, 3, 4}, 1, x]//TraditionalForm
Plot[M[x], {x, 0, L}, PlotStyle → Thickness[0.01]]
```

$$\frac{1}{2}\left(10-\left|x-1\right|-\left|x-2\right|-\left|x-3\right|-\left|x-4\right|\right)$$



Находим форму нейтральной оси балки.

```
Eu = 8; J = 1;
sU = DSolve[{Eu · J · u''[x] == M[x], u[0] == 0, u[L] == 0}, u, x];
W = sU[[1, 1, 2]];
ss = SimplifyHeavisideTheta[W[x], {1, 2, 3, 4}, 3, x];
U[x_] = Simplify[Expand[ss]]
U[x]//TraditionalForm
Plot[U[x], {x, 0, L}, PlotStyle → Thickness[0.01]]
```

$$\frac{1}{96}\left(100-150x+30x^2-\left|x-1\right|^3-\left|x-2\right|^3-\left|x-3\right|^3-\left|x-4\right|^3\right)$$

График функции $U(x)$ показан на предыдущем рисунке в середине.

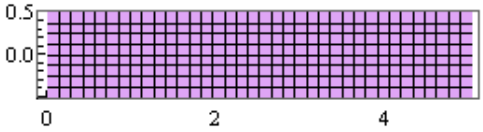
Целью следующего блока кода является графическое представление продольных напряжений в балке. Вначале напишем код, который рисует прямоугольную балку. Он использует вспомогательную функцию

```
P[x_, a_, w_] = (w + Abs[x − a] − Abs[x − a − w])/2;
Plot[P[x, 1, 1], {x, −1, 4}, AspectRatio → Automatic]
```

График функции $P(x,a,w)$ при $a=1, w=1$ показан на предыдущем рисунке справа.

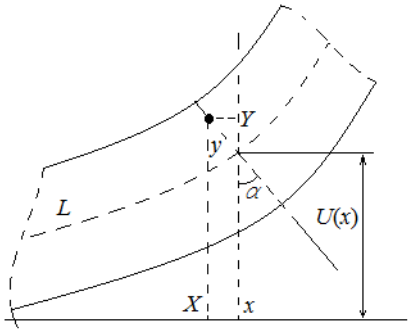
Теперь составляем параметрическое уравнение области прямоугольника, представляющего балку (с методикой составления параметрических уравнений ограниченных областей вы можете познакомиться по пособиям автора, представленным в разделах спецкурса «Аналитические методы геометрического моделирования»). Для графического представления области балки мы используем параметрическое уравнение ее поверхности в пространстве с третьей координатой *Z*=0.

```
h = 1; hh = h/2; (* полувысотабалки *)
x[u_, v_] = P[u, 0, L];
y[u_, v_] = −hh + P[v, −hh, 2hh];
ParametricPlot3D[{x[u, v], y[u, v], 0}, {u, 0, L}, {v, −hh, hh},
                 ViewPoint → {0, 0, Infinity}, Mesh → {35, 7}]
```



Используя уравнения плоской области $x(u,v)$, $y(u,v)$ недеформированной балки и уравнение $U(x)$ ее нейтральной оси, можно написать параметрическое уравнение области деформированной балки.

В соответствии с гипотезой плоских сечений, все поперечные сечения, которые были перпендикулярны нейтрально оси балки до деформирования, остаются перпендикулярными нейтральной оси балки после ее деформирования.



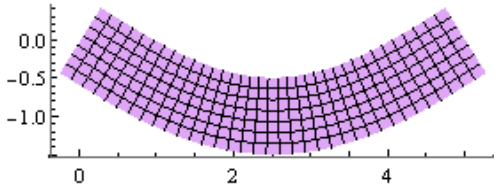
Точки, имевшие до деформирования координаты (x,y), переходят в точки с координатами (*X*,*Y*), где $X = x - y\cos\alpha$, $Y = U(x) + y\cos\alpha$ (см. рисунок). Но $tg\,\alpha = U'(x)$ и, поэтому, $X = x - \dfrac{U'_x(x)\,y}{\sqrt{1+(U'_x(x))^2}}$ и $Y = U(x) + \dfrac{y}{\sqrt{1+(U'_x(x))^2}}$. Тогда функции $X(u,v), Y(u,v)$, представляющие параметрические уравнения области изогнутой балки будут иметь вид

$$X(u,v) = x(u,v) - \frac{U'_x(x(u,v))y(u,v)}{\sqrt{1+(U'_x(x(u,v)))^2}}$$

$$Y(u,v) = U(x(u,v)) + \frac{y(u,v)}{\sqrt{1+(U'_x(x(u,v)))^2}}$$

Строим область деформированной балки. Для этого вычисляем производную $U_1(x) = U'_x(x)$ и создаем вспомогательную функцию $\sqrt{1+(U'_x(x))^2}$.

```
U1[x_] = Simplify[U'[x]/.Table[Abs[-i + x]^2 Abs'[-i + x] →
                    (-i + x) Abs[-i + x], {i, 4}], x ∈ Reals]
Usq[x_] = √(1 + U1[x]^2);
X[u_, v_] = x[u, v] - U1[x[u, v]] y[u, v] / Usq[x[u, v]];
Y[u_, v_] = U[x[u, v]] + y[u, v] / Usq[x[u, v]];
ParametricPlot3D[{X[u, v], Y[u, v], 0}, {u, 0, L}, {v, -hh, hh},
                 ViewPoint → {0, 0, Infinity}, Mesh → {34, 6}]
```



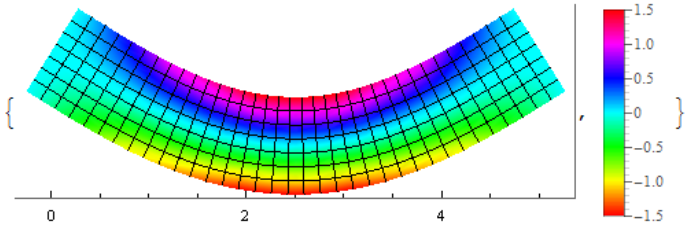
Следующим шагом будет показать в цвете продольные напряжения σ_x в балке. Если поперечное сечение балки прямоугольное, то σ_x вычисляются по формуле $\sigma_x(x,y) = \frac{M(x)\cdot y}{J}$, где $M(x)$ – момент инерции в сечении x балки, y – вертикальное смещение точки балки относительно нейтральной оси. Чтобы использовать эту функцию в качестве функции цвета, ее нужно привести к диапазону [0, 1]. Для этого нужно знать минимальные и максимальные значения напряжения σ_x. В нашей задаче они, очевидно, равны напряжениям в срединном сечении балки в верхней и нижней точках $\sigma_{x\,\max/\min} = \pm\frac{M(L/2)\cdot(h/2)}{J}$, где h – высота балки.

Создаем функцию продольных напряжений tensionX и нормированную в диапазоне от 0 до1 функцию beamColor.

```
tensionX[u_, v_] = M[x[u, v]] · y[u, v] / J;
tensmin = -M[L/2] hh / J;
tensmax = M[L/2] hh / J;
beamColor[u_, v_] = (tensionX[u, v] - tensmin) / (tensmax - tensmin);
```

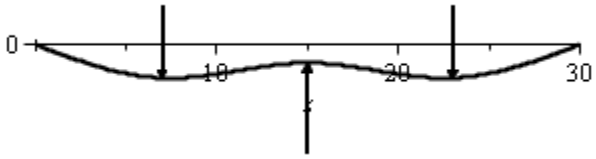
Рисуем область изогнутой балки, раскрашенную в соответствии с продольными напряжениями. Рядом рисуем палитру напряжений.

```
{ParametricPlot3D[{X[u, v], Y[u, v], 0}, {u, 0, L}, {v, -hh, hh},
    ViewPoint → {0, 0, Infinity}, Mesh → {34, 6},
    ColorFunction → (Hue[beamColor[#4, #5]]&),
    ColorFunctionScaling → False, Axes → {True, False, False}],
 BarLegend[{Hue, {-1.5, 1.5}}, LegendMarkerSize → 150]}
```



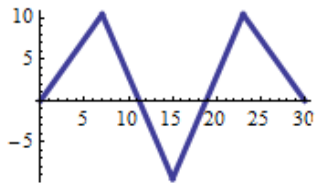
□

Пример 5.2 Пусть балка свободно оперта по краям и находится под действием трех сосредоточенных нагрузок. Схема приложения нагрузок показана на следующем рисунке.



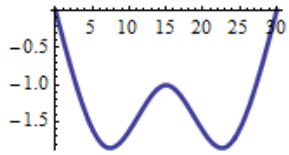
Решим задачу, в которой положим L=30, E=2, J=54, h=6. Сосредоточенные нагрузки $f_i = 4, -5, 4$ приложены в точках $x_i = 7, 15, 23$. Приведем только строки кода, которые изменены по сравнению с предыдущей задачей

```
Remove[M, f, u, x, U, W]; L = 30;
f[x_] = 4DiracDelta[x - 7] - 5DiracDelta[x - 15] + 4DiracDelta[x - 23]
...
SimplifyHeavisideTheta[M[x], {7, 15, 23}, 1, x]//TraditionalForm
...
```



$$\mathrm{M}[x] = \frac{1}{2}(-4 \mid x-23 \mid +5 \mid x-15 \mid -4 \mid x-7 \mid +45)$$

```
Eu = 2; J = 54;
...
ss = SimplifyHeavisideTheta[W[x], {7, 15, 23}, 3, x];
...
```



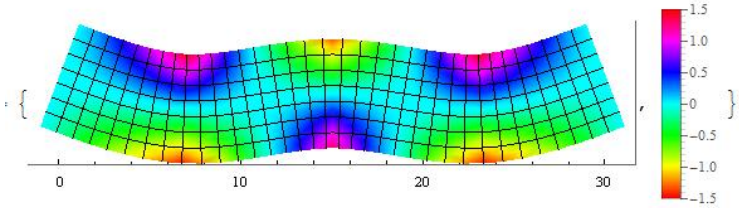
$$\frac{-4|x-23|^3 + 5|x-15|^3 - 4|x-7|^3 + 135x^2 - 4050x + 33165}{1296}$$

