**ЛАБОРАТОРНЫЕ РАБОТЫ ПО КУРСУ**

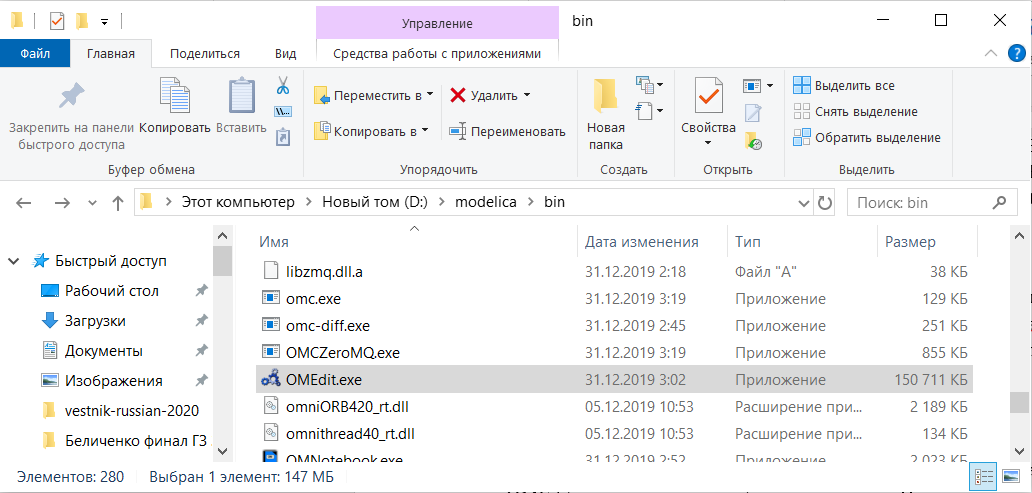
**«МЕТОДЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ДИНАМИКИ СИСТЕМ ТЕЛ: ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД»**

**Введение**

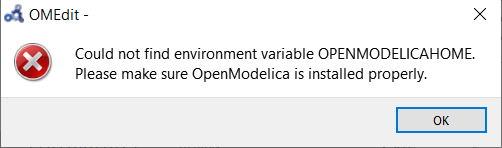
1. **Необходимое ПО для работы.**

Для выполнения лабораторных работ, являющихся предметом этого описания, потребуется скачать и установить программу «OpenModelica», имеющуюся в открытом доступе в интернете. Скачать её можно на этом сайте в разделе «Downloads» <https://openmodelica.org/> . Для работы в операционной системе Windows ссылка <https://openmodelica.org/download/download-windows> .

После установки, если программа не создала ярлыков в меню «Пуск» и на рабочем столе, войдите в папку, куда была установлена программа, далее в папку «bin», где найдите файл «OMEdit.exe», который и требуется запустить.



При запуске программа выдаст ошибку, однако, запустится и будет работать.



1. **Интерфейс программы**

Это будет описано позднее.

1. **Особенности языка Modelica**

Это будет описано позднее.

**ЛАБОРАТОНЫЕ РАБОТЫ**

1. **Построение фазового портрета системы дифференциальных уравнений второго порядка**

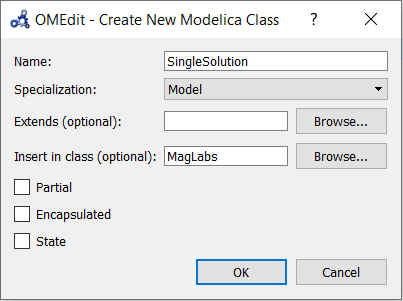
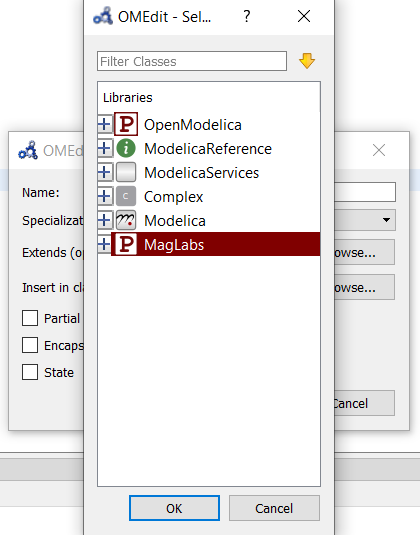
Целью первой ознакомительной лабораторной работы является численное интегрирование системы дифференциальных уравнений с помощью средств OpenModelica, и построение графиков её решений, в частности, создание фазового портрета системы.

Пусть дано дифференциальное уравнение второго порядка:

Представим его в нормальной форме Коши, введя дополнительную переменную :

и перейдём к написанию программы.

