>>> f = Function("f")(x) # f is a function of x

>>> # f\_ will be the derivative of f with respect to x

>>> f\_ = Derivative(f, x)

Запишем константы (которые не переменные и не функции)

m = sm.symbols('m', float=True)

k = sm.symbols('k', float=True)

**Запись дифференциальных уравнений**

Для представления уравнения *f"(x)-2f'(x)+f(x)=sin(x)* надо написать:

>>> diffeq = Eq(f(x).diff(x, x) - 2\*f(x).diff(x) + f(x), sin(x))

>>> diffeq

2

d d

f(x) - 2 ──(f(x)) + ───(f(x)) = sin(x)

dx 2

dx

**dsolve - Решение ОДУ (ODE - Ordinary differential equation)**

**dsolve** возвращает объект типа Eq , потому что в общем случае решение ОДУ.

C1; , C2; и далее - независимые константы.

>>> dsolve(diffeq, f(x))

x cos(x)

f(x) = (C1 + C2\*x)\*e + ──────

2

>>> dsolve(f(x).diff(x)\*(1 - sin(f(x))), f(x))

f(x) + cos(f(x)) = C1;

**Подстановки**

import sympy as sm

t = sm.symbols('t')

x = sm.Function('x')

print x

f1 = sm.diff(x(t), t) + x(t)

print f1 # x(t) + Derivative(x(t), t)

f2 = f1.replace(x, sm.sin)

print f2 # sin(t) + Derivative(sin(t), t)

f3 = f1.replace(x, sm.sin).doit()

print f3 # sin(t) + cos(t)

**Примеры использования**

**Касательная к графику функции (использование производной)**

Касательная к графику функции f(x) в точке х = x0 - это прямая, проходящая через точку (х0, f(x0)) и имеющая ту же производную (наклон), что функция в данной точке.

Касательная T1(x) описывается уравнением

T1(x) = f(x0) + f'(x0)(x-x0)

Найти: уравнение касательной к функции f(x) = x2/2 в точке x0 = 1.

Определим переменные и зададим функцию

>>> import sympy as sm

>>> x = sm.Symbol('x')

>>> f=sm.S('1/2')\*x\*\*2

>>> f

x\*\*2/2

Найдем производную и запишем ее в df

>>> df = f.diff(x)

>>> df

x

Определим T1, подставив x=1

>>> T\_1 = f.subs({x:1}) + df.subs({x:1})\*(x - 1)

>>> T\_1

x - 1/2

Ответ: y = x - 1/2 - это уравнение касательной к функции f(x) = f(x) = x2/2 в точке x0 = 1

Проверим, правильно ли мы нашли касательную.

Касательная к функции в точке проходит через ту же точку f(x0) = T1(x0) и имеет тот же наклон в этой точке f'(x0) = T1'(x0)

>>> T\_1.subs({x:1}) == f.subs({x:1})

True

>>> diff(T\_1,x).subs({x:1}) == diff(f,x).subs({x:1})

True

[Нарисуем графики](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html) функции и касательной в окрестности точки.

Сначала импортируем нужные пакеты

import numpy as np

import matplotlib.pyplot as plt

потом определим диапазон и зададим функции.

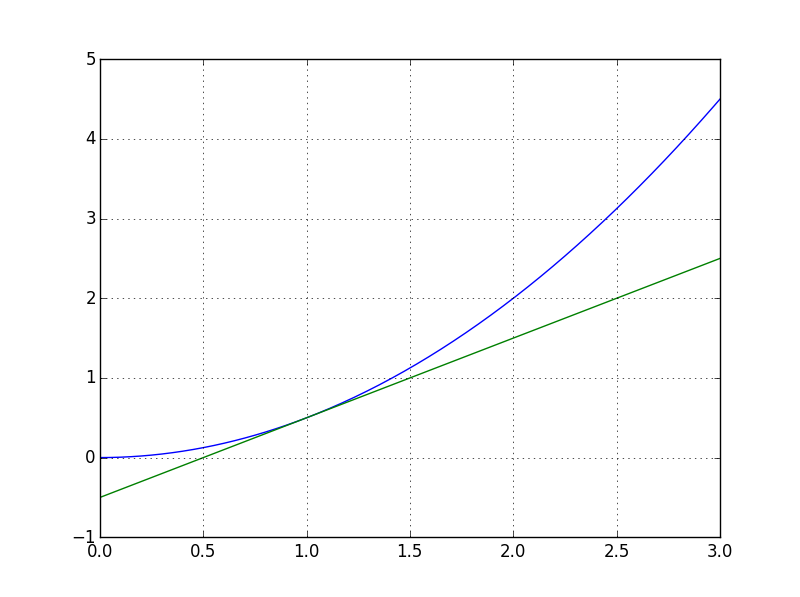
IDEA! Придумайте для переменных в рисовании графиков другие имена. Если вы использовали x в символьных вычислениях, возьмите другое имя для задания интервалов оси, например, а.

