Report for Worksheet 5: Fluid Dynamics

Markus Baur and David Beyer

July 13, 2020

Contents

1 The Navier Stokes Equation

This section follows the textbook by Landau and Lifshitz. The change of momentum in a fluid is described by the tensor Π_{ij} of momentum flux:

$$\partial_t \left(\rho u_i\right) = -\partial_j \Pi_{ij}. \tag{1}$$

Π_{ij} can be written as

$$\Pi_{ij} = p\delta_{ij} + \rho u_i u_j - \sigma_{ij} \tag{2}$$

where σ_{ij} is the viscous stress tensor which describes the drag force due to the viscosity of the fluid. To arrive at the incompressible Navier Stokes equation, we have to make several assumptions about σ_{ij}:

- σ_{ij} depends only on the derivates of the velocity, because a viscous drag force appears only when different parts of the fluid move at different velocities
- For a velocity field that does not change too rapidly in space we can approximate σ_{ij} as being dependent only on the first derivatives $\partial_j u_i$
- We approximate the dependence of σ_{ij} on the first derivatives $\partial_j u_i$ as linear
- σ_{ij} does not contain terms which are independent of the first derivatives $\partial_j u_i$
- σ_{ij} vanishes for a uniform rotation of the whole fluid

One can then show that the most general form of the viscuous stress tensor which fulfills all of the assumptions is

$$\sigma_{ij} = \eta \left(\partial_j u_i + \partial_i u_j - \frac{2}{3}\delta_{ij}\partial_k u_k\right) + \zeta \delta_{ij} \partial_j u_j \tag{3}$$

where η is called the dynamic viscosity and ζ is called the second viscosity. This relation is also known as the linear stress constitutive equation for fluids.

The Navier Stokes equation can be obtained by plugging Equation 3 into Equation 1.

$$\rho\left(\partial_t + \mathbf{u}\cdot\nabla\right)\mathbf{u} = -\nabla p + \nabla\cdot\left(\eta\left(\nabla\mathbf{u} + \left(\nabla\mathbf{u}\right)^{\mathrm{T}} - \frac{2}{3}\left(\nabla\cdot\mathbf{u}\right)\mathbb{I}\right) + \zeta\left(\nabla\cdot\mathbf{u}\right)\mathbb{I}\right) + \mathbf{f} \tag{4}$$

The macroscopic variables that does not appear in the incompressible Navier Stokes equation is the second viscosity ζ which describes the dissipation of energy due to compression or expansion. It is easy to see that the term containing ζ vanishes when the incompressibility condition is assumed to be true. Furthermore, this form of the Navier Stokes equation does not assume that the viscosities η and ζ are constant in space.

To show that the equation

$$\nabla\cdot\mathbf{u} = 0 \tag{5}$$

is equivalent to incompressibility, we note that the mass of the fluid is conserved. This means that there is a continuity equation[1] associated with the mass density ρ and the mass flux $\mathbf{j} = \rho\mathbf{u}$:

$$\partial_t\rho + \nabla\cdot\mathbf{j} = \partial_t\rho + \nabla\cdot\left(\rho\mathbf{u}\right) = 0. \tag{6}$$

For an incompressible flow, the mass density ρ is constant in space and time and this continuity equation reduces to

$$\nabla\cdot\mathbf{u} = 0. \tag{7}$$

2 Flow Between Two Plates: Analytical Solution

First, to determine which components of $\mathbf{u}$ are nonzero and on which variables they depend, we use the symmetries and incompressibility. Because the two parallel plates are inifinitely extended in the x- and z-directions, the system is translationally invariant in these directions. Hence, the velocity field $\mathbf{u}$ should only depend on y:

$$\mathbf{u} = \mathbf{u}(y).$$

Furthermore, there is no pressure drop, confinement or force in the z-direction, hence the velocity in the z-direction should be constant. By using a Galilei transformation, we can thus always find an inertial systems for which $u_z = 0$. Using the incompressibility condition

$$\partial_y u_y(y) = 0 \tag{8}$$

we get a linear velocity profile for u_y in the y-direction:

$$u_y(y) = a + b\cdot y. \tag{9}$$

Because of the no-slip boundary conditions at the walls, both a and b have to be zero, i.e. u_y also vanishes. We have now determined that the velocity field is of the form

$$\mathbf{u} = u_x(y)\mathbf{e}_x. \tag{10}$$

For a steady state (i.e. time-independence) in the absence of external forces, the Navier Stokes equation

$$\rho\left(\partial_t + \mathbf{u}\cdot\nabla\right)\mathbf{u} = -\nabla p + \eta\nabla^2\mathbf{u} + \mathbf{f} \tag{11}$$

