Handouts/Lecture 6 (Mar 29)/Lecture 6: Differential Equations/Example: ODE & Plotting.sagews

Author	**Eugene Strakhov**
Date	**2017-03-28T19:11:52**
Project	**bf655f90-eca0-470e-9c17-2df7d93ad139**
Location	[Handouts/Lecture 6 (Mar 29)/Lecture 6: Differential Equations/Example: ODE & Plotting.sagews]
Original file	[Example: ODE & Plotting.sagews]

Визуализация решений дифференциальных уравнений

Модель динамики изменения численности популяции

Рассмотрим некоторую популяцию, имеющую достаточно пищи и ограниченной от влияния хищников. Обозначим $N(t)$ число особей в момент времени t. Тогда естественно предположить, что скорость роста числа особей пропорциональна их количеству с некоторым положительным коэффициентом r (т. е. в каждый момент времени r-я часть особей даёт потомство):

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = r \cdot N(t).$$

Это дифференциальное уравнение с разделяющимися переменными имеет решение $N(t) = e^{rt}$, т. е. мы получаем, что численность популяции растёт экспоненциально. На практике это не так, ведь особи конкурируют между собой за ресурсы на ограниченной территории. Для того, чтобы модель соответствовала реальной ситуации, введём поправочный множитель $(1 - \frac{N}{K})$, где $K > 0$:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = rN\left(1 - \frac{N}{K}\right). \tag{1}$$

При значениях N, близких к нулю, этот коэффициент близок к единице, следовательно, имеем рост численности, близкий к экспоненциальному. Если N близко к K, то поправка близка к нулю и рост популяции останавливается (в правой части уравнения стоит нуль, т. е. $N = \mathrm{const}$). Если же N превышает K, правая часть уравнения становится отрицательной и численность популяции начинает убывать. Поэтому параметр K можно интерпретировать как предельную ёмкость среды — число особей, при котором ещё возможен прирост численности популяции.

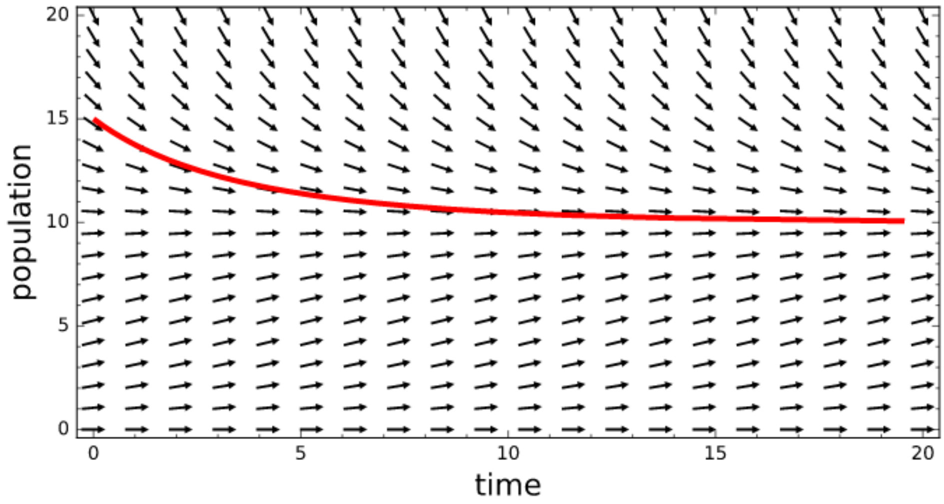
Уравнение (1) называется **логистическим уравнением**, или **уравнением Ферхюльста** (по фамилии впервые исследовавшего его бельгийского математика Pierre François Verhulst (1804–1849)). Интересно, что данное уравнение Ферхюльст получил, рассматривая модель роста численности населения.

```
t = var('t')
N = function('N')(t)
r = 0.2
K = 10
deq = diff(N, t) == r*N*(1-N/K)
N0 = 15
f = desolve(deq, N, ics=[0, N0], show_method=True, contrib_ode=True)
show(f)
```