```
h = 6; hh = h/2;
...
U1[x_] = Simplify[U'[x]/.{Abs[-7 + x]^2 Abs'[-7 + x] → (-7 + x)Abs[-7 + x],
        Abs[-15 + x]^2 Abs'[-15 + x] → (-15 + x)Abs[-15 + x],
        Abs[-23 + x]^2 Abs'[-23 + x] → (-23 + x)Abs[-23 + x]}, x ∈ Reals]
...
```

$$\mathbf{tensmin} = -\frac{\boldsymbol{M}[7]\mathbf{hh}}{\boldsymbol{J}};$$

$$\mathbf{tensmax} = \frac{\boldsymbol{M}[7]\mathbf{hh}}{\boldsymbol{J}};$$

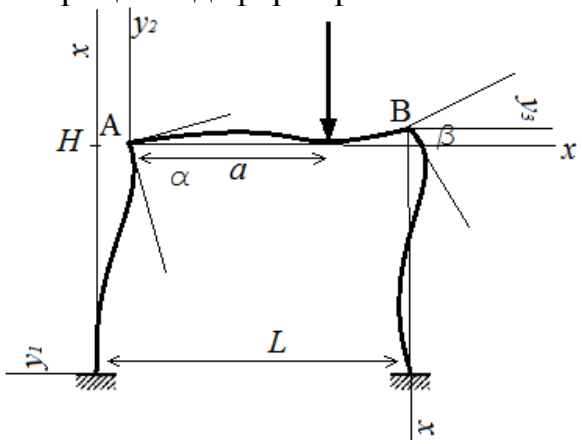
…



Для экономии места мы также не привели промежуточные рисунки, которые генерирует наш код.

4.5.6 Расчет рамы

Жесткой рамой называется такая стержневая система у которой все узловые соединения являются жесткими. Жесткий узел характеризуется тем, что угол между осями стержней его образующих не изменяется при действии нагрузки. Такой пример показан на следующем рисунке – углы α и β между касательными к осям горизонтальной и вертикальным балкам остается неизменным (прямым) в процессе деформирования.

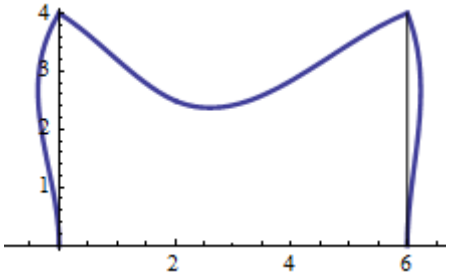


Когда рама деформируется, то обе касательные, проведенные в узле А, поворачиваются на одинаковый угол, иначе величина угла α изменилась бы. Тоже самое можно сказать и о касательных в точках В. С математической точки зрения расчет рамы сводится к решению системы ОДУ, описывающих деформацию стержней, задания граничных условий закрепления и сопряжения.

Рассмотрим раму, показанную на предыдущем рисунке. Размеры таковы: L=6, H=4, a=2. Положим также, что вертикальные и горизонтальные смещения узлов А и В равны нулю (они малы и ими можно пренебречь). Свяжем с каждым стержнем свою систему координат $(x, y_i)(i = 1,2,3)$, направления осей которых показаны на рисунке. Прогиб левого вертикального стержня обозначим через $y_1(x)(0 \le x \le H)$, прогиб верхнего горизонтального обозначим через $y_2(x)(0 \le x \le L)$, прогиб правого вертикального стержня – через $y_3(x)(0 \le x \le H)$. Условия неизменности углов (жесткие узлы А и В) будут выполнены, если в этих точках совпадают производные к соответствующим

прогибам. Т.е. эти условия имеют вид $y'_1(H)=y'_2(0), y'_2(L)=y'_3(0)$. Также в этих узлах одинаковые изгибающие моменты сопрягаемых стержней $E_1J_1\frac{d^2y_1(H)}{dx^2}=E_2J_2\frac{d^2y_2(0)}{dx^2}$ и $E_2J_2\frac{d^2y_2(L)}{d\,x^2}=E_3J_3\frac{d^2y_3(0)}{dx^2}$. Условия жесткого защемления опор дают граничные условия вида $y_1(0)=0, y'_1(0)=0$, $y_3(H)=0, y'_3(H)=0$. Тогда для решения задачи имеем следующий код

```
H = 4; L = 6; E1 = 1; J1 = 1; E2 = 1; J2 = 1; E3 = 1; J3 = 1;
ss = DSolve[{
      E1J1u1''''[x] == 0,
      E2J2u2''''[x] == -DiracDelta[x - 2],
      E3J3u3''''[x] == 0,
      u1[0] == 0, u1[H] == 0, u1'[0] == 0, E1J1u1''[H] == E2J2u2''[0],
      u1'[H] == u2'[0], u2[0] == 0, u2[L] == 0, u3[0] == 0,
      E2J2u2''[L] == E3J3u3''[0], u2'[L] == u3'[0], u3[H] == 0,
      u3'[H] == 0}, {u1, u2, u3}, x]
y1 = u1/.ss[[1]];
y2 = u2/.ss[[1]];
y3 = u3/.ss[[1]];
p1 = ParametricPlot[{-y1[t], t}, {t, 0, H}];
p2 = ParametricPlot[{t, H + y2[t]}, {t, 0, L}];
p3 = ParametricPlot[{L + y3[t], H - t}, {t, 0, H}];
p4 = Graphics[Line[{{L, 0}, {L, H}}]];
Show[{p1, p2, p3, p4}, PlotRange → {{-1, L + 1}, {-1, H + 1}}]
```



Для представления формы деформированной рамы, мы записали параметрические уравнения кривых(осей стержней) в глобальной системе координат. Выражения для прогибов в локальных координатах имеют вид

```
y1[x]
SimplifyHeavisideTheta[y2[x], {2}, 3, x] //TraditionalForm
y3[x]
```

$$-\frac{11}{288}(-4x^2+x^3)$$

$$\frac{1}{324}(-27\,|x-2|^3+11x^3+63x^2-522x+216)$$

$$\frac{7}{288}(16x-8x^2+x^3)$$

4.5.7 Поперечные колебания струны с грузами

Пример 7.1 *Поперечные колебания струны с двумя грузами.* Имеется 2 груза массы *M*, расположенные на струне длиной $L = 3a$ в точках $x=a$ и $x=2a$. Отрезки струны между грузами одинаковы, невесомы и подчиняются закону Гука. Натяжение в равновесии равно T. Обозначим вертикальное отклонение грузов от положения равновесия на оси струны через $y_1(t)$ и $y_2(t)$. Ограничимся рассмотрением малых колебаний. Уравнения движения грузов имеют вид

$$M\frac{d^2y_1}{dt^2} = T\left(\frac{y_2 - y_1}{a}\right) - T\left(\frac{y_1 - y_0}{a}\right)$$

$$M\frac{d^2y_2}{dt^2} = T\left(\frac{y_3 - y_2}{a}\right) - T\left(\frac{y_2 - y_1}{a}\right)$$

где y_0, y_3 - смещения левого и правого концов закрепленной струны, т.е. нули. Положим a=1, M=1, T=1. Тогда получим систему

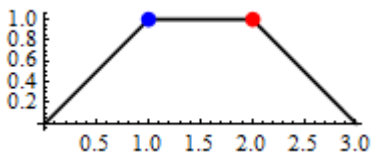
$$\begin{cases} \dfrac{d^2y_1}{dt^2} = y_2 - 2y_1 \\ \dfrac{d^2y_2}{dt^2} = -2y_2 + y_1 \end{cases}$$

Рассмотрим колебание системы без начальной скорости. Для этого решим систему ОДУ относительно функций $y_1(t), y_2(t)$ с единичными начальными смещениями грузов и нулевыми начальными скоростями.

```
Remove[y1, y2]
sys = {y1''[t] == y2[t] - 2y1[t], y2''[t] == -2y2[t] + y1[t],
       y1[0] == 1, y2[0] == 1, y1'[0] == 0, y2'[0] == 0}
ds = DSolve[sys, {y1, y2}, t]
y1 = ds[[1, 1, 2]]
y2 = ds[[1, 2, 2]]
```

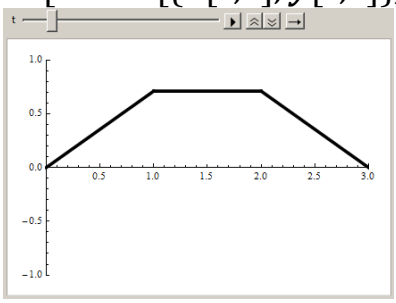
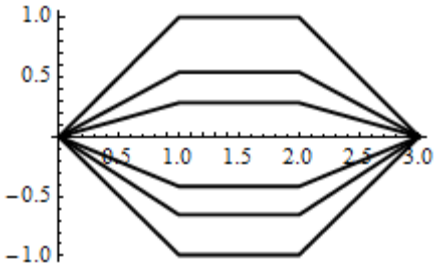
Зная функции $y_1(t), y_2(t)$, составим кусочное параметрическое уравнение струны и нарисуем начальное состояние системы

```
x[τ_, t_] = τ;
y[τ_, t_] = Piecewise[{{y1[t]τ, τ < 1},
                       {y1[t] + (y2[t] - y1[t])(τ - 1), τ < 2},
                       {y2[t] - y2[t](τ - 2), τ < 3}}, 0]
ParametricPlot[{x[t, 0], y[t, 0]}, {t, 0, 3},
 pilog-> {PointSize[Large], Blue, Point[{1, y1[0]}], Red, Point[{2, y2[0]}]}]
```



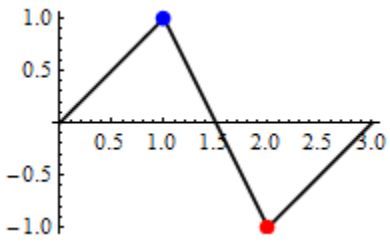
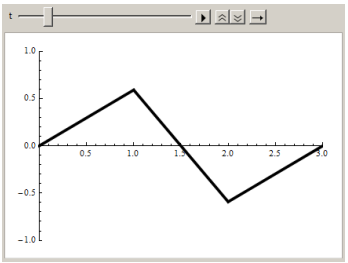
Теперь построим анимацию колебаний системы (след. рисунок слева) и график нескольких положений струны в различные моменты времени (рисунок справа).

```
Animate[
  ParametricPlot[Evaluate[{x[τ, t], y[τ, t]}], {τ, 0, 3},
                                  PlotRange → {{0, 3}, {−1, 1}}],
  {t, 0, 6}, AnimationRunning → False]
ParametricPlot[Table[{x[t, i], y[t, i]}, {i, 0, 5}], {t, 0, 3}]
```

Полученное синхронное колебание обеих масс представляет первую моду колебаний системы. Для построения второй моды колебаний, представляющей движение масс в «противофазе», можно повторить весь предыдущий код, только надо задать начальные смещения тоже в «противофазе».

```
Remove[y1, y2]
sys = {y1''[t] == y2[t] − 2y1[t], y2''[t] == −2y2[t] + y1[t],
                  y1[0] == 1, y2[0] == −1, y1'[0] == 0, y2'[0] == 0}
ds = DSolve[sys, {y1, y2}, t]
y1 = ds[[1, 1, 2]]
y2 = ds[[1, 2, 2]]
x[τ_, t_] = τ;
y[τ_, t_] = Piecewise[{{y1[t]τ, τ < 1},
                       {y1[t] + (y2[t] − y1[t])(τ − 1), τ < 2},
                       {y2[t] − y2[t](τ − 2), τ < 3}}, 0]
ParametricPlot[{x[t, 0], y[t, 0]}, {t, 0, 3},
 pilog−> {PointSize[Large], Blue, Point[{1, y1[0]}], Red, Point[{2, y2[0]}]}]
Animate[
    ParametricPlot[Evaluate[{x[τ, t], y[τ, t]}], {τ, 0, 3},
                                  PlotRange → {{0, 3}, {−1, 1}}],
    {t, 0, 6}, AnimationRunning → False]
```

Начальное состояние системы показано на предыдущем рисунке слева, а панель анимации – справа. Для создания произвольных колебаний (суперпозиции первой и второй мод) достаточно задать произвольные начальные смещения. Попробуйте, например, задать условия $y_1(0)=1$, $y_2(0)=0$.

Пример 7.2 *Поперечные колебания струны с n грузами.* Имеется n грузов массы M, расположенных в точках $x = a, 2a, 3a, \ldots, na$. Отрезки невесомой струны/нити между грузами одинаковы и подчиняются закону Гука. Натяжение в равновесии равно T. Обозначим вертикальное отклонение грузов от положения равновесия на оси струны через $y_i(t)$ $(i = 1,2,\ldots,n)$. Ограничимся рассмотрением малых колебаний. Уравнения движения имеют вид

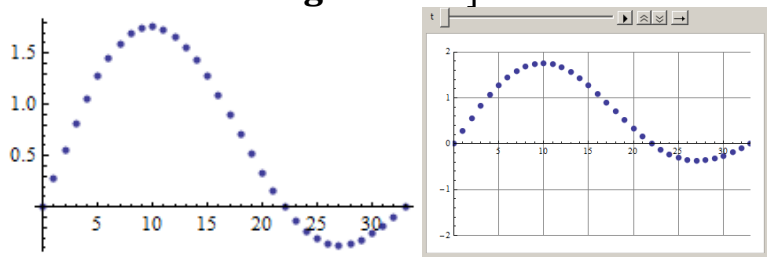
$$M\frac{d^2y_i}{dt^2} = T\left(\frac{y_{i+1}-y_i}{a}\right) - T\left(\frac{y_i - y_{i-1}}{a}\right) \quad (i = 1,2,\ldots,n)$$

При этом, в силу условия закрепления струны на концах, следует считать, что $y_0(t) = 0$, $y_{n+1}(t) = 0$. Рассмотрим колебание системы без начальной скорости. Положим a=1, M=1, T=1. Тогда имеем

$$\frac{d^2y_i}{dt^2} = y_{i+1} - 2y_i + y_{i-1} \quad (i = 1,2,\ldots,n)$$

Напишем код для решения этой системы ОДУ и построим анимацию движения грузов.

