Numerical methods @ Astrophysics

Viktor Votruba & Zdeněk Janák

Motivation

During the study of astrophysics, students are often exposed to problems, which must be solved using standard numerical methods, like optimization, integration, differentiation or Monte Carlo simulation. With the power of the Python programming language and the huge amount of scientific libraries it is very often a simple task. Unfortunately, sometimes students are not able to solve the problems, due to generally poor programming skills (not only in Python). Developed lectures about numerical methods in astrophysics is the way to help them.

Optimization problem

One of the typical tasks in the analysis of the binary stars is the derivation of the spectroscopic orbital parameters from the observations, namely from the radial velocity curve fitting. Usually one can use a method known as differential corrections, but for educational purpose we used genetical algorithms, which are more robust and the results can be used as the first guess for the parameters. We used the module PyPIKAIA, Python version of the Fortran subroutine PIKAIA.

Burgers' equation

> Burgers' equation is a fundamental partial differential equation occuring in various areas of applied mathematics, such as fluid mechanics, nonlinear acoustics, gas dynamics, traffic flow. It is named for Johannes Martinus Burgers (1895–1981). -- Wikipedia

Problem 1: Inviscid Burgers'equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$

find numerical solution on the interval $x = [1,2]$
and initial conditions $u = 1$ for $x \leq 1.5$ and $u = 0$ for $x > 1.5$.

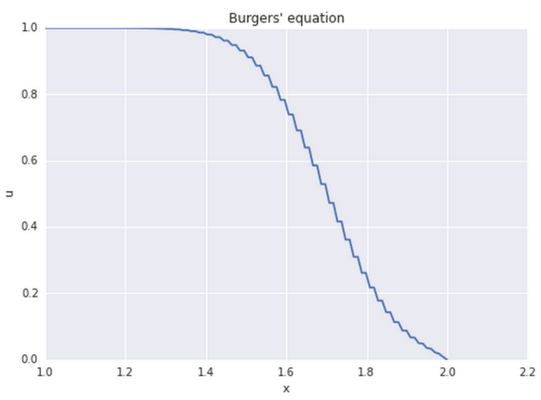
```
In [2]: import numpy as np
        from matplotlib import pyplot as plt
        from ipywidgets import interact

In [3]: def lax_method(u, dx, dt, v, time):
            """Solution using Lax method"""
            for t in range(time):
                uu = np.copy(u)
                for i in range(1, len(u)-1):
                    u[i] = 0.5 * (uu[i+1] + uu[i-1]) \
                           - 0.5 * v * dt / dx * (uu[i+1] - uu[i-1])
            return u

In [4]: @interact(dx=(1e-3, 1e-1, 1e-3), dt=(1e-3, 1, 1e-2),
                  velocity=(0.01, 1, 1e-2),  time=(0, 500))
        def plot_result(dx=0.01, dt=0.01, velocity=0.1, time=200):
            """Interactivly plot the result"""
            ## initial conditions
            x = np.linspace(1, 2, 1/dx)
            u = np.ones_like(x)
            u[x > 1.5] = 0
            ## plot the result
            plt.title("Burgers' equation")
            plt.xlabel("x")
            plt.ylabel("u")
            plt.plot(x, lax_method(u, dx, dt, velocity, time))
            plt.grid(True)
```

dx 0.01
dt 0.01
velocity 0.1
time 200

None



Feel free to download, modify and run this notebook.

Hydrodynamics in cube

Solving of the hydrodynamics equations in the various astrophysical configurations is one of the major problem in numerical astrophysics. For fast and furious computing it is necessary to use different languages instead of Python. But for simple problems and educational purposes Python can be used very well. Basic principles of the computational fluid dynamics can be illustrated on the solution of the Burgers' equation, a simple nonlinear advection equation. With the help of the interactive capabilities of the IPython Notebook, students can easily explore the influence of various parameters, like time or space steps, advective velocity or initial conditions.

Inspiration

J.R. Johansson. *Lectures on scientific computing with Python*. https://github.com/jrjohansson/scientific-python-lectures.

P. Charbonneau. *Genetic Algorithms in Astronomy and Astrophysics*. Astrophysical Journal Supplement v.101, p.309, 1995.

M. Zingale. *pyro: A teaching code for computational astrophysical hydrodynamics*.

EuroSciPy 2015 ~ 8th European Conference on Python in Science, Cambridge, UK, 26th-30th August 2015

 This project is funded by MUNI/FR/1697/2014.