a=np.arange(0,3.01,0.01)

plt.plot(a,0.5\*a\*\*2,a,a-0.5)

plt.grid(True)

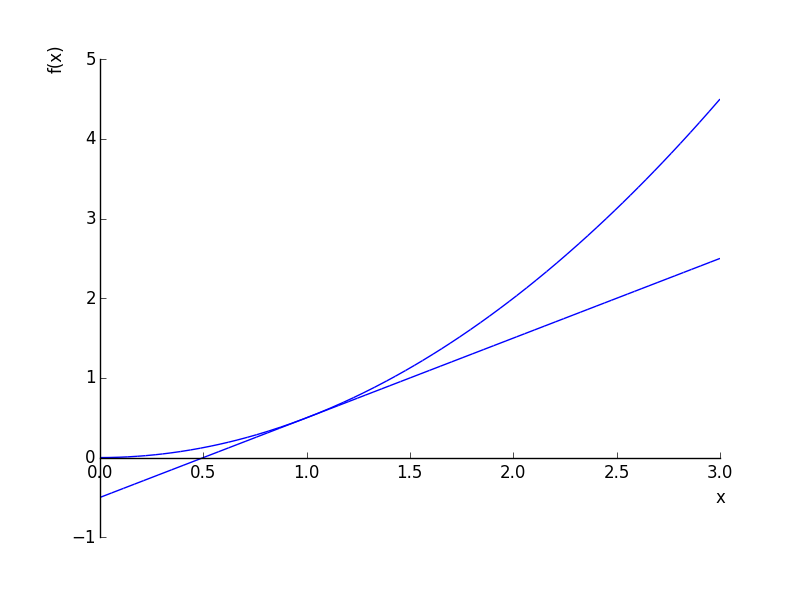
plt.show()

* график уравнения касательной к параболе:   
  

Или нарисуем этот график средствами sympy.plotting.

from sympy.plotting import plot

plot(f, T\_1, (x, 0, 3))

* касательная к функции, нарисованная в sympy:   
  

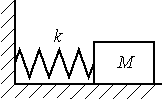
**Уравнение гармонических колебаний**

**Постановка задачи**

Дано: На пружине жесткостью k закреплен груз массой M. От положения равновесия груз отвели на расстояние L и отпустили. Трением в системе пренебречь.

Найти: Уравнение движения груза x(t)

Вычислить: х через 1 секунду от начала колебаний, для пружины жесткостью k = 5 Н/м и груза массой 1 кг, если первоначальное отклонение L = 0.1 м.

* Груз на пружине:   
  

**Физическая теория**

Считаем x=0 положение, где пружина находится в положении равновесия (не растянута и не сжата).

На груз действует сила упругости F = -kx

По второму закону Ньютона ma = F = -kx

Ускорение a - это вторая производная по времени от x(t), т.е.

mx'' = -kx

x'' + (k/m)\*x = 0

Заменим отношение k/m на w2. (Это даст требование, что w - не отрицательное).

x'' + w\*\*2 \* x = 0

**Решаем дифференциальное уравнение**

Запишем это дифференциальное уравнение и решим его.