[1]The total mass in any volume V can only change by a flux of mass $\mathbf{j}$ into or out of the volume through the boundary ∂V:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int\limits_V \rho\,\mathrm{d}V = -\int\limits_{\partial V}\mathbf{j}\,\mathrm{d}\mathbf{A}.$$

The right hand side can be rewritten using the divergence theorem, so we get:

$$\int\limits_V \left(\partial_t\rho + \nabla\cdot\mathbf{j}\right)\,\mathrm{d}V = 0.$$

Because we did not specify the volume V, the integrand has to be zero and we get the (local) continuity equation.

reduces to

$$\rho\left(\mathbf{u}\cdot\nabla\right)\mathbf{u} = -\nabla p + \eta\nabla^2\mathbf{u} \tag{12}$$

which for the given geometry further simplifies to (the pressure only depends on x):

$$\rho u_x(y)\underbrace{\partial_x\mathbf{u}(y)}_{=0} = \begin{pmatrix} -\partial_x p(x) + \eta\partial_y^2 u_x(y) \\ 0 \\ 0 \end{pmatrix}. \tag{13}$$

So we have reduced the problem to

$$\partial_x p(x) = \eta\partial_y^2 u_x(y). \tag{14}$$

Because the left hand side does not depend on y and the right hand side does not depend on x, both have to be constant. This means that the pressure profile is given by

$$p(x) = p_0 - \Delta p\cdot x \tag{15}$$

where Δp is the pressure drop (the negative of the pressure difference) per length. The velocity profile is given by

$$u_x(y) = -\frac{\Delta p}{2\eta}y^2 + c_1 y + c_2 \tag{16}$$

with integration constants c_i. To determine the c_i, we use the no-slip boundary conditions:

$$c_2 = 0 \tag{17}$$

$$-\frac{\Delta p}{2\eta}d_y^2 + c_1 d_y = 0 \quad\Rightarrow\quad c_1 = \frac{d_y\Delta p}{2\eta}. \tag{18}$$

This results in the velocity profile

$$u_x(y) = \frac{\Delta p}{2\eta}y\left(d_y - y\right). \tag{19}$$

To find the maximum velocity in the channel, we set the derivative with respect to y equal to zero:

$$0 = \partial_y u_x(y)\bigg|_{y=y_{\max}} = \frac{\Delta p}{2\eta}\left(d_y - 2y_{\max}\right) \tag{20}$$

$$\Rightarrow y_{\max} = \frac{d_y}{2} \tag{21}$$

This means that the maximum velocity is exactly in the middle between the two plates. The maximum velocity has a value of

$$u_x(y_{\max}) = \frac{d_y^2\Delta p}{8\eta} \tag{22}$$

and depends linearly on the pressure drop.

If there is no pressure drop but a constant force density f in the x-direction, the steady state Navier Stokes equation for this geometry becomes

$$-f = \eta\partial_y^2 u_x(y) \tag{23}$$

and the velocity profile is

$$u_x(y) = \frac{f}{2\eta}y\left(d_y - y\right). \tag{24}$$

The flow profile is still parabolic in this case and the maximum velocity is given by

$$u_x(y_{\max}) = \frac{d_y^2 f}{8\eta}. \tag{25}$$

3 Nondimensionalization and Reynolds Number

To bring the Navier Stokes equation to a dimensionless form, we rescale $\mathbf{r}$, t and $\mathbf{u}$:

$$\mathbf{r}^* \equiv \frac{\mathbf{r}}{L} \tag{26}$$

$$t^* \equiv \frac{t}{T} \tag{27}$$

$$\mathbf{u}^* \equiv \frac{\mathbf{u}}{U} \tag{28}$$

where L is a typical length scale, U is a typical velocity scale and $T = L/U$ is the corresponding time scale. Using the chain rule, the derivatives transform in the following way:

$$\partial_t = \frac{\partial}{\partial t} = \underbrace{\frac{\partial t^*}{\partial t}}_{=\frac{1}{T}} \frac{\partial}{\partial t^*} = \frac{1}{T}\partial_{t^*} \tag{29}$$

$$\nabla = \mathbf{e}_i \frac{\partial}{\partial x_i} = \mathbf{e}_i \underbrace{\frac{\partial x_j^*}{\partial x_i}}_{=\frac{\delta_{ij}}{L}} \frac{\partial}{\partial x_j^*} = \frac{1}{L}\mathbf{e}_i \frac{\partial}{\partial x_i^*} = \frac{1}{L}\nabla^*. \tag{30}$$