```
n = 32;
eqns = Flatten[Table[{yi''[t] == (yi+1[t] - 2yi[t] + yi-1[t]),
                      yi[0] == Sin[πi/(n+1)] + Sin[2πi/(n+1)], yi'[0] == 0}, {i, n}]];
eqns = Join[eqns, {y0[t] == 0, yn+1[t] == 0}];
vars = Table[yi, {i, 0, n + 1}];
sol = NDSolve[eqns, vars, {t, 0, 2(n + 1)}];
ListPlot[Table[{i, yi[0]}/. sol[[1]], {i, 0, n + 1}], PlotStyle → PointSize[0.02]]
Animate[
    ListPlot[Table[{i, yi[t]}/. sol[[1]], {i, 0, n + 1}],
        PlotStyle → PointSize[0.02], PlotRange → {{0, n + 1}, {−2, 2}},
        GridLines → Automatic],
    {t, 0, 2 n}, AnimationRunning → False]
```

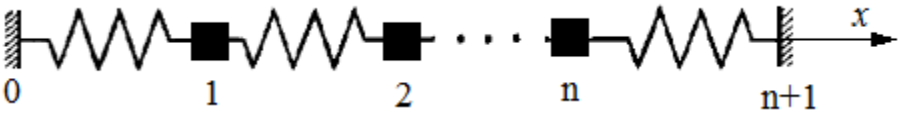


Здесь на левом графике показано начальное состояние системы, справа –панель анимации.

Попробуйте использовать большее/меньшее n. Задайте другие начальные смещения, например, $y_i[0] == \frac{n}{2} - \text{Abs}[i - \frac{n}{2}]$.

4.5.8 Модель Ферми – Улама – Пасты

Представим простейшую одномерную модель кристалла – цепочку одинаковых грузиков массы m, соединенных одинаковыми пружинками жесткостью k. Обозначим через y_i отклонение (вдоль оси X) грузика i от положения равновесия.



Тогда, в соответствии с законом Гука, уравнения движения *i*-го грузика имеет вид

$$m\frac{d^2y_i}{dt^2} = k(y_{i+1} - y_i) - k(y_i - y_{i-1}) \quad (i = 1,2,\ldots,n),$$

который полностью совпадает с уравнениям поперечных колебаний невесомой струны с грузами.

Ферми с коллегами С. Уламом и Дж. Пастой предположил, что сила $F(y_{i+1}, y_i)$, с которой взаимодействуют $(i+1)$ –й и i – й атомы имеет небольшую нелинейную добавку. Это значит, что возвращающая сила имеет вид $F(y_{i+1}, y_i) = k\,\Delta l + \alpha\,\Delta l^2$, где $\Delta l = y_{i+1} - y_i$ и α мало. Это приводит к следующей системе ОДУ

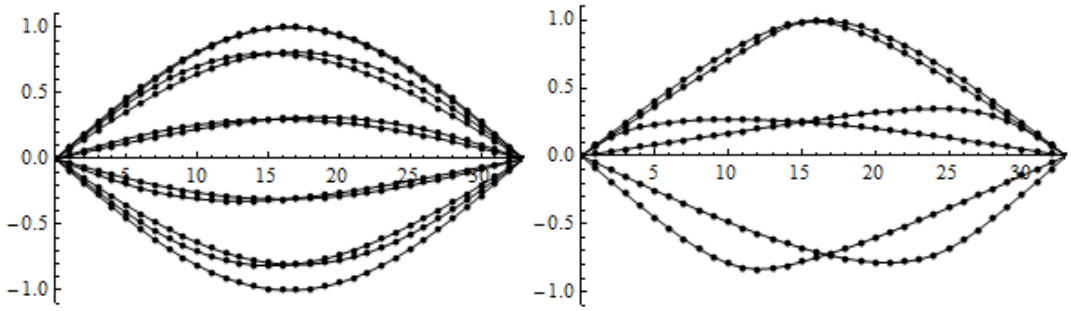
$$m\frac{d^2y_i}{dt^2} = k(y_{i+1} - 2y_i + y_{i-1}) + \alpha\left((y_{i+1} - y_i)^2 - (y_i - y_{i-1})^2\right) \quad (i = 1,2,\ldots,n)$$

При решении этой системы Ферми с коллегами следили не за отдельными частицами, а за поведением мод колебаний, которые при $\alpha = 0$ совпадают с синусоидами·

Рассмотрим движение, при котором в начальный момент времени возбуждена только одна первая мода с периодом T. Код решения этой задачи повторяет код предыдущего примера с той разницей, что уравнения системы ОДУ теперь нелинейные. Положим *k*=1, *n*=32, α=1 и начальные смещения зададим в форме первой моды $y_i(0) = \sin\left(\frac{\pi i}{n+1}\right)$.

```
n = 32; a = 1;
T = 66; (* период линейных колебаний *)
eqns1 = Flatten[Table[
                {y_i''[t] == (y_(i+1)[t] - 2y_i[t] + y_(i-1)[t]) +
                      a · ((y_(i+1)[t] - y_i[t])^2 - (y_i[t] - y_(i-1)[t])^2),
                 y_i[0] == Sin[πi/(n+1)], y_i'[0] == 0}, {i, n}]];
eqns1 = Join[eqns1, {y_0[t] == 0, y_(n+1)[t] == 0}];
vars1 = Table[y_i, {i, 0, n + 1}];
sol1 = NDSolve[eqns1, vars1, {t, 0, 200(n + 1)}, MaxSteps → 100000];
p1 = Table[ListPlot[Table[
              {i, y_i[tk]}/. sol1[[1]], {i, 0, n + 1}],
              PlotStyle → {PointSize[0.01], Red}, PlotRange → All,
              Joined → True, Mesh → All],
        {tk, 0, T, T/10}];
Show[p1, PlotRange → {{0, n + 1}, {-1.1, 1.1}}]
```

Обратите внимание, что при решении системы ОДУ использована опция MaxSteps→100000, которая необходима для большого временного интервала, на котором определяется численное решение.



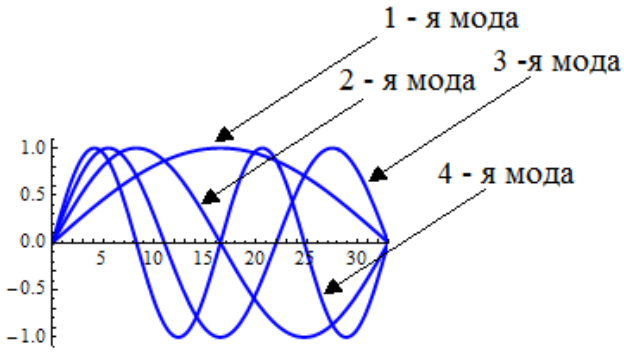
На предыдущем рисунке слева показаны формы колебаний в моменты времени $t = 0, \frac{T}{10}, \frac{2T}{10}, \ldots, T$. Следующий код строит графики решения при $t = 2T, 2T + \frac{T}{5}, \ldots, 3T$, которые показаны на предыдущем рисунке справа. Для наглядности на обоих графиках смещение грузов отложены в вертикальном направлении.