>>> t = sm.symbols('t') # time

>>> x = sm.Function('x')

>>> w = sm.Symbol('w', positive=True)

>>> eq = sm.Eq(sm.diff(x(t), t, t) + w\*\*2 \* x(t), 0)

>>> eq

w\*\*2\*x(t) + Derivative(x(t), t, t) == 0

>>> sol = sm.dsolve(eq)

>>> sol

x(t) == C1\*sin(t\*w) + C2\*cos(t\*w)

Мы получили общее решение этого уравнения, содержащие константы С1 и С2.

**Проверяем, что решение правильное**

Проверим, что это действительно решение нашего уравнения.

IDEA! **rhs** - right hand side. Возвращает правую часть уравнения. НИКАКИХ ( ) !!!

>>> x = sol.rhs

>>> x

C1\*sin(t\*w) + C2\*cos(t\*w)

Найдем вторую производную и запишем ее в df2

>>> df2 = sm.diff(x, t, t)

>>> df2

-w\*\*2\*(C1\*sin(t\*w) + C2\*cos(t\*w))

или сразу подставим в уравнение для проверки

>>> sm.diff(x,t,t) + w\*\*2 \*x == 0

True

**Найдем константы С1 и С2 из граничных условий**

Из всего множества решений нам нужны только такие, где x(0) = L (в начальный момент времени груз отвели от положения равновесия на расстояние L.

В начальный момент времени 0 равнялась и скорость, т.е. x'(0) = 0

Запишем x(0) = L в виде T1 = 0

>>> T1 = x.subs({t:0}) - L

>>> T1

C2 - L

Запишем x'(0) = 0 в виде T2 = 0

>>> T2 = sm.diff(x,t).subs({t:0})

>>> T2

C1\*w

Решим систему уравнений T1=0 и T2=0 относительно неизвестных С1 и С2.

>>> C1, C2 = sm.symbols("C1 C2")

>>> solconst = sm.solve([T1, T2], [C1, C2])

>>> solconst

{C1: 0, C2: L}

Подставим решение в x(t)

>>> res = x.subs(solconst)

>>> res

L\*cos(t\*w)

Вспомним, что мы определяли k/m как w2

>>> m = sm.symbols('m', float=True, positive=True)

>>> k = sm.symbols('k', float=True, positive=True)

>>> w = sm.Symbol('w', positive=True)

>>> w0= sm.solve(w\*\*2 - k/m, w)

>>> w0

[sqrt(k)/sqrt(m)]

>>> w0 = w0[0]

>>> w

sqrt(k)/sqrt(m)

>>> res0 = res.subs({w:w0})

>>> res0

L\*cos(sqrt(k)\*t/sqrt(m))

Мы получили уравнение движения x(t).

Найдем значение x через 1 секунду при заданных в задаче значениях.

>>> res0.subs({t:1, L:0.1, k:5, m:1})

0.1\*cos(sqrt(5))

>>> res0.subs({t:1, L:0.1, k:5, m:1}).n()

-0.0617272876457167

**Решение в виде одного файла**

# -\*- coding: utf-8 -\*-

'''

Груз на горизонтальной пружине. Трения нет. Внешней силы нет.

'''

'''

Дано: На пружине жесткостью k закреплен груз массой M. От положения равновесия груз отвели на расстояние L и отпустили. Трением в системе пренебречь.

Найти: Уравнение движения груза x(t)

Вычислить: х через 1 секунду от начала колебаний, для пружины жесткостью k = 5 Н/м и груза массой 1 кг, если первоначальное отклонение L = 0.1 м.

Считаем x=0 положение, где пружина находится в положении равновесия (не растянута и не сжата).

На груз действует сила упругости F = -kx

По второму закону Ньютона ma = F = -kx

Ускорение a - это вторая производная по времени от x(t), т.е.

mx'' = -kx

x'' + (k/m)\*x = 0

Заменим отношение k/m на w2. (Это даст требование, что w - не отрицательное).

x'' + w\*\*2 \* x = 0

'''

import sympy as sm

from sympy.plotting import plot

t = sm.symbols('t') # time

x = sm.Function('x')

w = sm.Symbol('w', positive=True)