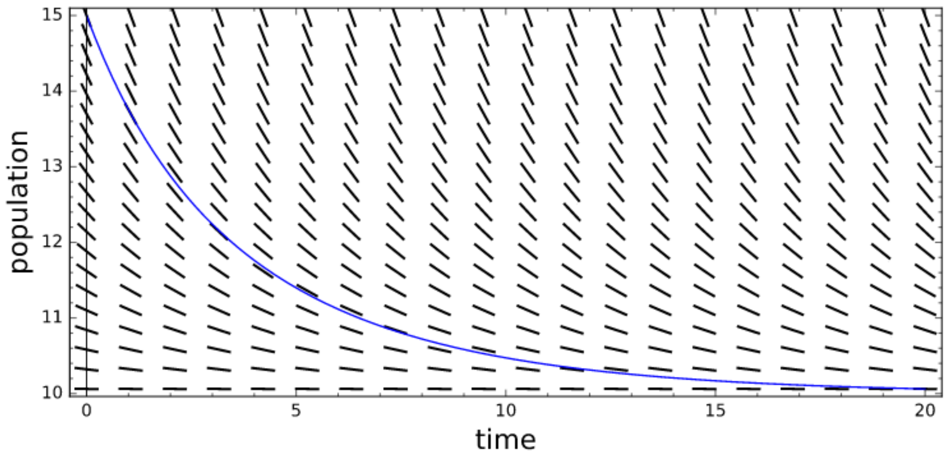
[$-5\,\log(N(t) - 10) + 5\,\log(N(t)) = t + 5\,\log(15) - 5\,\log(5)$, separable]

```
# Решение найдено в неявном виде, поэтому его график возможно построить только как implicit_plot
# Также построим поле направлений движения - plot_slope_field
y = var('y')
ff = f[0].log_simplify().substitute(N==y)
show(ff)
ip = implicit_plot(ff, (t,0,20), (y,0,20), color='red', linewidth=3)
psf = plot_slope_field(r*y*(1-y/K), (x,0,20), (y,0,20), headaxislength=3, headlength=3)
show(ip+psf, aspect_ratio=0.5, axes_labels=['time', 'population'])
```

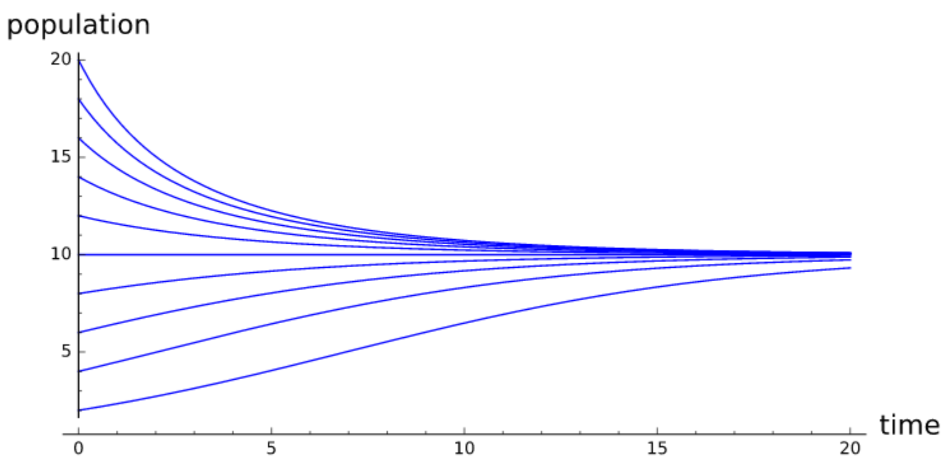
$$\log\left(\frac{y^5}{(y-10)^5}\right) = t + \log(243)$$