```
p2 = Table[ListPlot[Table[
                {i, yi[tk]}/. sol1[[1]], {i, 0, n + 1}],
                PlotStyle → {PointSize[0.01], Red}, PlotRange → All,
                Joined → True, Mesh → All],
        {tk, 2T, 3T, T/5}];
Show[p2, PlotRange → {{0, n + 1}, {−1.1, 1.1}}]
```

Как видим, форма решения все еще похожа на половину волны синусоиды, т.е. на первую моду колебаний. Однако, уже наблюдается сдвиг максимума в стороны от середины отрезка [0, n+1]. Первая мода еще узнаваема при $t \approx 7T$.

Следующий код строит графики 4 - х первых мод линейных колебаний.

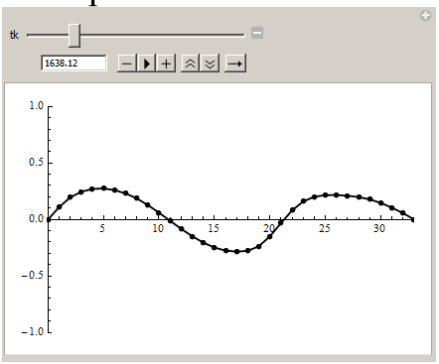
```
pl = Table[Plot[Sin[πxk/(n + 1)], {x, 0, n + 1},
                PlotStyle → {Blue, Thickness[0.01]}], {k, 1, 4}];
Show[pl, PlotRange → {{0, n + 1}, {−1.1, 1.1}}]
```

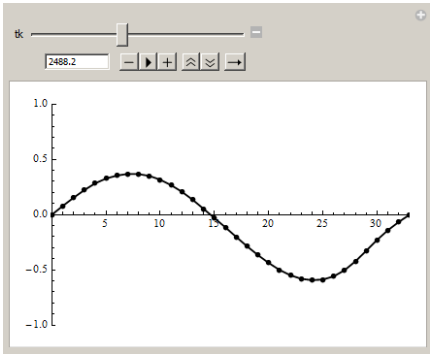


Вернемся к нелинейным колебаниям. При $t \approx 25T$ мы наблюдаем в основном 3 – ю моду. Чтобы в этом убедиться, достаточно использовать манипулятор (следующий рисунок слева) и код, приведенный ниже.

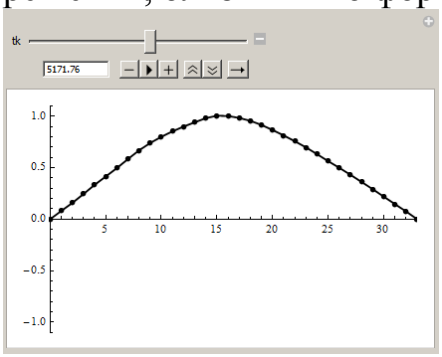
```
Manipulate[
    ListPlot[Table[{i, y_i[tk]}/. sol1[[1]], {i, 0, n + 1}],
        PlotStyle → {PointSize[0.015], Black, Thickness[0.005]},
        PlotRange → {{0, n + 1}, {−1, 1}}, Joined → True, Mesh → All],
{tk, 24T, 28T, 0.01T}]
```

При $t \approx 38.5T$ мы наблюдаем в основном 2 – ю моду. Измените в предыдущем коде последнюю строку на $\{\text{tk}, 36T, 40T, 0.01T\}]$ и вы получите манипулятор, показанный справа.





Графики решений, построенные при $t \approx 51.5T$ снова напоминают 3 – ю моду. При $t \approx 77T$ мы снова возвращаемся к первой моде. Внесите изменение в последнюю строку предыдущего кода $\{\text{tk}, 75T, 80T, 0.01T\}]$ и вы получите манипулятор с графиками решений, близкими по форме к 1 – й моде.



Подведем итог, который впервые наблюдали Ферми и его сотрудники. В начальные моменты времени форма колебаний близка к форме первой моды, но потом начинается перекачивание энергии в другие моды, и форма решения больше напоминает сумму нескольких синусоид с различным периодом (сумма мод). При $t \approx 25T$ возбуждена в основном 3 – я мода, при $t \approx 38.5T$ преобладает 2 – я мода. Затем при $t \approx 51.5T$ форма опять близка к 3 – й моде, и при $t \approx 77T$ снова возвращается к первой моде.

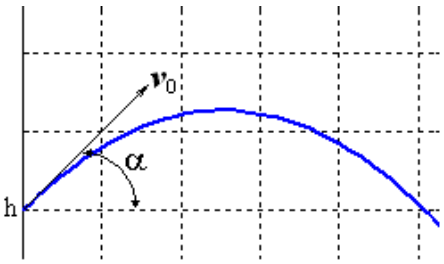
Численные эксперименты сотрудников Ферми показали, что такое поведение не случайность. Увеличение числа грузиков, изменение значения параметра α, изменение формы нелинейности (например, $\alpha \Delta l^3$ вместо $\alpha \Delta l^2$) не меняет поведение системы. Моды не сливаются, а происходит выделение некоторых из них, которые доминируют по очереди. Когда возвращается начальная мода, все повторяется. Время возвращения T_R (в нашем примере $T_R \approx 77\,T$) зависит от n, от вида нелинейности, но «солирование» низших мод и возвращение при $t = T_R$ наблюдалось сотрудниками Ферми во всех расчетах.

4.5.9 Движение тела, брошенного под углом к горизонту

Решим задачу Коши, описывающую движение тела, брошенного с начальной скоростью v_0 под углом α к горизонту в предположении, что сопротивление воздуха пропорционально квадрату скорости. В векторной форме уравнение движения имеет вид

$$m\ddot{\mathbf{r}} = -\gamma \cdot \mathbf{v} \left| \mathbf{v} \right| - m\mathbf{g},$$

где $\mathbf{r}(t)$ радиус – вектор движущегося тела, $\mathbf{v} = \dot{\mathbf{r}}(t)$ – вектор скорости тела, γ – коэффициент сопротивления, $m\mathbf{g}$ вектор силы веса тела массы m, $\mathbf{g}$ – вектор ускорения свободного падения.



Особенность этой задачи состоит в том, что движение заканчивается в заранее неизвестный момент времени, когда тело падает на землю.

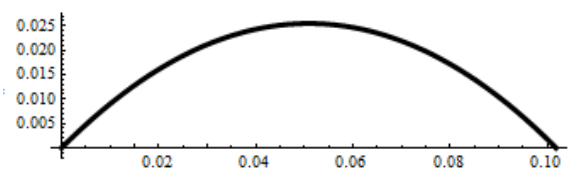
Если обозначить $k = \gamma / m$, то в координатной форме мы имеем систему уравнений

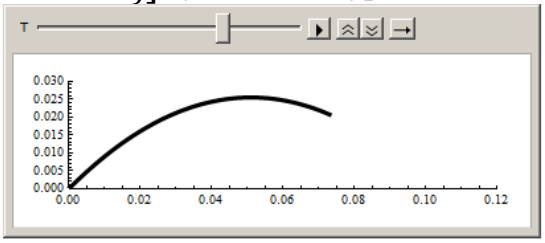
$$\ddot{x} = -k\,\dot{x}\sqrt{\dot{x}^2 + \dot{y}^2}$$

$$\ddot{y} = -k\,\dot{y}\sqrt{\dot{x}^2 + \dot{y}^2} - g$$

к которой следует добавить начальные условия: $x(0) = 0$, $y(0) = h$ (h начальная высота), $\dot{x}(0) = v_0 \cos\alpha$, $\dot{y}(0) = v_0 \sin\alpha$.

```
α = π/4; v0 = 1; h = 0; k = 0.01; g = 9.81;
sys = {x''[t] == -k x'[t] Sqrt[x'[t]^2 + y'[t]^2],
       y''[t] == -k y'[t] Sqrt[x'[t]^2 + y'[t]^2] - g,
       x[0] == 0, y[0] == h, x'[0] == v0Cos[α], y'[0] == v0Sin[α]};
s = NDSolveValue[Join[sys,
       {WhenEvent[y[t] == 0, "StopIntegration"]}], {x, y}, {t, 0, ∞}]
tmax = s[[1, 1, 1, 2]]
ParametricPlot[Evaluate[{s[[1]][t], s[[2]][t]}], {t, 0, tmax}]   (рис. слева)
Animate[
ParametricPlot[Evaluate[{s[[1]][t], s[[2]][t]}], {t, 0, T},
            PlotRange → {{0, 0.12}, {0, 0.03}}], {T, 0, tmax}]
```



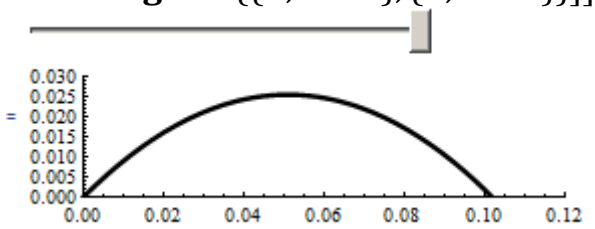


Чтобы для определения длительности полета нам не приходится подбирать значение `tmax` «на глазок», мы использовали функцию `WhenEvent` для

определения момента наступления события «высота полета равна нулю» и задали реакцию на это событие `StopIntegration` (остановка вычислений). Момент времени, в который произошло событие остановки вычислений, является параметром решения «InterpolationFunction» и доступ к нему мы получили с помощью индексации.

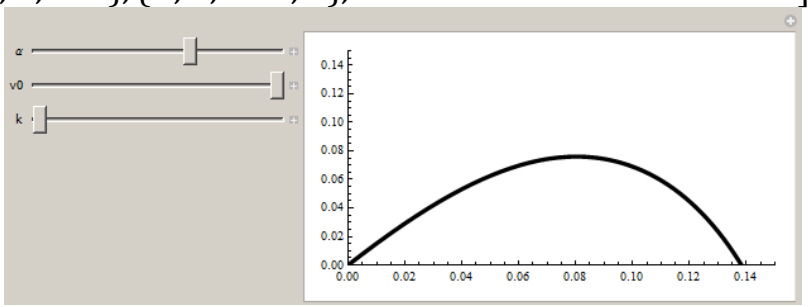
Вместо анимации можно использовать элемент «слайдер»

```
DynamicModule[{T =.01},
  {Slider[Dynamic[T],{T,0.2}],
          Dynamic[
              ParametricPlot[Evaluate[{s[[1]][t],s[[2]][t]}],{t,0,T},
                  PlotRange → {{0,0.12},{0,0.03}}]]}//TableForm]
```



Для наблюдением за поведением траектории тела при различных углах «бросания» α , начальных скоростях v_0 и коэффициентах сопротивления среды k удобно использовать манипулятор.

```
h = 0; g = 9.81;
Manipulate[
  sys = {x''[t] == -kx'[t]√(x'[t]^2 + y'[t]^2),
         y''[t] == -ky'[t]√(x'[t]^2 + y'[t]^2) - g, x[0] == 0, y[0] == h};
   s = NDSolveValue[Join[sys,{x'[0] == v0Cos[α], y'[0] == v0Sin[α]},
          {WhenEvent[y[t] == 0,"StopIntegration"]}],{x,y},{t,0,∞}];
  tmax = s[[1,1,1,2]];
  ParametricPlot[Evaluate[{s[[1]][t],s[[2]][t]}],{t,0,tmax},
       PlotRange → {{0,0.15},{0,0.15}}, AspectRatio → 0.5],
  {{α,π/4},0.1,π/2 - 0.06,0.1},
  {{v0,1},0.1,2,0.1},{k,5,100,1}, ControlPlacement → Left]
```



4.5.10 Движение тел под действием тяготения

Пример 10.1 Исследуем задачу о движении планеты вокруг Солнца под действием тяготения.

Закон всемирного тяготения Ньютона в векторной форме имеет вид $m\mathbf{a}=F\mathbf{e}$, где $\boldsymbol{a}$ – вектор ускорения, $\mathbf{e}$ – единичный вектор, направленный от планеты к солнцу, $F=\gamma\frac{mM}{R^2}$ – величина силы притяжения, R – расстояние между притягивающимися телами, m – масса планеты, M – масса солнца, γ – гравитационная постоянная. Свяжем начало координат с массивным телом (солнцем). В любой момент времени функции $(x(t),y(t))$ представляют радиус-вектор планеты. Сила, с которой солнце притягивает планету, будет направлена вдоль единичного вектора

$$\mathbf{e}=\left(\frac{-x(t)}{\sqrt{x^2(t)+y^2(t)}},\frac{-y(t)}{\sqrt{x^2(t)+y^2(t)}}\right)$$

имеющего противоположное направление радиус-вектору планеты. Тогда

$$m\begin{Bmatrix}x''(t)\\y''(t)\end{Bmatrix}=-\gamma\frac{mM}{(x^2(t)+y^2(t))^{3/2}}\begin{Bmatrix}x(t)\\y(t)\end{Bmatrix},$$

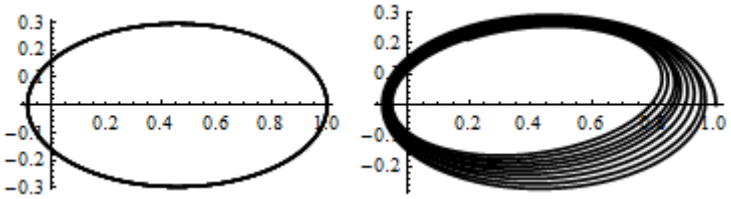
Введем обозначение $\gamma M=k$. Получим следующую систему обыкновенных дифференциальных уравнений относительно неизвестных функций $x(t)$, $y(t)$ – координат движущейся планеты:

$$\begin{cases}x''(t)=-\dfrac{k\,x(t)}{(x^2(t)+y^2(t))^{3/2}}\\[2ex] y''(t)=-\dfrac{k\,y(t)}{(x^2(t)+y^2(t))^{3/2}}\end{cases}$$

Для модельной задачи выберем k=1 и зададим начальные условия $x(0)=1,\ x'(0)=0,\ y(0)=0,\ y'(0)=1$. Решим задачу при погрешностях вычислений, заданных для функции `NDSolve` по – умолчанию, и при пониженных погрешностях.