**1.1.** Напишем вначале программу, строящую одно решение данной системы уравнений. Создаём файл File -> New Modelica Class -> задаём имя «SingleSolution», оставляем тип файла «Model» и вставляем в созданную для лабораторных работ библиотеку.



model SingleSolution

equation

end SingleSolution;

В нашей модели имеются две переменные – и . Создадим в первой части программы две переменные типа Real, и зададим им начальные условия и . Во второй части программы напишем дифференциальные уравнения (производная от переменной по времени записывается с помощью оператора der( )).

model SingleSolution

Real x(start=1);

Real y(start=0);

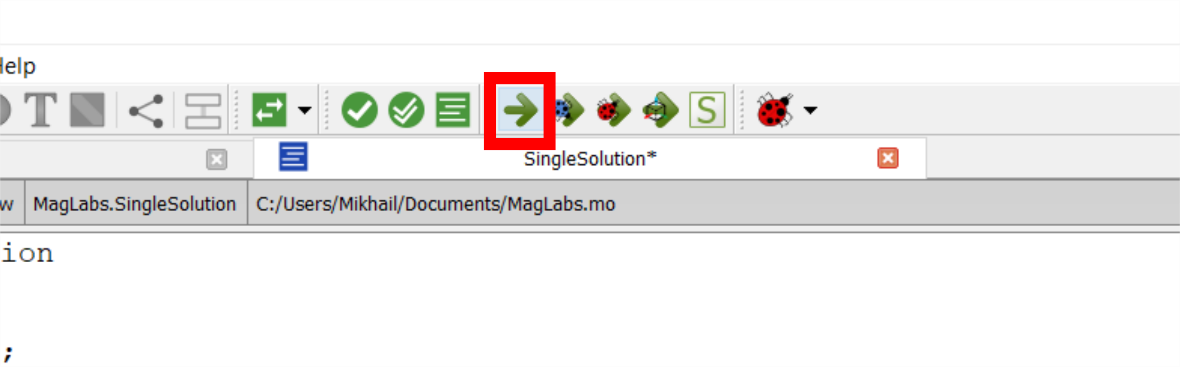
equation

der(x) = y;

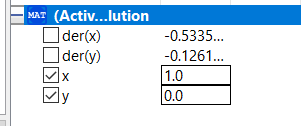
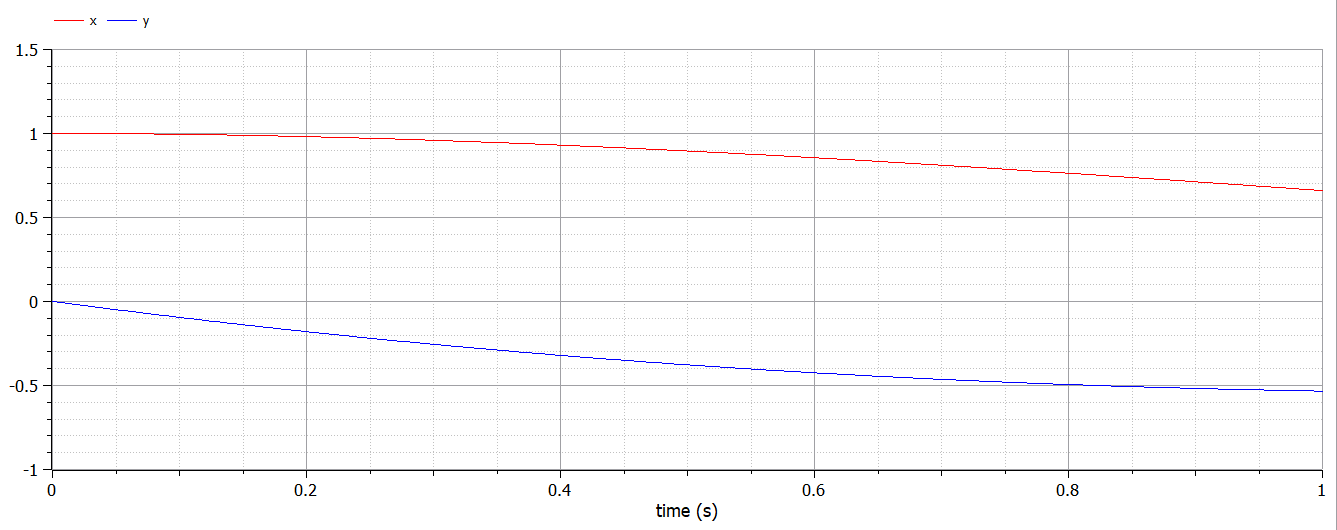
der(y) = -y - x;

end SingleSolution;

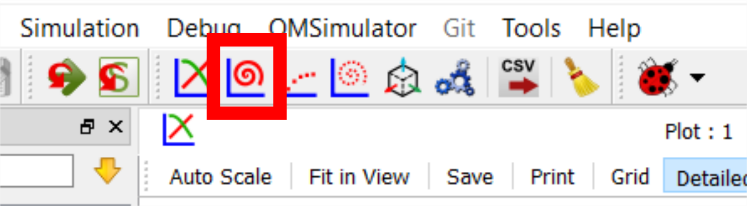
Запустим симуляцию программы.



В левом углу в списке переменных выбираем те переменные, зависимость которых от времени желаем увидеть на графике.

Чтобы получить фазовую кривую, надо создать окно параметрического графика.