Plugging these substitutions into the Navier Stokes equation

$$\rho \left(\partial_t + \mathbf{u}\cdot\nabla\right)\mathbf{u} = -\nabla p + \eta \nabla^2 \mathbf{u} + \mathbf{f}, \tag{31}$$

we arrive at

$$\rho U \left(\underbrace{\frac{1}{T}}_{=\frac{U}{L}} \partial_{t^*} + \frac{U}{L}\mathbf{u}^* \cdot \nabla^* \right) \mathbf{u}^* = -\frac{1}{L}\nabla^* p + \frac{\eta U}{L^2}\nabla^{*2}\mathbf{u}^* + \mathbf{f}. \tag{32}$$

Multiplying by $L^2/U\eta$ and defining the scaled force density $\mathbf{f}^*$ and pressure p^*

$$\mathbf{f}^* \equiv \frac{L^2}{U\eta}\mathbf{f} \tag{33}$$

$$p^* \equiv \frac{L}{U\eta} p \tag{34}$$

we get dimensionless Navier Stokes equation

$$\underbrace{\frac{\rho U L}{\eta}}_{\equiv \mathrm{Re}} \left(\partial_{t^*} + \mathbf{u}^* \cdot \nabla^*\right)\mathbf{u}^* = -\nabla^* p^* + \ \nabla^{*2}\mathbf{u}^* + \mathbf{f}^*. \tag{35}$$

The Reynolds number Re is dimensionless and characterizes the fluid flow, it can be interpreted as the ratio of inertial forces and friction forces (due to viscosity). The low Reynolds number regime ($\mathrm{Re} \to 0$) is dominated by friction forces and corresponds to laminar flow. By setting $\mathrm{Re} = 0$ we get the steady-state Stokes equation which is linear and can be used to describe systems in the low Reynolds number regime like colloidal particles which are immersed in a fluid. The high Reynolds number regime ($\mathrm{Re} \to \infty$) is dominated by the nonlinear inertial term and corresponds to turbulent flow, which is an example of deterministic chaos. It is relevant for macroscopic objects like planes or cars.

4 The Lattice Boltzmann Method

The Boltzmann transport equation for the single particle distribution $f^{(1)}(\mathbf{r}, \mathbf{p}, t)$ is given by

$$\frac{\mathrm{d}}{\mathrm{d}t} f^{(1)}(\mathbf{r}, \mathbf{p}, t) = \partial_t f^{(1)} + \mathbf{v}\cdot\nabla_{\mathbf{r}} f^{(1)} + \mathbf{F}\cdot\nabla_{\mathbf{p}} f^{(1)} = \Omega(f^{(1)}) \tag{36}$$

where $\Omega(f^{(1)})$ is the collision operator. Without collisions, $f^{(1)}(\mathbf{r}, \mathbf{p}, t)$ is conserved (Liouville's theorem).

The BGK approximation consists of approximating the collision operator by

$$\Omega_{\mathrm{BGK}}(f^{(1)}) = -\frac{1}{\tau}\left(f^{(1)} - f^{(1)}_{\mathrm{eq}}\right) \tag{37}$$

where $f^{(1)}_{\mathrm{eq}}$ is the equilibrium distribution and τ the characteristic relaxation time towards equilibrium. The Boltzmann transport equation in the BGK approximation then reads

$$\partial_t f^{(1)} + \mathbf{v} \cdot \nabla_{\mathbf{r}} f^{(1)} + \mathbf{F} \cdot \nabla_{\mathbf{p}} f^{(1)} = -\frac{1}{\tau}\left(f^{(1)} - f^{(1)}_{\mathrm{eq}}\right). \tag{38}$$

In the Lattice Boltzmann method, space and time are discretized with step sizes Δx and Δt and velocities are chosen from a finite set $\{\mathbf{c}_i\}$, so the single particle distribution function becomes

$$f^{(1)}(\mathbf{r}, \mathbf{p}, t) \to f_i(\mathbf{r}, t). \tag{39}$$

The discretized Boltzmann equation/ lattice Boltzmann equation can then be written as

$$f_i(\mathbf{r} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i(\mathbf{r}, t) + \Omega\left(f_i(\mathbf{r}, t)\right) = f_i(\mathbf{r}, t) - \frac{\Delta t}{\tau}\left(f_i(\mathbf{r}, t) - f_i^{\mathrm{eq}}(\mathbf{r}, t)\right) \tag{40}$$

where the second equality holds in the BGK approximation. The lattice Boltzmann equation can be split into two parts: the collision is given by

$$f_i^*(\mathbf{r}, t) = f_i(\mathbf{r}, t) + \Omega\left(f_i(\mathbf{r}, t)\right) = f_i(\mathbf{r}, t) - \frac{\Delta t}{\tau}\left(f_i(\mathbf{r}, t) - f_i^{\mathrm{eq}}(\mathbf{r}, t)\right) \tag{41}$$

and the streaming is given by

$$f_i(\mathbf{r} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i^*(\mathbf{r}, t). \tag{42}$$