```
sys = {x''[t] == -x[t]/(x[t]^2 + y[t]^2)^(3/2), y''[t] == -y[t]/(x[t]^2 + y[t]^2)^(3/2),
       x[0] == 1, x'[0] == 0, y[0] == 0, y'[0] == 0.4}
s = NDSolveValue[sys, {x, y}, {t, 0, 20}]
sb = NDSolveValue[sys, {x, y}, {t, 0, 20},
                  AccuracyGoal → 3, PrecisionGoal → 6]
ParametricPlot[{s[[1]][t], s[[2]][t]}, {t, 0, 20}, AspectRatio → Automatic]
ParametricPlot[{sb[[1]][t], sb[[2]][t]}, {t, 0, 20}, AspectRatio → Automatic]
```

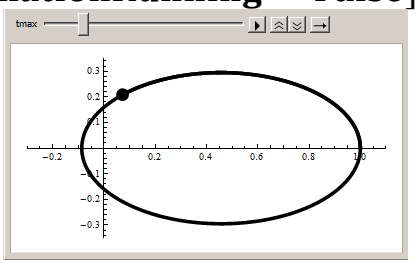
На следующем рисунке показана получаемая траектория движения планеты (кривая с параметрическим уравнением $x(t), y(t)$)



Левая траектория получена при стандартной точности вычислений, правая – при пониженной. В решении `sb` отчетливо видно как накопление погрешности вычислений влияет на результат.

Можно построить анимацию движения

```
s = NDSolveValue[sys, {x, y}, {t, 0, 20}]
Animate[
    p1 = ParametricPlot[Evaluate[{s[[1]][t], s[[2]][t]}], {t, 0, tmax},
            PlotRange → {{-0.3, 1.1}, {-0.35, 0.35}}, PlotPoints → 1000];
    p2 = Graphics[Disk[{s[[1]][tmax], s[[2]][tmax]}, 0.025]];
    Show[{p1, p2}],
    {tmax, 0, 20, 0.1}, AnimationRunning → False]
```



Пример 10.2 Смоделируем задачу о запуске искусственного спутника на орбиту Луны. В этом случае в уравнениях движения спутника под действием тяготения будет присутствовать внешняя сила $\mathbf{f}(t)$, действующая в течение ограниченного времени. Ее величину и длительность действия надо подобрать так, чтобы спутник вышел на орбиту Луны.

Отметим, что спутник должен подняться на достаточную высоту, чтобы его орбита не захватывала поверхность планеты. Поэтому внешняя сила должна вначале быть вертикальной (или иметь большую вертикальную составляющую), а затем быть горизонтальной, чтобы придать спутнику необходимую орбитальную скорость. Если вертикальная составляющая ускорения будет не достаточной, то траектория спутника (эллипс) будет задевать поверхность планеты.

В качестве центрального тела будем рассматривать Землю, с которой свяжем начало координат. Луна двигается под действием притяжения Земли и ее координаты обозначим $x_m(t), y_m(t)$. Координаты спутника обозначим $x_s(t), y_s(t)$, начальные значения которых должно быть на поверхности Земли. Притяжением Луны к спутнику можно пренебречь. Уравнения движения спутника будут иметь вид

$$\begin{cases} x_s''(t) = -\dfrac{\gamma M\, x_s(t)}{(x_s{}^2(t) + y_s{}^2(t))^{3/2}} + a_x(t) + \dfrac{\gamma m(x_s(t) - x_m(t))}{((x_s(t) - x_m(t))^2 + (y_s(t) - y_m(t))^2)^{3/2}} \\ y''(t) = -\dfrac{\gamma M\, y_s(t)}{(x_s{}^2(t) + y_s{}^2(t))^{3/2}} + a_y(t) + \dfrac{\gamma m(y_s(t) - y_m(t))}{((x_s(t) - x_m(t))^2 + (y_s(t) - y_m(t))^2)^{3/2}} \end{cases}$$

Здесь M – масса Земли, m – масса Луны, $(a_x(t), a_y(t))$ - вектор ускорения, вызываемый внешней силой (работающим двигателем ракеты). Заметим, что мы не учитываем факт изменения массы спутника, возникающего при сгорании топлива ракеты.

Луна также двигается под действием притяжения Земли и уравнения ее движения (такие же как в предыдущем примере) имеют вид

$$\begin{cases} x_m''(t) = -\dfrac{\gamma M\, x_m(t)}{(x_m{}^2(t) + y_m{}^2(t))^{3/2}} \\ y_m''(t) = -\dfrac{\gamma M\, y_m(t)}{(x_m{}^2(t) + y_m{}^2(t))^{3/2}} \end{cases}$$

В результате мы получаем систему четырех ОДУ, к которым надо добавить начальные условия. Здесь мы рассмотрим модельную задачу с вымышленными значениями масс и размеров.

Начальные условия спутника определяют, что в нулевой момент времени ракета-носитель спутника находится на поверхности планеты (радиуса 1) и имеет небольшую нормальную к поверхности скорость (вдоль оси X)

$$x_s(0) = 1,\quad y_s(0) = 0,\ x_s'(0) = 0.2,\ y_s'(0) = 0.$$

Начальные условия для Луны определяют ее положение и скорость

$$x_m(0) = 1.9,\quad y_m(0) = 0,\ x_m'(0) = 0,\ y_m'(0) = 0.8$$

Решим задачу и построим анимацию движения Луны и спутника, а также анимацию видимого движения спутника с поверхности Луны.

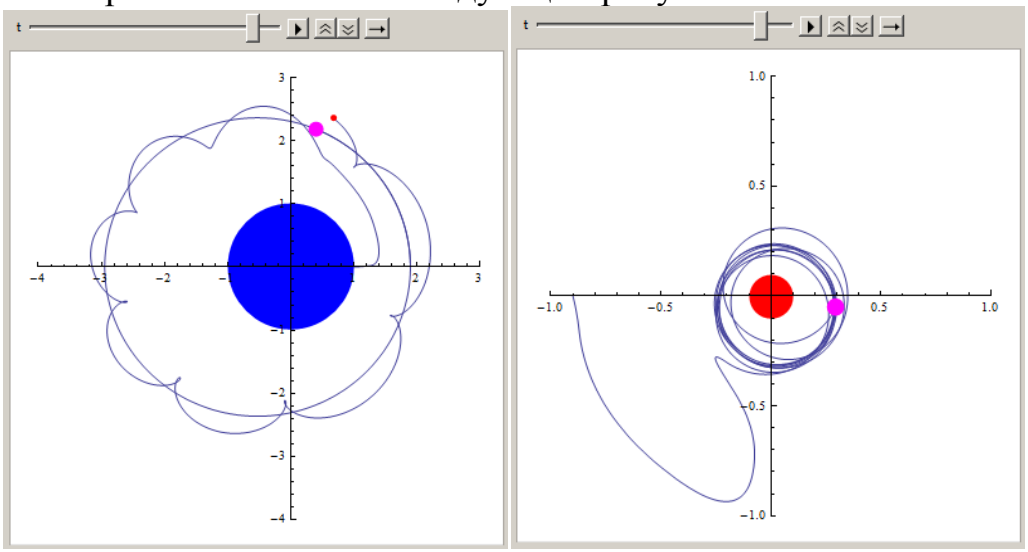
```
Remove[xs, ys, xm, ym, x, y, x1, y1]
M = 10; m = 1; γ = 0.1; t0 = 30;
ax[t_] = Piecewise[{{0, t ≤ 0.2}, {1.4, t ≤ 1}, {-0.5, t ≤ 2}, {-0.9, t
      ≤ 2.5}, {1, t ≤ 3.1}, {0, t > 3.1}}];
ay[t_] = Piecewise[{{0, t ≤ 0.2}, {0, t ≤ 1}, {1.2, t ≤ 2}, {-0.2, t ≤ 2.5}, {-2, t
      ≤ 3.1}, {0, t > 3.1}}];
eq1 = xs''[t] == -γ M xs[t] / (xs[t]^2 + ys[t]^2)^(3/2) + ax[t] +
      γ m (xm[t] - xs[t]) / ((xm[t] - xs[t])^2 + (ym[t] - ys[t])^2)^(3/2);
eq2 = ys''[t] == -γ M ys[t] / (xs[t]^2 + ys[t]^2)^(3/2) + ay[t] +
      γ m (ym[t] - ys[t]) / ((xm[t] - xs[t])^2 + (ym[t] - ys[t])^2)^(3/2);
```

$$\mathbf{eq3} = \mathbf{xm}''[t] == \frac{-\gamma M \mathbf{xm}[t]}{(\mathbf{xm}[t]^2 + \mathbf{ym}[t]^2)^{3/2}};$$

$$\mathbf{eq4} = \mathbf{ym}''[t] == \frac{-\gamma M \mathbf{ym}[t]}{(\mathbf{xm}[t]^2 + \mathbf{ym}[t]^2)^{3/2}};$$

```
bc = {xs[0] == 1, ys[0] == 0, xs'[0] == 0.2, ys'[0] == 0,
      xm[0] == 1.9, ym[0] == 0, xm'[0] == 0, ym'[0] == 0.8};
sys = Join[{eq1, eq2, eq3, eq4}, bc];
s = NDSolve[sys, {xs, ys, xm, ym}, {t, 0, t0}]
x = xs/.s[[1, 1]]; y = ys/.s[[1, 2]];
x1 = xm/.s[[1, 3]]; y1 = ym/.s[[1, 4]];
Animate[
  pp = ParametricPlot[{x[τ], y[τ]}, {τ, 0, t}, PlotRange → {{-4, 3}, {-4, 3}}];
  ss = ParametricPlot[{x1[τ], y1[τ]}, {τ, 0, t}, PlotRange → {{-4, 3}, {-4, 3}}];
  ssd = Graphics[{RGBColor[1, 0, 0], Disk[{x[t], y[t]}, 0.05]}];
  gr = Graphics[{RGBColor[1, 0, 1], Disk[{x1[t], y1[t]}, 0.12]}];
  earth = Graphics[{RGBColor[0, 0, 1], Disk[{0, 0}, 1]}];
  Show[{pp, ssd, ss, gr, earth}, Axes → True],
  {t, 0.1, 30, 0.05}, AnimationRunning → False]
```

Результат построения показан на следующем рисунке слева.



Следующий код строит траекторию движения спутника, наблюдаемую с Луны.