После создания окна выбираем из списка переменных сначала независимую переменную, а потом зависимую. Поскольку мы хотим построить фазовую кривую , вначале выбираем , а потом .

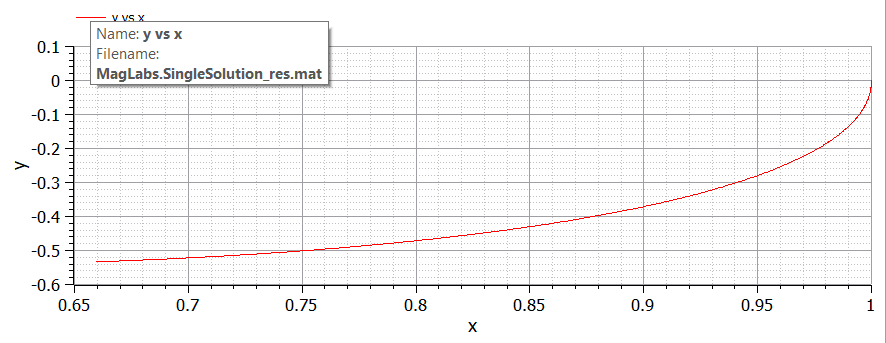
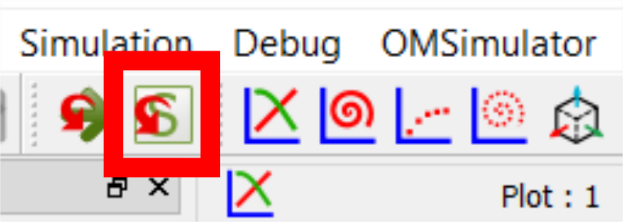
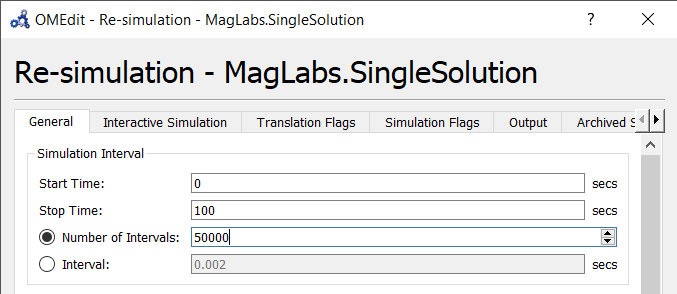


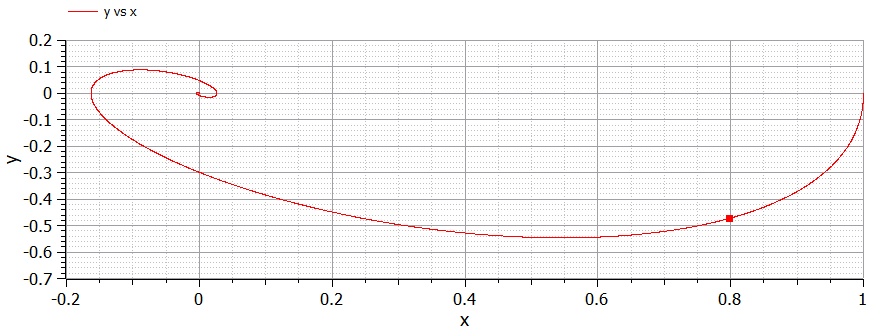
График фазовой кривой появился, однако, симуляция системы проводится по умолчанию на интервале времени от 0 до 1 сек. Изменим параметры симуляции.



Установим время от 0 до 100 и количество интервалов 50000.



Теперь фазовая кривая имеет вид:



**1.2.** Напишем теперь программу для построения множества фазовых кривых. Создадим новую модель PhasePortret. Введём новый параметр n, равный числу фазовых кривых, и объявим переменные и , как массивы с соответствующим числом элементов.

model PhasePortret

parameter Integer n = 11;

Real x[n];

Real y[n];

Пусть начальные значения переменных будут различаться, а начальные значения переменных равны нулю. Для задания начальных значений каждой кривой воспользуемся секцией «начальных уравнений» и циклом.

initial equation

for i in 1:n loop

x[i] = 0.6 - 0.1 \* i;

y[i] = 0;

end for;

Для записи системы уравнений для каждой пары переменных и также воспользуемся циклом.