# запишем уравнение

# x'' + x \* w\*\*2 = 0

fun = sm.diff(x(t), t, t) + w\*\*2 \* x(t)

eq = sm.Eq(fun, 0)

print 'Equation :'

print eq

# решим уравнение в общем виде

sol = sm.dsolve(eq)

print 'Solition in general form:'

print sol

print '\n---\n'

# gf = fun.subs({x,xsol})

# print gf

x = sol.rhs

print 'sol.rhs = '

print x

print fun

# проверим, что решение найдено верно

# хочу тут не руками копи-пастить, а использовать fun и проверить - является ли решением или нет

if sm.diff(x,t,t) + w\*\*2 \*x == 0 :

print 'Check: sm.diff(x,t,t) + w\*\*2 \*x == 0 ..... OK'

print 'With begin conditions:'

# Учтем начальные условия:

# x(0) = L => x(0) - L = 0

# x'(0) = 0

# запишем начальные условия через найденные решения

L = sm.symbols('L')

T1 = x.subs({t:0}) -L

T2 = sm.diff(x, t).subs({t:0})

print T1, T2

# решим полученную систему относительно неизвестных констант C1 и C2

C1, C2 = sm.symbols("C1 C2")

solconst = sm.solve([T1, T2], [C1, C2])

print solconst

# подставим константы С1 и С2 в решение и получим ответ x(t) = ...

res = x.subs(solconst)

print 'x(t)='+str(res)

m = sm.symbols('m', float=True, positive=True)

k = sm.symbols('k', float=True, positive=True)

w = sm.Symbol('w', positive=True)

w0= sm.solve(w\*\*2 - k/m, w)

w0=w0[0]

print 'w='+str(w0)

res0 = res.subs({w:w0})

print 'x(t)='+str(res0)

# подставим численные условия задачи

# получим ответ в виде численной формулы

x = res0.subs({L:0.1, k:5, m:1})

print 'x(t)='+str(x)

print 'x(1)='+str(res0.subs({t:1, L:0.1, k:5, m:1}))

# и в виде числа

print res0.subs({t:1, L:0.1, k:5, m:1}).n()

plot(x, (t, 0, 5))

**Физические задачи**

Текст взят из задачника Филиппова по дифференциальным уравнениям. Параграф 12.

В физических зачах надо прежде всего решить, какую из величин взять за независимую переменную, а какую - за искомую функцию. Затем надо выразить, насколько изменится искомая функция y, когда независимая переменная x получит приращение Δx, т.е. выразить разность y(x+Δx) - y(x) через величины, о которых говорится в задаче. Разделив эту разность на Δx и перейдя к пределу Δx -> 0, получим дифференциальное уравнение, из которого можно найти искомую функцию. В большинстве задач содержатся условия, с помощью которых можно определить значения постоянных, входящих в общее решение дифференциального уравнения.

Иногда дифференциальное уравнение можно составить более простым путем, воспользовавшись физическим смыслом производной (если независимая переменная t, то dy/dt есть скорость изменения величины y).

Пример: В сосуд, содержащий 10 л воды, непрерывно поступает со скоростью 2 л в минуту раствор, в каждом литре которого содержится 0.3 кг соли. Поступающий в сосуд раствор перемешивается с водой и смесь вытекает с той же скоростью. Сколько соли будет в растворе через 5 минут?

Решение:

* Примем за независимую переменную время t, а за искомую функцию y(t) - количество соли в сосуде через t минут после начала опыта.
* Найдем, на сколько изменится количество соли за промежуток времени от t до t+Δt. В одну минуту поступает 2 л раствора, а в Δt минут 2Δt литров;

В этих 2Δt литрах содержится 0.3\*2Δt = 0.6Δt кг соли.