```
Animate[
  pp = ParametricPlot[{x[τ] - x1[τ], y[τ] - y1[τ]}, {τ, 0, t},
                      PlotRange → {{-1, 1}, {-1, 1}}];
  ssd = Graphics[{RGBColor[1, 0, 0], Disk[{0, 0}, 0.1]}];
  gr = Graphics[{RGBColor[1, 0, 1], Disk[{x[t] - x1[t], y[t] - y1[t]}, 0.04]}];
  Show[{pp, ssd, gr}, Axes → True],
  {t, 0.1, 30, 0.05}, AnimationRunning → False]
```

Траектория спутника, наблюдаемая с Луны, показана на предыдущем рисунке справа.

4.5.11 Математический маятник

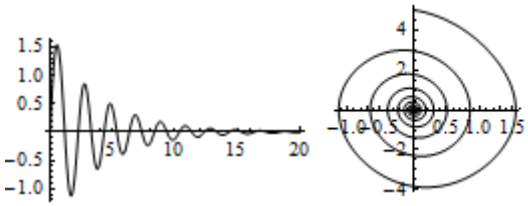
Исследуем поведения математического маятника. Пусть масса груза равна единице, а стержень, на котором подвешена масса, невесом. Тогда дифференциальное уравнение движения груза имеет вид

$$\varphi'' + k\,\varphi' + \omega^2 \sin\varphi = 0$$

где $\varphi(t)$ угол отклонения маятника от положения равновесия (нижнее положение), параметр k характеризует величину трения, $\omega^2 = g/l$ (g ускорение свободного падения, l – длина маятника). Для определения конкретного движения к уравнению движения надо добавить начальные условия $\varphi(0) = \varphi_0$, $\varphi'(0) = \varphi_0'$. Выберем следующие значения параметров k=0.5, $\omega^2 = 10$ и начальные значения $\varphi_0 = 0$, $\varphi_0' = 5$.

Решаем задачу, строим график решения (следующий рисунок слева) и фазовую траекторию (рисунок справа)

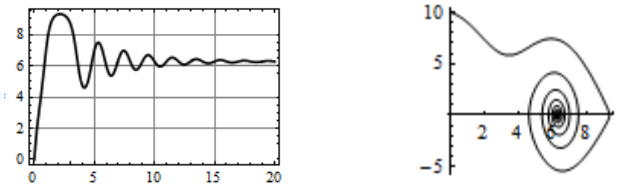
```
k = 0.5; w = 10;
s = NDSolveValue[{x''[t] + k x'[t] + w Sin[x[t]] == 0, x[0] == 0,
                  x'[0] == 5}, x, {t, 0, 20}]
Plot[s[t], {t, 0, 20}]
ParametricPlot[Evaluate[{s[t], s'[t]}], {t, 0, 20}, AspectRatio → 1]
```



Как видно из левого графика, максимальный угол отклонения маятника не превышают $\pi/2$ и колебания маятника затухают.

Увеличим начальную скорость до 10. Решаем задачу, строим график решения (следующий рисунок слева) и фазовую траекторию (рисунок справа)

```
s = NDSolveValue[{x''[t] + k x'[t] + w Sin[x[t]] == 0, x[0] == 0,
                  x'[0] == 10}, x, {t, 0, 20}]
Plot[s[t], {t, 0, 20}]
p1=ParametricPlot[Evaluate[{s[t], s'[t]}], {t, 0, 20}, AspectRatio → 1]
```

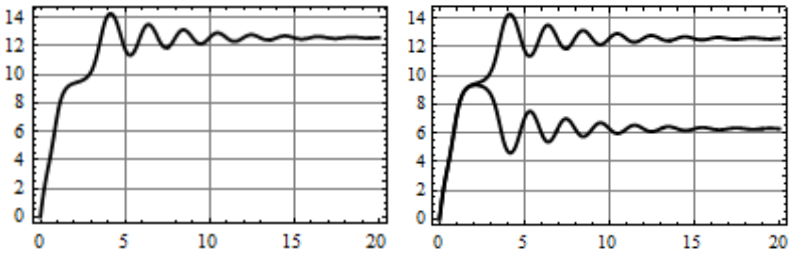


Максимальное значение угла составляет примерно 10 радиан. Маятник сделал один полный оборот вокруг точки закрепления (угол отклонения увеличился на $2\cdot\pi$), а затем колебания затухают в окрестности значения $2\cdot\pi$ (для маятника угол поворота $2\cdot\pi$ представляет то же, что и 0 радиан, т.е. положение равновесия).

Обратим внимание на следующий факт – точность вычислений в последнем примере имеет существенное значение. Снизим относительную и

абсолютную погрешности опциями `AccuracyGoal→3,PrecisionGoal→3` и найдем решение той же задачи

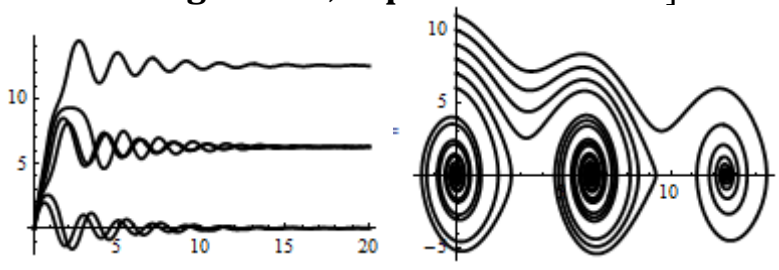
```
sbad = NDSolveValue[{x''[t] + kx'[t] + wSin[x[t]] == 0,
  x[0] == 0, x'[0] == 10}, x, {t, 0, 20}, AccuracyGoal → 3, PrecisionGoal →
3]
p2 = Plot[sbad[t], {t, 0, 20}, GridLines → Automatic,
                                Frame → True, PlotRange → All]
Show[p1, p2]
```



На правом рисунке показаны оба графика решений: второй более высокий, полученный при пониженной точности вычислений, и первый – полученный при стандартной точности. Отличие весьма значительное. Верхняя кривая представляет маятник, сделавший два полных оборота вокруг точки подвеса, а нижняя – маятник, сделавший только один полный оборот. Изменение точности привело к «качественному» изменению решения – два оборота маятника вместо одного!

Построим несколько графиков угла отклонения (след. рис. слева) и фазовых траекторий (след. рис. справа), задавая различную начальную скорость.

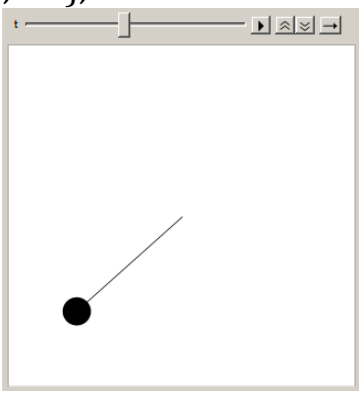
```
k = 0.5; w = 10;
s = ParametricNDSolveValue[{x''[t] + kx'[t] + wSin[x[t]] == 0,
                            x[0] == 0, x'[0] == v0}, x, {t, 0, 20}, {v0}]
Plot[Table[s[v0][t], {v0, 6, 11, 1}], {t, 0, 20},
                          PlotStyle → {Thickness[0.01], {Black}}]
ParametricPlot[Table[{s[v0][t], s[v0]'[t]}, {v0, 6, 11, 1}], {t, 0, 20},
                        PlotRange → All, AspectRatio → 0.7]
```



Как видим, начальная скорость при *v=6, 7* недостаточна, чтобы маятник прошел верхнюю точку и сделал хотя бы один полный оборот. При начальной скорости *v= 8, 9, 10* маятник совершает один полный оборот, а затем его колебания затухаю. При *v=11* маятник смог выполнить два полных оборота и только после этого его колебания стали затухать вокруг положения равновесия.

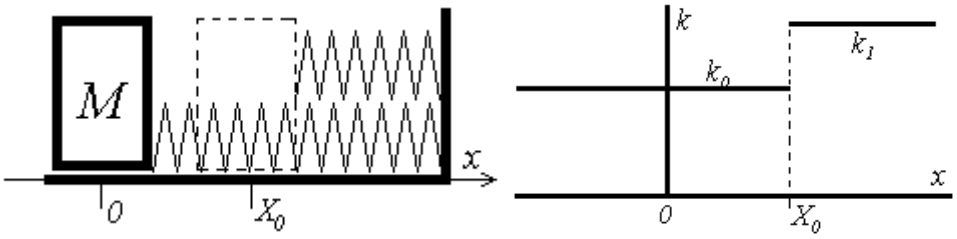
Для наглядности, построим еще анимацию движения модели маятника

```
φ = NDSolveValue[{x''[t] + kx'[t] + wSin[x[t]] == 0,
                  x[0] == 0, x'[0] == 10}, x, {t, 0, 20}]
g1[t_] = Line[{{0, 0}, {-Sin[φ[t]], -Cos[φ[t]]}}];
d1[t_] = Disk[{-Sin[φ[t]], -Cos[φ[t]]}, 0.1];
Animate[Graphics[{g1[t], d1[t]}, PlotRange → {{-1.1, 1.1}, {-1.1, 1.1}}],
        {t, 0, 4π}, AnimationRunning → False]
```



4.5.12 Колебание массивного тела под действием пружин

Пример 12.1 Решим задачу о колебании массивного тела массы M на невесомых пружинах, одна из которых короче другой и короткая не привязана к телу (см. рисунок), трение отсутствует.



Для тела M выполняется уравнение движения $\ddot{x}(t)+k^2x(t)=0$. Здесь коэффициент k отвечает за жесткость пружины. При движении вправо и достижении положения X_0 тело M присоединяется ко второй пружине и жесткость системы меняется. Когда тело движется влево, и проходит положение X_0, оно отрывается от второй пружины и жесткость уменьшается. Т.о. жесткость зависит от смещения груза M относительно положения равновесия, т.е. $k=k(x)$. При этом функция $k(x)$ является кусочно постоянной. График функции $k(x)$ приведен на предыдущем рисунке справа.

Пусть $k_0=1$, $k_1=1.5$ и начальные значения $x(0)=-1$, $x'(0)=v_0=1$. Для сравнения решим задачу при постоянном k=1 и при кусочно – постоянной жесткости k. Тогда

```
uc = NDSolve[{x''[t] + x[t] == 0, x[0] == -1, x'[0] == 0}, x, {t, 0, 20}]
up = NDSolve[{x''[t] + (1 + 0.5UnitStep[x[t]])^2 x[t] == 0,
              x[0] == -1, x'[0] == 0}, x, {t, 0, 20}]
Plot[{x[t]/.uc[[1, 1]], x[t]/.up[[1, 1]]}, {t, 0, 20}]
ParametricPlot[{Evaluate[{x[t]/.up[[1, 1]], x'[t]/.up[[1, 1]]}],
                Evaluate[{x[t]/.uc[[1, 1]], x'[t]/.uc[[1, 1]]}]}, {t, 0, 20}]
```

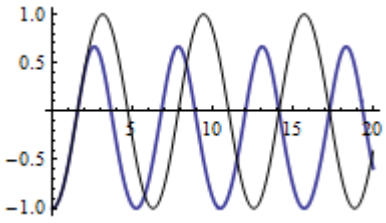
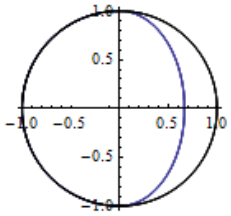



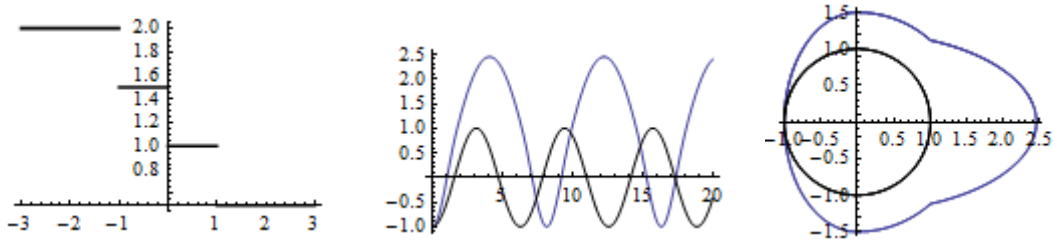
График решения показан на следующем рисунке синей (более низкой) кривой. Черная (более высокая кривая) представляет решение для случая неизменного `k=1`. На правом рисунке показана фазовая траектория синей кривой, черная (окружность) представляет фазовую траекторию для `k=1`.

Здесь для функции $k(x)$ мы выбрали представление `1+0.5 UnitStep[x[t]]`. Допустимо использование функции Хэвисайда `1+0.5 HeavisideTheta[x[t]]`. Можно также создать свою кусочную функцию `f[x_]=Piecewise[{{1,x<0}},1.5]` и использовать ее при решении уравнения, например, так

```
up = NDSolve[{x''[t] + f[x[t]]^2 x[t] == 0,
              x[0] == -1, x'[0] == 0}, x, {t, 0, 20}]
```

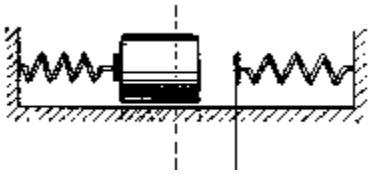
В последнем случае кусков функции f можно задать много.