equation

for i in 1:n loop

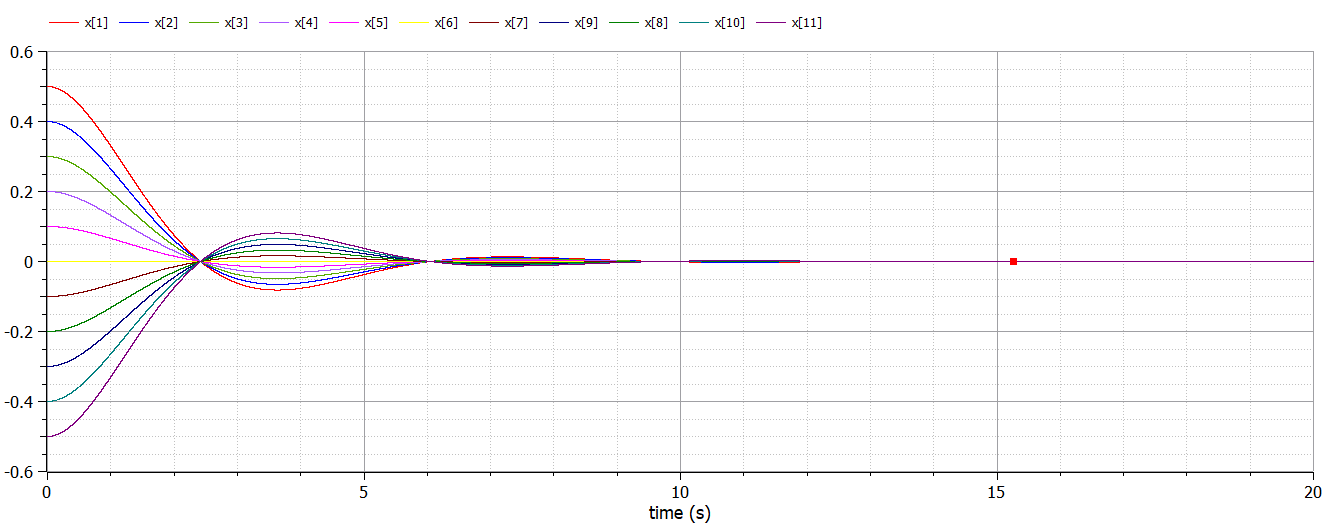
der(x[i]) = y[i];

der(y[i]) = -y[i] - x[i];

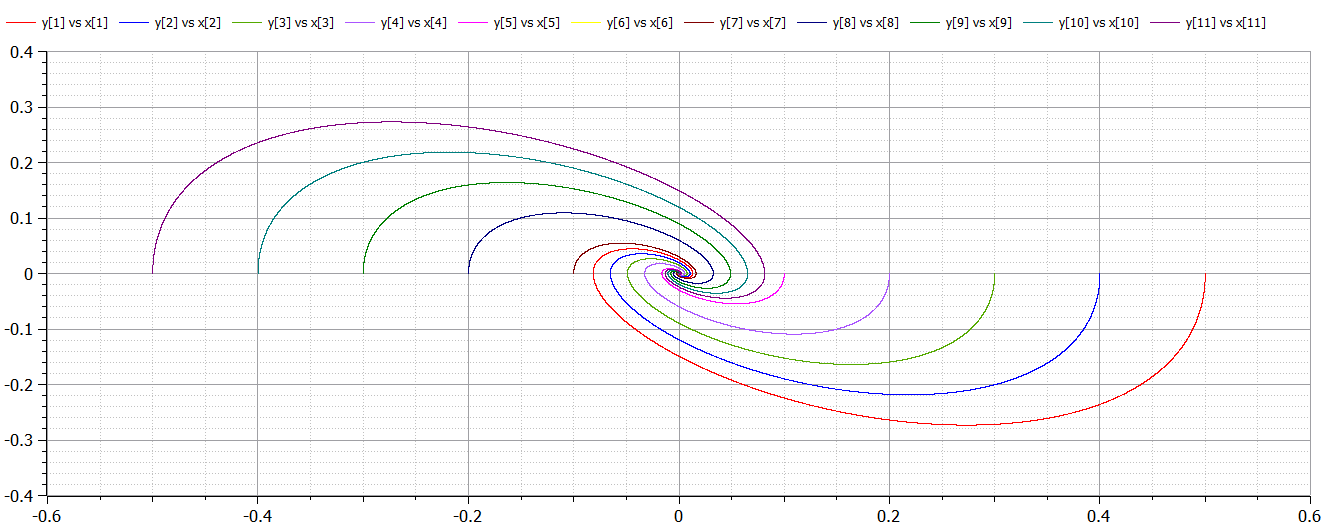
end for;

end PhasePortret;

Запустим симуляцию системы до 20 секунд и построим графики зависимостей

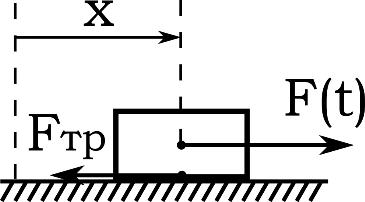


И, наконец, в окне параметрического графика попарно выбирая и , получим фазовый портрет системы.



1. **Построение модели гибридного автомата**

Целью второй лабораторной работы является построение модели гибридного автомата. Это модель, описываемая обыкновенными дифференциальными уравнениями, но имеющая способность переключаться между режимами движения в то время, как каждый из режимов движения описывается своими уравнениями.



