```
from numpy import *
from matplotlib.pyplot import *

# Функция f возвращает значение правой части решаемого ОДУ
def f(u,t,lambd):
    f = lambd*u*(t - u)
    return f

# Функция f_u возвращает значение частной производной f_u
def f_u(u,t,lambd):
    f_u = lambd*(t - 2*u)
    return f_u

# Функция реализует решение ОДУ на сетке с M интервалами
# по схеме, определяемой параметром alpha
def ODESolving(t_0,T,u_0,lambd,M,alpha):
    # Определение сетки
    tau = (T - t_0)/M
    t = linspace(t_0,T,M + 1)
    # Выделение памяти под массив сеточных значений решения ОДУ
    u = zeros(M + 1)
    # Задание начального условия
    u[0] = u_0
    # Реализация схемы из семейства ROS1
    # конкретная схема определяется коэффициентом alpha
    for m in range(M):
        w_1 = f(u[m],t[m] + tau/2,lambd)\
        /(1 - alpha*tau*f_u(u[m],t[m],lambd))
        u[m + 1] = u[m] + tau*w_1.real
    return t, u

# Определение входных данных задачи
t_0 = -1.; T = 2.
u_0 = 3.; lambd = 10.

# Определение числа интервалов сетки,
# на которой будет искаться приближённое решение
M = 50

# Отрисовка решения
figure()
t, u = ODESolving(t_0,T,u_0,lambd,M,0.)
plot(t,u,'-ro',markersize=5,label='ERK1')
t, u = ODESolving(t_0,T,u_0,lambd,M,0.5)
plot(t,u,'-yo',markersize=5,label='KN')
t, u = ODESolving(t_0,T,u_0,lambd,M,1.)
plot(t,u,'-bo',markersize=5,label='DIRK1')
t, u = ODESolving(t_0,T,u_0,lambd,M,(1+1j)/2)
plot(t,u,'-go',markersize=5,label='CROS1')
title('График u(t)')
xlabel('t'); ylabel('u')
xlim((t_0,T)); ylim((0,u_0))
legend()
```

Комментарий к файлу:

Листинг программы, реализующей решение модельного жёсткого ОДУ с помощью различных схем из одностадийного семейства схем Розенброка-Ванера.