```
f[x_] = Piecewise[{{2, x < -1}, {1.5, x < 0}, {1, x < 1}}, 0.5];
Plot[f[x], {x, -3, 3}]
uc = NDSolve[{x''[t] + x[t] == 0, x[0] == -1, x'[0] == 0}, x, {t, 0, 20}]
up = NDSolve[{x''[t] + f[x[t]]^2 x[t] == 0,
              x[0] == -1, x'[0] == 0}, x, {t, 0, 20}]
Plot[{x[t] /. up[[1, 1]], x[t] /. uc[[1, 1]]}, {t, 0, 20}]
ParametricPlot[{Evaluate[{x[t] /. up[[1, 1]], x'[t] /. up[[1, 1]]}],
                Evaluate[{x[t] /. uc[[1, 1]], x'[t] /. uc[[1, 1]]}]}, {t, 0, 20},
               AspectRatio → Automatic, PlotPoints → 1000]
```



На предыдущем рисунке слева показана, созданная нами кусочная функция, в середине – график решения (синяя кривая), справа – фазовая траектория (синяя кривая).

Пример 12.2. Рассмотрим систему, показанную на следующем рисунке. Груз массы m находится посередине, обе пружины касаются его, но не соединяются с ним. Когда груз находится посередине, растяжение пружин равно нулю. Жесткости пружин разные и их массой можно пренебречь. При смещении груза из положения равновесия возникают колебания.



Уравнение движения груза при $x<0$ имеет вид

$$\ddot{x}+\frac{c_1}{m}x=0,\ \ x<0$$

Уравнение движения груза при $x>0$ имеет вид

$$\ddot{x}+\frac{c_2}{m}x=0,\ \ x>0$$

Таким образом, единое уравнение можно записать в виде

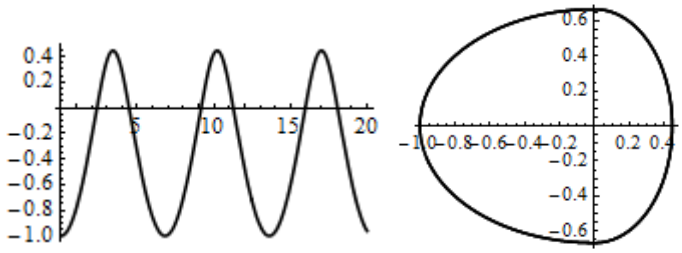
$$\ddot{x}+k^2x=0, \tag{1}$$

где

$$k^2(x)=\begin{cases}c_1/m,\ x<0\\ c_2/m,\ x>0\end{cases}=\begin{cases}k_1^2,\ x<0\\ k_2^2,\ x>0\end{cases}=k_1^2+\left(k_2^2-k_1^2\right)H(x)$$

где $H(x)$ – функция Хэвисайда. Обратите внимание, что коэффициент $k(x)$ зависит от решения $x(t)$. Функция `DSolve` не может решить эту задачу, поэтому используем `NDSolve`.

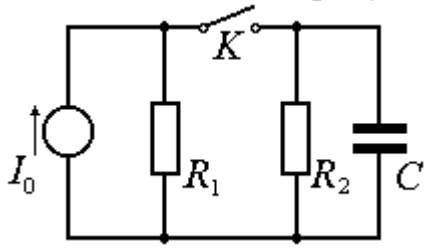
```
k1 = 2/3; k2 = 3/2; k[x_] = Piecewise[{{k1^2, x < 0}, {k2^2, x ≥ 0}}];
s = NDSolve[{x''[t] + k[x[t]]x[t] == 0, x[0] == -1, x'[0] == 0}, x, {t, 0, 20}]
Plot[s[[1, 1, 2]][t], {t, 0, 20}]
ParametricPlot[Evaluate[{s[[1, 1, 2]][t], D[s[[1, 1, 2]][t], t]}], {t, 0, 20}]
```



Слева показан график зависимости решения от времени, справа – фазовая траектория.

4.5.13 Генератор пилообразных напряжений

Пример 13.1. Простой генератор напряжений пилообразной формы состоит из параллельно соединенных емкости C и сопротивления R_2, которые периодически подключаются к параллельно соединенным генератором постоянного тока I_0 и сопротивлением R_1 (см. рисунок).



Коммутирующее устройство, периодически осуществляющее включение и отключение, ради простоты на схеме представлено в виде рубильника K, который замкнут в течение первой части цикла длительностью t_1 секунд и разомкнут в течение второй части периода длительностью T-t_1 секунд. Задача состоит в определении закона нарастания напряжения между обкладками конденсатора C (в начальный момент конденсатор разряжен).

Здесь мы рассмотрим не периодический процесс, а только один шаг, состоящий во включении и отключении рубильника. Обозначим через $u_1(t)$ напряжение конденсатора при замкнутом рубильнике, т.е. в интервале времени $0=t_0<t<t_1$, и $u_2(t)$ напряжение на конденсаторе при разомкнутом рубильнике для моментов времени $t>t_1$.

Дифференциальные уравнения для разных интервалов времени имеют вид

$$C\frac{d\,u_1}{d\,t}+\frac{1}{R_0}u_1=I_0 \text{ для } 0\le t\le t_1;$$

$$C\frac{d\,u_2}{d\,t}+\frac{1}{R_2}u_2=0 \text{ для } t\ge t_1,$$

где $\frac{1}{R_0}=\frac{1}{R_1}+\frac{1}{R_2}$. Так как напряжение на конденсаторе может меняться только непрерывно, то значение u_1 в момент t_1 равно начальному значению u_2 в этот же момент времени $u_1(t_1)=u_2(t_1)$. Оба эти уравнения мы можем представить в виде одного ДУ $\frac{d\,u}{d\,t}+a(t)\cdot u=f(t)$, где $a(t)$ и $f(t)$ кусочно-постоянные функции.

На участке $0\le t<t_1$ функция $a(t)$ равна $a_1=\frac{1}{C\,R_0}$, и для $t\ge t_1$ равна $a_2=\frac{1}{C\,R_2}$.

Функция $f(t)$ на участке $0\le t<t_1$ равна $f_1=\frac{I_0}{C}$, а для $t\ge t_1$ равна $f_2=0$.

Соответственно функция $u(t)$ на этих временных участках равна $u_1(t)$ и $u_2(t)$. Т.о. мы приходим к задаче

$$\frac{d\,u}{d\,t}+a(t)\cdot u=f(t),\ u(0)=0,$$

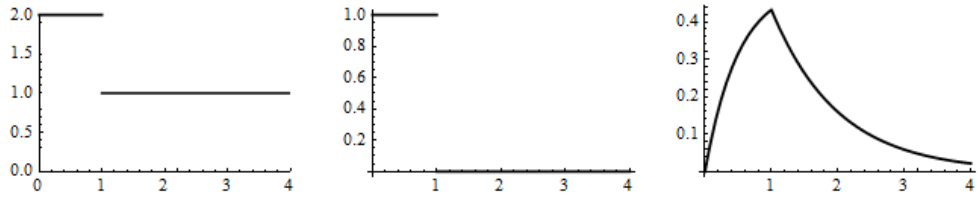
где

$$a(t)=\begin{cases}a_1\,,\ t<t_1\\ a_2\,,\ t>t_1\end{cases} \qquad \text{и} \qquad f(t)=\begin{cases}f_1\,,\ t<t_1\\ 0\,,\ t>t_1\end{cases}$$

Теперь можно написать код решения задачи (положим C=1).

```
R1 = 1; R2 = 1; I0 = 1; f1 = I0; t1 = 1;
R0 = R1 R2/(R1 + R2); a1 = 1/R0; a2 = 1/R2;
a[t_] = Piecewise[{{a1, t < t1}}, a2];
f[t_] = Piecewise[{{f1, t < t1}}, 0];
sU = DSolve[{u'[t] + a[t]u[t] == f[t], u[0] == 0}, u, t];
p1 = Plot[a[t], {t, 0, 4}, PlotRange → {{0, 4}, {0, 2}}];
```

```
p2 = Plot[f[t], {t, 0, 4}];
p3 = Plot[sU[[1, 1, 2, 2]], {t, 0, 4}];
GraphicsRow[{p1, p2, p3}]
```



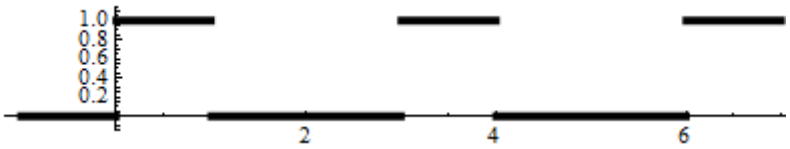
Слева показан график функции $a(t)$, в середине – функции $f(t)$, справа – график решения.

Пример 13.2. Пусть рубильник включается и выключается периодически. Это значит, что функции $a(t)$ и $f(t)$ должны совпадать с одноименными функциями из предыдущего примера на отрезке периода T и затем повторяться периодически. Для моделирования кусочно – постоянной периодической функции создадим одну специальную функцию, которую назовем *Periodic Step Function = Psf*

$$Psf(x,a,w) = \left[\frac{x}{w}\right] - \left[\frac{x-a}{w}\right]$$

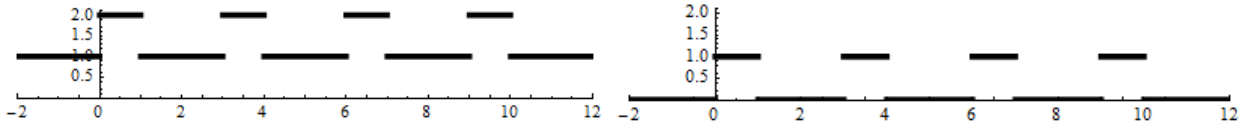
Это периодическая с периодом w функция. Для $a<w$ на отрезке $0 \le x < a$ функция равна 1, на остальном куске периода $a \le x < w$ функция равна 0. Случай $a \ge w$ мы не рассматриваем. На следующем рисунке приведен график функции *Psf*(*x*,*1*,*3*).