* С другой стороны за время Δt из сосуда вытекает 2Δt литра раствора.
* В момент t в сосуде (10 литрах) содержится y(t) кг соли, следовательно, в 2Δt литрах содержалось бы 2Δt/10 \* y(t) кг соли, если бы за время Δt содержание соли в сосуде не менялось.
* Но так как оно все время меняется на величину, бесконечно малую при Δt->>0, то в вытекающих 2Δt литрах содержится 0.2Δt\*(y(t)+α) кг соли с α->0 при Δt->0
* Итак, в растворе, втекающем за время Δt содержится 0.6Δt кг соли, а в вытекающем 0.2Δt\*(y(t)+α) кг соли.
* Приращение количества соли y(t+Δt)-y(t) равно разности полученных величин, т.е.  
  y(t+Δt)-y(t) = 0.6Δt - 0.2Δt\*(y(t)+α)
* Разделим на Δt и перейдем к пределу Δt->0. В левой части получится производная y'(t), а в правой 0.6 - 0.2y(t), так как Δα->0 при Δt->0.
* Итак, имеем дифференциальное уравнение y'(t) = 0.6 - 0.2y(t)
* Решая его, получим y(t) = 3 - Ce-0.2t
* Так как при t=0 соли в сосуде не было, то y(0) = 0. Применяя граничные условия найдем, что С=3.
* Подставим значение С и получим y(t) = 3 - 3e-0.2t
* При t=5 в сосуде будет y(t) = 3 - 3e-0.2\*5 = 3-3e- ~ 1.9 кг соли

# -\*- coding: utf-8 -\*-

'''

В сосуд, содержащий 10 л воды,

непрерывно поступает со скоростью 2 л в минуту раствор,

в каждом литре которого содержится 0.3 кг соли.

Поступающий в сосуд раствор перемешивается с водой и

смесь вытекает с той же скоростью.

Сколько соли будет в растворе через 5 минут?

'''

# y' = 0.6 - 0.2\*y

import sympy as sm

from sympy.plotting import plot

t = sm.symbols('t') # time

y = sm.Function('y')

# запишем уравнение

# y' = 0.6 - 0.2\*y => fun = 0

fun = sm.diff(y(t), t) - (0.6 - 0.2\*y(t) )

eq = sm.Eq(fun, 0)

print 'Equation :'

print eq

# решим уравнение в общем виде

sol = sm.dsolve(eq)

print 'Solition in general form:'

print sol

ysol = sol.rhs

print 'ysol ='

print ysol

# ysol(0) = 0

ysol.subs(t,0)

print ysol.subs(t,0)

C1 = sm.symbols("C1")

c1\_sol = sm.solve( ysol.subs(t,0), C1)

print 'C1='

print c1\_sol

y1 = ysol.subs(C1, -3)

print y1

res = y1.subs(t, 5)

print 'Соли в сосуде через 5 минут'

print res

plot(y1, (t, 0, 5))

**Система ОДУ с граничными условиями**

Решим систему ОДУ y1' = y1 + 2 y2, y2' = -2 y1 + y2 + 2 exp(t),

с начальными условиями y1(0) = y2(0) = 1

>>> import sympy as sm

>>> t=sm.symbols('t')

>>> y1=sm.Function('y1')

>>> y2=sm.Function('y2')

>>> eqs=(sm.Eq(y1(t).diff(t),y1(t)+2\*y2(t)), sm.Eq(y2(t).diff(t),-2\*y1(t)+y2(t)+2\*sm.exp(t)))

>>> s=sm.dsolve(eqs) # General solution

>>> s

[y1(t) == 2\*(C1\*sin(2\*t) + C2\*cos(2\*t))\*exp(t),

y2(t) == (2\*C1\*cos(2\*t) - 2\*C2\*sin(2\*t))\*exp(t)]

>>> y1g=s[0].args[1]

>>> y2g=s[1].args[1]

>>> # Find C1 and C2 so that the initial condition is satisfied

>>> sol=sm.solve([y1g.subs(t,0)-1,y2g.subs(t,0)-1])

>>> sol

{C1: 1/2, C2: 1/2}

>>> y1=y1g.subs(sol)

>>> y2=y2g.subs(sol)

>>> [y1,y2]

[2\*(sin(2\*t)/2 + cos(2\*t)/2)\*exp(t), (-sin(2\*t) + cos(2\*t))\*exp(t)]

[TatyanaDerbysheva](http://acm.mipt.ru/twiki/bin/view/Main/TatyanaDerbysheva) - 30 Mar 2016

* [Filippov.djvu](http://acm.mipt.ru/twiki/pub/Cintro/SymPy2/Filippov.djvu): Задачник Филиппова по дифференциальным уравнениям