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

# Функция f возвращает значение правой части решаемого ОДУ
def f(u):
    f = u**2
    return f

# Функция f_u возвращает значение частной производной f_u
def f_u(u):
    f_u = 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,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])/(1 - alpha*tau*f_u(u[m]))
        u[m + 1] = u[m] + tau*w_1.real
    return t, u

# Определение входных данных задачи
t_0 = 0.; T = 2.; u_0 = 1.

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

# Вычисление и отрисовка решения
figure()
t, u = ODESolving(t_0,T,u_0,M,0.)
plot(t,u,'-ro',markersize=5,label='ERK1')
t, u = ODESolving(t_0,T,u_0,M,0.5)
plot(t,u,'-yo',markersize=5,label='KN')
t, u = ODESolving(t_0,T,u_0,M,1.)
plot(t,u,'-bo',markersize=5,label='DIRK1')
t, u = ODESolving(t_0,T,u_0,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((-30,30))
legend()
```

Комментарий к файлу:

Листинг программы, реализующей поиск решения задачи Коши, решение которой существует не на всём промежутке [t_0,T].