```
Psf[x_, a_, w_] = Floor[x/w] - Floor[(x - a)/w];
Plot[Psf[x, 1, 3], {x, -1, 7}, AspectRatio → Automatic]
```

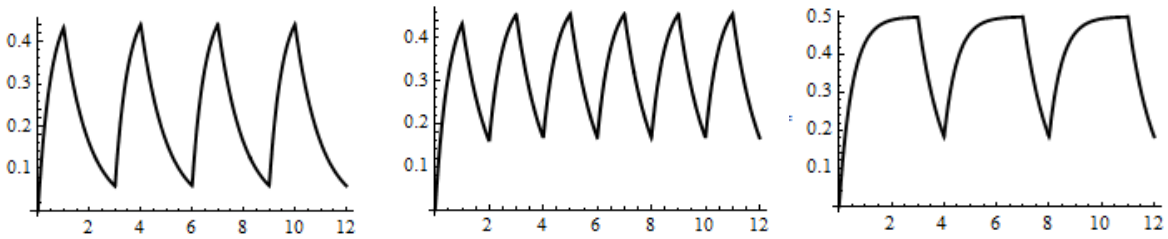


Эта импульсивная функция принимает значения +1 и 0. Используя эту функцию, можно смоделировать периодические функции, совпадающие с $a(t)$ и $f(t)$ на отрезке периода T. Пусть T=3. Тогда создадим требуемые функции и решим соответствующее ОДУ

```
Remove[t, a, f, u]; t1 = 1; T = 3;
a[t_] = 1 + Psf[t, t1, T];
f[t_] = Psf[t, t1, T];
Plot[a[t], {t, -2, 12}, PlotRange → {{-2, 12}, {0, 2.1}}]
Plot[f[t], {t, -2, 12}, PlotRange → {{-2, 12}, {0, 2.1}}]
nds = NDSolve[{u'[t] + a[t]u[t] == f[t], u[0] == 0}, u, {t, 0, 12}]
xs = u/.nds[[1]];
Plot[xs[t], {t, 0, 12}]
```



На предыдущем рисунке слева показана периодическая кусочно – постоянная функция $a(t)$, а справа – периодическая кусочно – постоянная функция $f(t)$. На следующем рисунке слева показан график решения.



В предыдущем решении время включения рубильника составляло треть времени одного периода ($t_1 = 1; T = 3$). Если положить $t_1 = 1; T = 2$, то график решения будет таким, как показано на предыдущем рисунке в середине. Для его построения в предыдущем коде вы должны параметру T присвоить другое значение, например, $T = 2$. В случае $t_1 = 3; T = 4$ график решения получится таким, как показано на рисунке справа.

4.5.14 Двухвидовая модель Вольтерра «хищник – жертва»

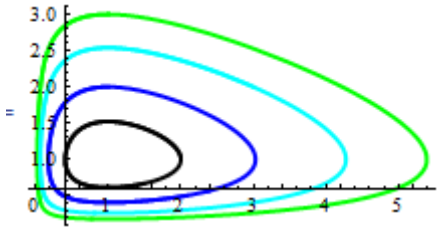
Рассмотрим двухвидовую модель «хищник – жертва», впервые построенную Вольтерра для объяснения колебаний рыбных уловов. Имеются два биологических вида, численностью в момент времени `t`, соответственно, `x(t)` и `y(t)`. Особи первого вида являются пищей для особей второго вида (хищников). Численности популяций в начальный момент времени известны. Требуется определить численность видов в произвольный момент времени. Математической моделью задачи является система дифференциальных уравнений Лотки – Вольтерра

$$\begin{cases} \dfrac{dx}{dt} = (a - b\,y)x \\ \dfrac{dy}{dt} = (-c + d\,x)y \end{cases}$$

где `a,b,c,d` – положительные константы. Проведем расчет численности популяций при $a = 3, b = 3, c = 1, d = 1$, для нескольких вариантов начальных условий: $x(0) = 2,\ y(0) = 1$; $x(0) = 1,\ y(0) = 2$; $x(0) = 1,\ y(0) = 3$; $x(0) = 3,\ y(0) = 2$. Организуем начальные условия в виде списка и построим для них фазовые траектории.

```
a = 3; b = 3; c = 1; d = 1;
sys: = {x'[t] == (a - by[t])x[t], y'[t] == (-c + dx[t])y[t]}
bc := {{x[0] == 2, y[0] == 1}, {x[0] == 1, y[0] == 2},
                    {x[0] == 1, y[0] == 3}, {x[0] == 3, y[0] == 2}}
col: = {{Black}, {Blue}, {Green}, {Cyan}}
pp = Table[se = Join[sys, bc[[i]]];
          u = NDSolve[se, {x, y}, {t, 0, 7}];
          fx = u[[1, 1, 2]];
          fy = u[[1, 2, 2]];
```

```
        p = ParametricPlot[{fx[t], fy[t]}, {t, 0, 7},
                           PlotStyle → {Thickness[0.01], col[[i]]}];
        p, {i, 1, 4}];
Show[pp, PlotRange → All]
```

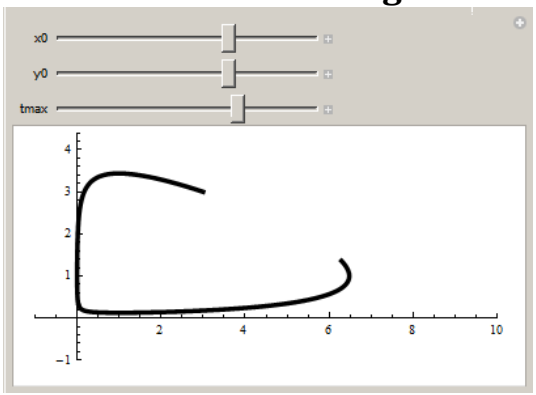


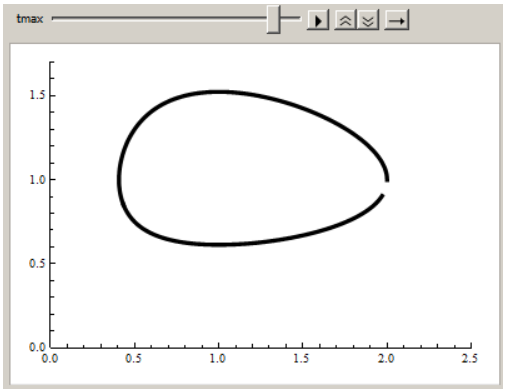
Из вида фазовых траекторий видно, что численность популяций меняется периодически. Для более детального анализа фазовых траекторий можно использовать «манипулятор» (следующий рисунок слева). В нем мы будем менять начальные значения x_0, y_0 искомых функций и интервал времени t_{max}.

```
a = 3; b = 3; c = 1; d = 1;
Module[{x, y, sys, se, u, fx, fy},
  sys: = {x'[t] == (3 − 3y[t])x[t], y'[t] == (−1 + x[t])y[t]};
  Manipulate[
    se = Join[sys, {x[0] == x0, y[0] == y0}];
    u = NDSolve[se, {x, y}, {t, 0, tmax}];
    ParametricPlot[Evaluate[{fx[t], fy[t]}], {t, 0, tmax},
                   PlotStyle → {Thickness[0.01], {Black}},
                   PlotRange → {{−1, 10}, {−1, 5}}],
    {x0, 1, 4, 0.5}, {y0, 1, 4, 0.5}, {tmax, 0.05, 7, 0.05},
    AutoAction → False, Initialization: >
    {fx: = u[[1, 1, 2]]; fy: = u[[1, 2, 2]]}]]
```

Можно построить анимацию (следующий рисунок справа)

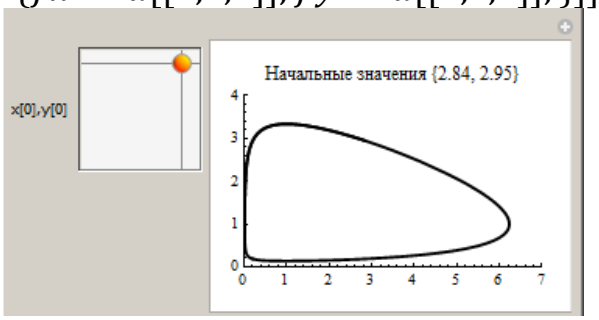
```
Animate[
    u = NDSolve[{x'[t] == (a − by[t])x[t], y'[t] == (−c + dx[t])y[t],
    x[0] == 2, y[0] == 1}, {x, y}, {t, 0, tmax}];
    fx = u[[1, 1, 2]];
    fy = u[[1, 2, 2]];
    ParametricPlot[Evaluate[{fx[t], fy[t]}], {t, 0, tmax},
                   PlotStyle → {Thickness[0.01], {Black}},
                   PlotRange → {{0, 2.5}, {0, 1.7}}], {tmax, 0.01, 4, 0.01},
                  AnimationRunning → False]
```





Можно использовать «2D манипулятор» для одновременного изменения начальных значений популяций.

```
Module[{u, se, x, y, pt, sys, fx, fy},
  a = 3; b = 3; c = 1; d = 1;
  sys = {x'[t] == (a - by[t])x[t], y'[t] == (-c + dx[t])y[t]};
  Manipulate[
    se:= Join[sys, {x[0] == pt[[1]], y[0] == pt[[2]]}];
    u = NDSolve[se, {x, y}, {t, 0, 7}];
    ParametricPlot[{fx[t], fy[t]}, {t, 0, 7},
                PlotStyle → {Thickness[0.01], Black},
                PlotRange → {{0, 7}, {0, 4}}, PlotLabel →
                  StringJoin["Начальные значения", ToString[pt]]],
    {{pt, {2, 2},x[0],y[0]}, {1, 1}, {3, 3}},
    ControlPlacement → Left, AutoAction → False,
     Initialization:> {fx:= u[[1,1,2]]; fy:= u[[1,2,2]]; }]]
```



Весь код мы включили в функцию `Module` для того, чтобы основные переменные кода, стоящие в списке первого аргумента и являющиеся локальными для модуля, не взаимодействовали с одноименными переменными блоков кода других примеров. Имеет смысл коды всех примеров обрамлять этой функцией.

Замечания о выполнении примеров

Если вы начинаете новую сессию работы с новым документом, то ввод и выполнение кода любого примера пройдет гладко так, как описано нами. Все примеры протестированы в системе Mathematica 9. Если вы работаете с версией *Mathemanica* 6, 7 или 8, то в них нет некоторых функций, которые мы описывали и использовали в примерах. Например, в прежних версиях системы нет функций `DifferentialRoot`, `DifferentialRootReduce`, `NDSolveValue`, `ParametricNDSolve`, `ParametricNDSolveValue` и некоторых связанных с ними функций. В примерах с функцией `NDSolveValue` вы можете выполнить замену на функцию `NDSolve`. Конечно придется выполнить и некоторые изменения последующего кода, поскольку результаты, возвращаемые этими функциями, используются по – разному. Коды, содержащие другие из указанных функций, вам придется существенно переработать. В версиях *Mathemanica* 4 и 5, кроме прочего, нет управляющих элементов `Manipulate` и других. Также в новых версиях

системы некоторые опции графических функций переработаны и имеют другие названия.

Более существенные замечания касаются случая, когда в одном документе вы будете выполнять множество из приведенных нами примеров. В этом случае будет возникать конфликт имен переменных, которые, в целях краткости, мы называем одинаково *x*, *y*, *t* и т.д. Если перед кодом примера вы будете добавлять команду `Remove[имя1, имя2, ...]` , где `имя1, имя2,` это имена переменных, используемых в коде, то в большинстве случаев этого достаточно для корректного выполнения текущего примера. Однако, в таком случае выполнение текущего примера может приводить к порче результатов некоторых примеров, выполненных вами ранее. Чтобы защитить код примера от влияния других примеров, его (код) следует включить в тело функции `Module` или `DynamicModule`. Первым аргументом этих функций должен быть список имен защищаемых (локализуемых в блоке) переменных, а вторым – код примера, команды которого нужно (обязательно) разделять точкой с запятой. При этом функцию `DynamicModule` нужно использовать тогда, когда в коде примера имеются динамические переменные. С целью экономии в большинстве примеров этого пособия мы эти функции не использовали.