The single particle distribution function $f^{(1)}(\mathbf{r}, \mathbf{p}, t)$ describes the number of particles in the infinitesimal volume $[\mathbf{r}, \mathbf{r} + \mathrm{d}\mathbf{r}] \times [\mathbf{p}, \mathbf{p} + \mathrm{d}\mathbf{p}]$ at a given time t. The assumption that a fluid can be described by a single particle distribution function is already an approximation, the exact dynamics of a fluid consisting of N particles is described by the Lioville equation for the N-particle distribution function. A description using the single particle distribution function assumes that the fluid is dilute, such that interactions can be neglected except for collisions, which are described by the collision operator in the Boltzmann transport equation. For the Lattice Boltzmann method, space, time and velocities are discretized, this means that the distribution function is only defined at discrete values.

Advantages of LBM:

- Well suited for parallel implementation
- Well suited for simulations in complex geometries, for example porous materials
- Well suited for moving boundaries
- Well suited to simulate multiphase and multicomponent flows
- LB fluid can bea easily coupled to MD simulations

Disadvantages of LBM:

- LBM is always time-dependent and thus not very efficient to calculate steady states
- The method is memory intensive

5 Flow Between Two Plates: Numerical Solution

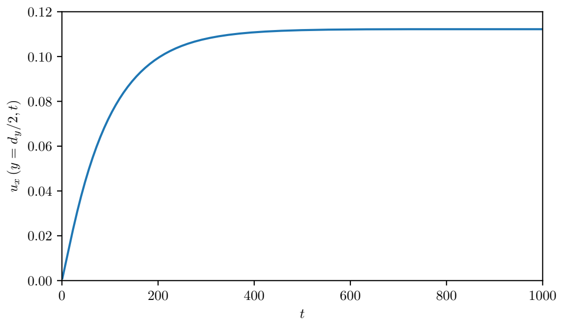
We ran Lattice Boltzmann simulations using ESPResSo for the force densities $f = 0.1$, 0.01, 0.001. The python script can be found in the appendix.

Figure 1a-Figure 1c show the time evolution of the velocity component u_x at the center of the plates for the different force densities. In all cases, the convergence to the steady state takes a time of about $t = 700$, this correponds to a total of about 70000 time steps. The python script which was used to produce the plots can be found in the appendix.

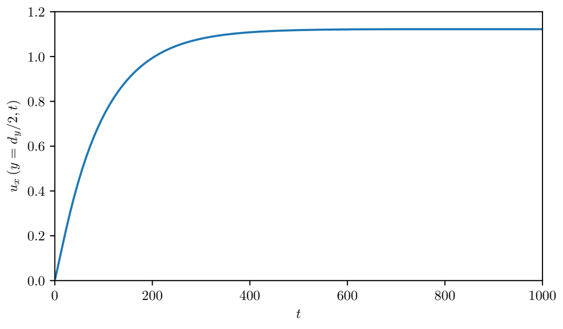
Figure 1d-Figure 1f compare the steady state velocity profile obtained from the different simulations to the analytical prediction (in the Lattice Boltzmann simulation, the walls are located between the two outermost nodes). The python script which was used to produce the plots can be found in the appendix. As the plots show, the results match quite well. The following table compares the maximum velocity which was obtained from the simulation and the analytical prediction:

force density f	max. vel. $u_x(d_y/2)$ (analytical)	max. vel. $u_x(d_y/2)$ (LBM)	error $\frac{u_{x,\text{analy.}}(d_y/2)-u_{x,\text{LBM}}(d_y/2)}{u_{x,\text{analy.}}(d_y/2)}$
0.001	0.1125	0.11220280098995868	0.264%
0.01	1.125	1.1220672418639193	0.261%
0.1	11.25	11.218233657290794	0.282%

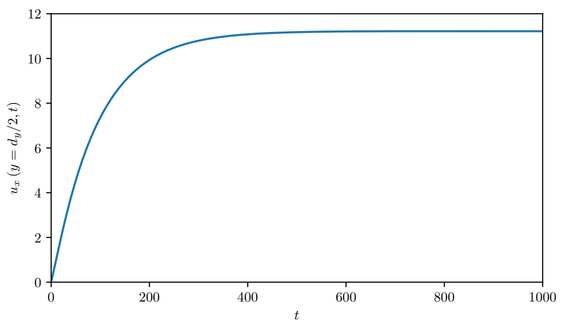
In all cases, the simulation result deviates less than 0.3% from the analytical result.