**1.1.** Простейший пример такой системы – скольжение тела по шероховатой плоскости. Пусть на тело, двигающееся по шероховатой плоскости действует сила , вынуждающая его двигаться. Тогда уравнение движения можно записать в виде

где – масса тела, а – сила трения. Сила трения, во-первых, зависит от направления движения. Если тело двигается вправо, сила трения направлена влево и равна ( – коэффициент трения, – ускорение свободного падения), и наоборот. Во-вторых, если тело покоится, то сила трения является неизвестной величиной, меньшей значения силы трения скольжения . Таким образом, сила трения может быть задана системой

Напишем программу, реализующую данную модель. Создаём файл «FrictionModel», зададим параметры , , , и и переменные , , и .

model FrictionModel

parameter Real m=1;

parameter Real f=0.1;

parameter Real g=9.81;

parameter Real F0=1.3;

parameter Real omega=5;

Real x(start=0);

Real v(start=0);

Real F;

Real Ftr;

Запишем уравнение движения для переменных и , закон изменения и сделаем структуру из операторов «если» для определения силы трения (ВНИМАНИЕ: в каждой ветке алгоритма должно быть одинаковое количество уравнений!).

equation

der(x)=v;

m\*der(v)=-Ftr+F;

F=F0\*sin(omega\*time);

if(v>0) then

Ftr=f\*m\*g;

elseif (v<0) then

Ftr=-f\*m\*g;

else

if F<f\*m\*g and F>-f\*m\*g then

Ftr=F;

elseif F>=f\*m\*g then

Ftr=f\*m\*g;

else

Ftr=-f\*m\*g;

end if;

end if;

end FrictionModel;

Но такая программа сможет работать лишь в части значений параметров. В случае, когда тело должно встать на продолжительное время, пока сила не сможет его сдвинуть с места, симуляция программы остановится и загрузит большую часть процессора. Дело в том, что в результате особенностей численного счёта программа никогда не сможет вернуться в состояние, когда сила трения определяется вариантами 3 и 4, так как нельзя численно получить, что скорость точно равна нулю. В этом случае будет происходить «дребезжание» системы, при котором сила трения будет с огромной частотой менять своё значение между и .

Чтобы избежать данного эффекта, введём некоторую искусственную «погрешность» , и будем считать, что если скорость по модулю меньше , то она равна нулю.

Введём соответствующий параметр и исправим условия в операторах if.

model FrictionModel

parameter Real m=1;

parameter Real f=0.1;

parameter Real g=9.81;

parameter Real F0=1.3;

parameter Real omega=5;

parameter Real Vtol=0.0000001;

Real x(start=0);

Real v(start=0.2);

Real F;

Real Ftr;

equation

der(x)=v;

m\*der(v)=-Ftr+F;

F=F0\*sin(omega\*time);

if(v>Vtol) then

Ftr=f\*m\*g;

elseif (v<-Vtol) then

Ftr=-f\*m\*g;

else

if F<f\*m\*g and F>-f\*m\*g then

Ftr=F;

elseif F>=f\*m\*g then

Ftr=f\*m\*g;

else

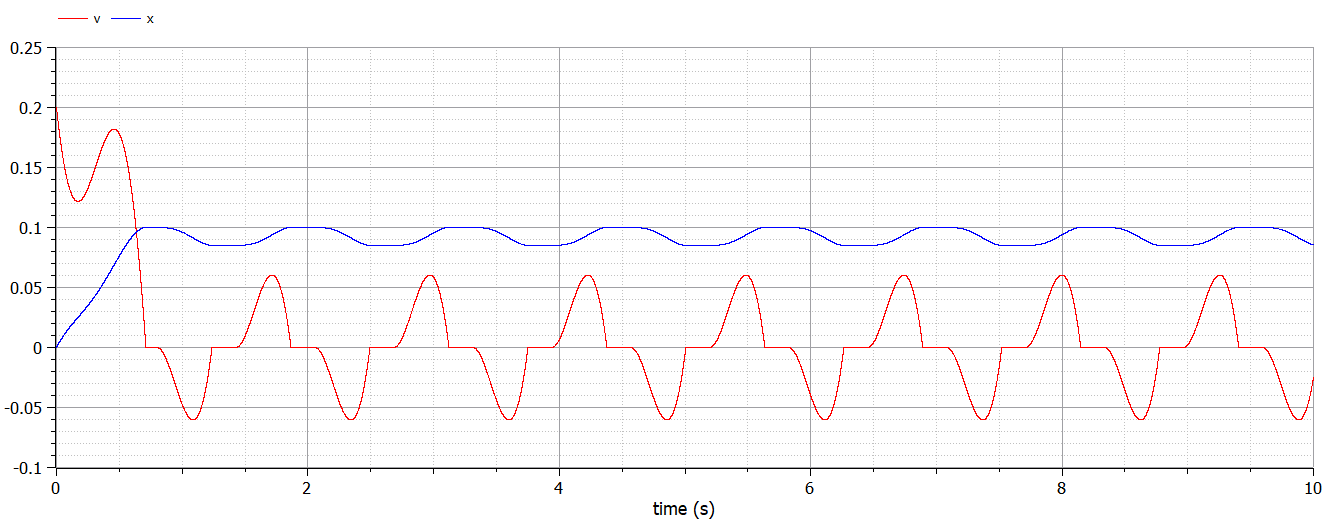
Ftr=-f\*m\*g;

end if;