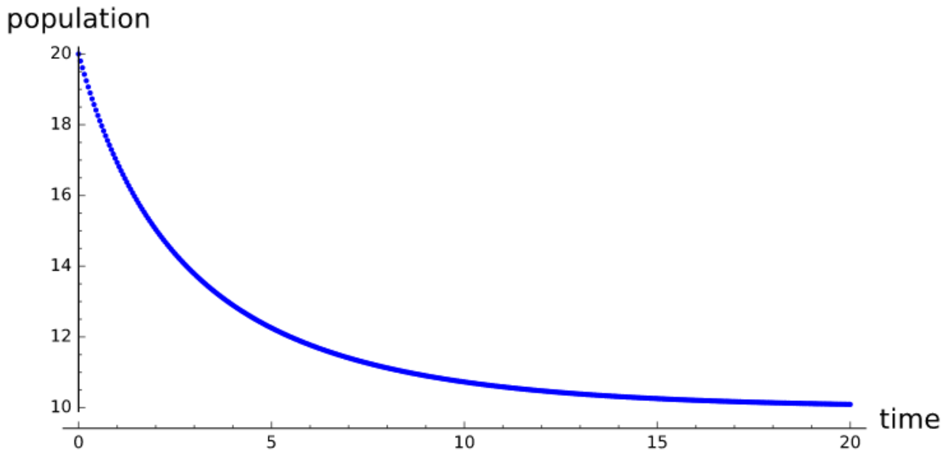

```
# Из рисунка видно, что численность популяции стремится к 10 (ёмкость среды)
# Далеко не всегда удаётся решить д. у. аналитически
# Воспользуемся численным решением
pn = desolve_rk4(deq, N, ics=[0, N0], output='slope_field', end_points=[0, 20]) # опции output: 'plot', 'lis
show(pn, axes_labels=['time', 'population'])
```




```
# Решим численно данное уравнение при различных начальных условиях: N0 = 2, 4, 6, ..., 20
pn = Graphics()
for N0 in srange(2, 20, 2, include_endpoint=True) :
    pn = pn + desolve_rk4(deq, N, ics=[0, N0], output='plot', end_points=[0, 20])
show(pn, axes_labels=['time', 'population'])
```




```
# Также можно строить график численного решения с помощью list_plot
pp = desolve_rk4(deq, N, ics=[0, 20], step=0.05, end_points=[0, 20])
lp = list_plot(pp)
lp.show(axes_labels=['time', 'population'])
```



31

```
desolve_rk4?
```

```
File: /projects/sage/sage-7.5/local/lib/python2.7/site-packages/sage/calculus/desolvers.py
Signature : desolve_rk4(de, dvar, ics=None, ivar=None, end_points=None, step=0.1, output='list', **kwds)
Docstring :
Solve numerically one first-order ordinary differential equation.
See also "ode_solver".

INPUT:

input is similar to "desolve" command. The differential equation
can be written in a form close to the plot_slope_field or desolve
command

* Variant 1 (function in two variables)

  * "de" - right hand side, i.e. the function f(x,y) from ODE
    y'=f(x,y)

  * "dvar" - dependent variable (symbolic variable declared by
    var)

* Variant 2 (symbolic equation)

  * "de" - equation, including term with "diff(y,x)"

  * "dvar" - dependent variable (declared as function of
    independent variable)

* Other parameters

  * "ivar" - should be specified, if there are more variables or
    if the equation is autonomous

  * "ics" - initial conditions in the form [x0,y0]

  * "end_points" - the end points of the interval

    * if end_points is a or [a], we integrate on between
      min(ics[0],a) and max(ics[0],a)

    * if end_points is None, we use end_points=ics[0]+10

    * if end_points is [a,b] we integrate on between
      min(ics[0],a) and max(ics[0],b)

  * "step" - (optional, default:0.1) the length of the step
    (positive number)

  * "output" - (optional, default: 'list') one of 'list', 'plot',
    'slope_field' (graph of the solution with slope field)

OUTPUT:

Return a list of points, or plot produced by list_plot, optionally
with slope field.

EXAMPLES:

   sage: from sage.calculus.desolvers import desolve_rk4
```

```
Variant 2 for input - more common in numerics:

   sage: x,y = var('x,y')
   sage: desolve_rk4(x*y*(2-y),y,ics=[0,1],end_points=1,step=0.5)
   [[0, 1], [0.5, 1.12419127424558], [1.0, 1.461590162288825]]

Variant 1 for input - we can pass ODE in the form used by desolve
function In this example we integrate bakwards, since "end_points <
ics[0]":

   sage: y = function('y')(x)
   sage: desolve_rk4(diff(y,x)+y*(y-1) == x-2,y,ics=[1,1],step=0.5, end_points=0)
   [[0.0, 8.904257108962112], [0.5, 1.909327945361535], [1, 1]]

Here we show how to plot simple pictures. For more advanced
aplications use list_plot instead. To see the resulting picture use
"show(P)" in Sage notebook.

   sage: x,y = var('x,y')
   sage: P=desolve_rk4(y*(2-y),y,ics=[0,.1],ivar=x,output='slope_field',end_points=[-4,6],thickness=3)

ALGORITHM:

4th order Runge-Kutta method. Wrapper for command "rk" in Maxima's
dynamics package.  Perhaps could be faster by using fast_float
instead.

AUTHORS:

* Robert Marik (10-2009)
```