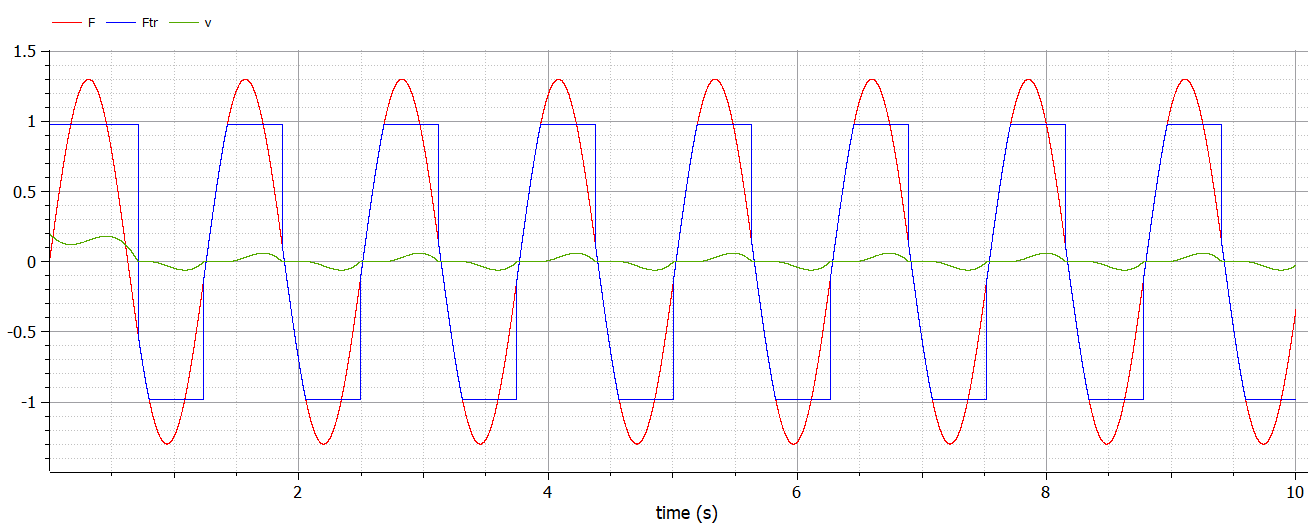
end if;

end FrictionModel;

Результаты работы программы показывают, что при таких значениях параметров тело, имея начальную скорость, выходит на стационарный режим, при котором оно совершает две длительные остановки за период изменения возмущающей силы.

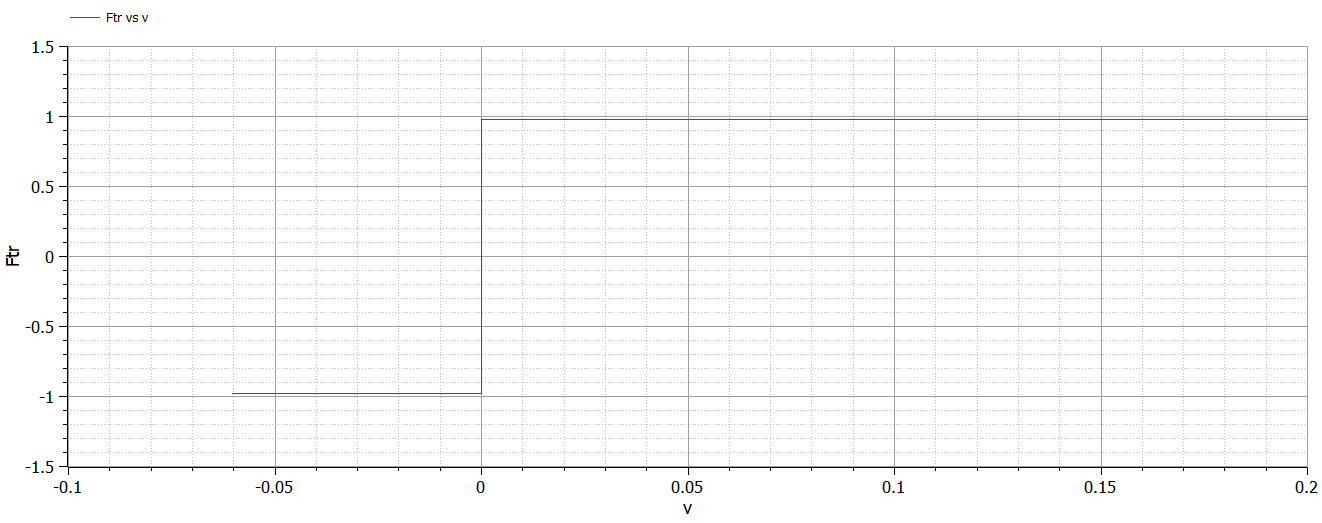


Графики силы трения и возмущающей силы с наложенным графиком скорости выглядят следующим образом.



Отсюда видно, что во время движения тела сила трения принимает наибольшее или наименьшее значение в зависимости от направления скорости движения, а в течение остановок тела становится равной возмущающей силе.

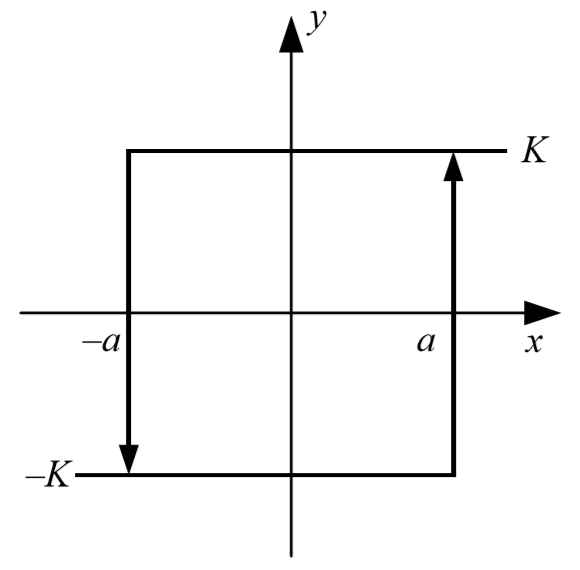
Также представляет интерес построить параметрический график зависимости силы трения от скорости тела.



Из этого графика наглядно видно, как ведёт себя сила трения в зависимости от скорости.

**1.2.** Рассмотрим теперь абстрактный пример с более сложной системой переключения. Пусть система описывается уравнением

Здесь , а зависимость задаётся системой



где и – положительные константы.

Такая схема переключения имеет неоднозначность относительно переменной на промежутке от до , и потому для её описания необходимо следить за режимом движения.

Напишем программу «HybridSystem » и зададим параметры системы , , и и переменные , , , и переменную , которая равна нулю, если система находится в состоянии, описываемом нижней прямой графика (режим 1), и переключится на единицу, когда величина превысит значение , и будет находиться в состоянии, описываемом верхней прямой (режим 2), до тех пор, пока величина не станет меньше .

model HybridSystem

parameter Real a=0.5;

parameter Real K=1;

parameter Real F0=1.3;

parameter Real omega=5;

Real x(start=0);

Real v(start=0);

Real F;

Real y;

Real flag(start=1);

Запишем уравнения движения и систему переключения значения переменной .

equation

der(x)=v;

der(v)=-y+F;

F=F0\*sin(omega\*time);

if(flag<0.5) then

if x>a then

y=K;

flag=1;

else

y=-K;

flag=0;

end if;

else

if x<-a then

y=-K;

flag=0;

else

y=K;

flag=1;

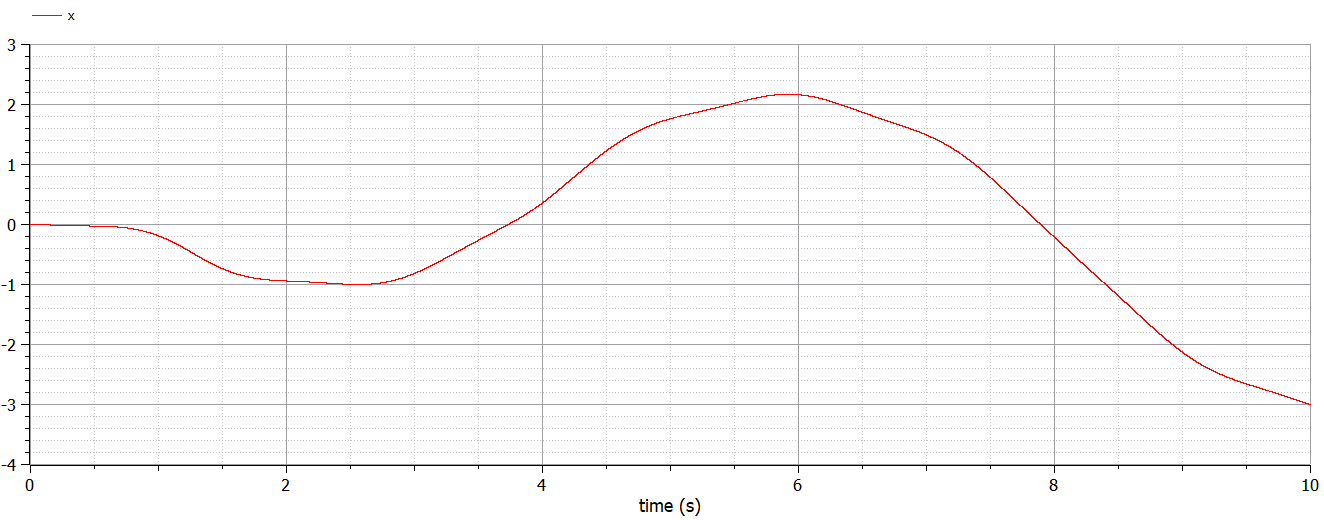
end if;

end if;

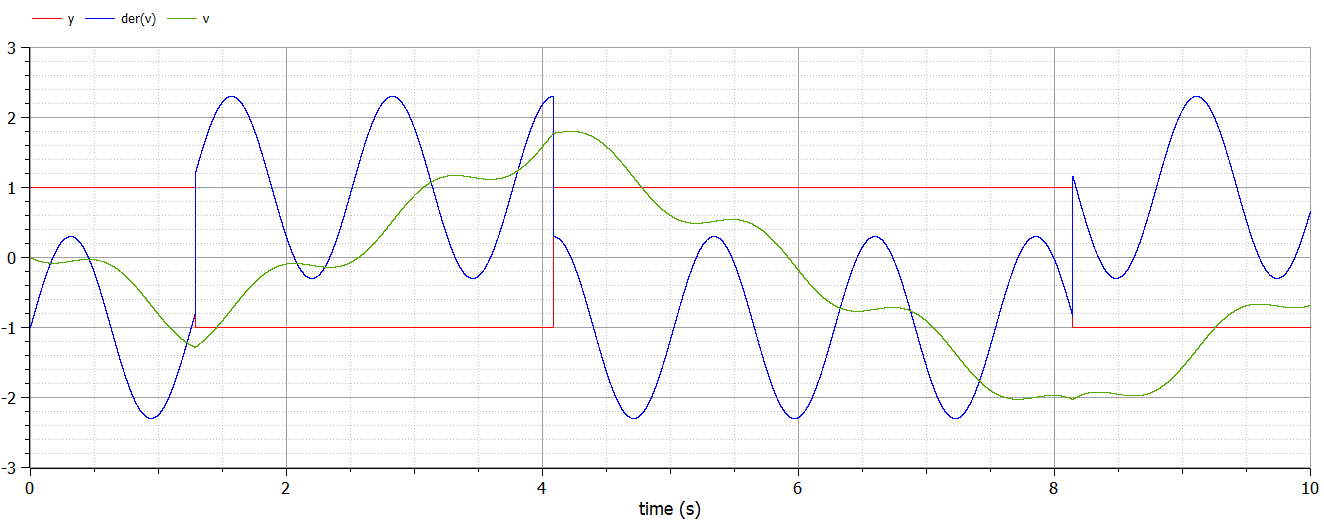
end HybridSystem;

Значение переменной сравнивается со значением 0.5 во избежание проблем с согласованием типов Real и Integer.

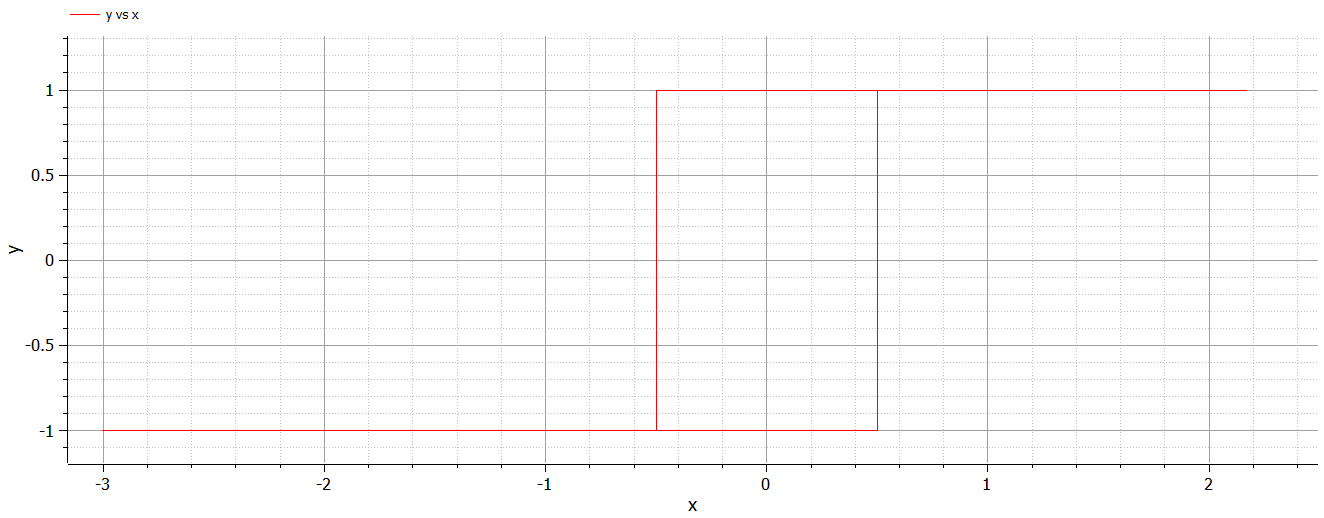
Решение системы имеет вид



Отметим, как вместе с переменной меняют свои значения величины и .



Построим параметрический график зависимости .



Отметим, что построенная зависимость совпадает с заданной изначально схемой.