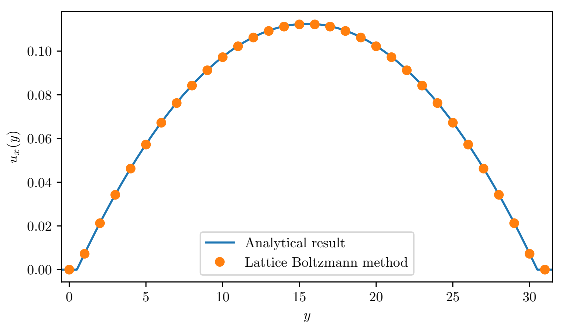
(a) Time evolution of the x-component u_x of the velocity in the middle between the plates for a force density of $f = 0.001$.



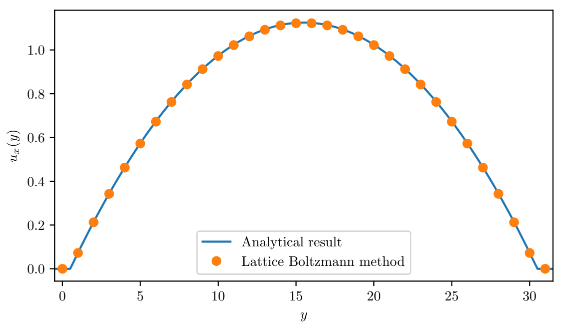
(b) Time evolution of the x-component u_x of the velocity in the middle between the plates for a force density of $f = 0.01$.



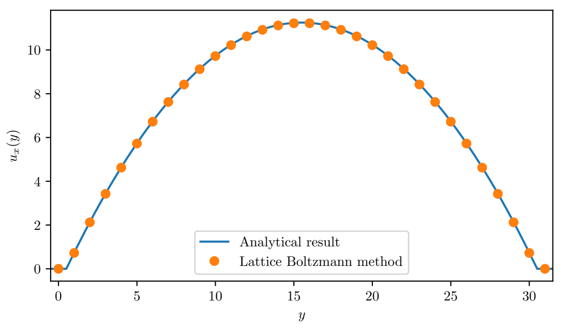
(c) Time evolution of the x-component u_x of the velocity in the middle between the plates for a force density of $f = 0.1$.



(d) Steady state velocity profile $u_x(y)$ for a force density of $f = 0.001$. Both the analytical result and the lattice Boltzmann result are shown.



(e) Steady state velocity profile $u_x(y)$ for a force density of $f = 0.01$. Both the analytical result and the lattice Boltzmann result are shown.



(f) Steady state velocity profile $u_x(y)$ for a force density of $f = 0.1$. Both the analytical result and the lattice Boltzmann result are shown.

Figure 1

6 Appendix: Python Code

Python code for the Lattice Boltzmann simulation with ESPResSo (script name espresso_LB.py):

```
import espressomd
from espressomd import shapes
from espressomd import lb
from espressomd import lbboundaries

import numpy as np
import os
import argparse
```

```
print( "␣" )
print( "====================================================" )
print( "=␣␣␣␣␣␣␣␣␣␣␣␣Lattice-Boltzmann␣Fluid␣␣␣␣␣␣␣␣␣␣␣␣␣␣=" )
print( "====================================================" )
print( "␣" )

print( "Program␣Information:␣\n" )
print( espressomd.code_info.features() )

parser = argparse.ArgumentParser(description='Simulation␣parameters')
parser.add_argument('-force', type=float,\\
help='Applied␣force␣density␣in␣the␣x-direction.', default=0.01)
parser.add_argument('-time', type=int,\\
help='Time␣for␣which␣the␣simulation␣is␣run', default=500)
args = parser.parse_args()

dir_path = os.path.dirname(os.path.realpath(__file__)) + '/velocity_'\\
+ str(args.force) + '.dat'

# geometry
box_l = 32.
padding = 1.

# fluid parameters
LB_params = {'agrid':1.,
             'dens':1.,
             'visc':1.,
             'tau':0.01,
             'ext_force_density':[args.force, 0., 0.],
             'kT':0.}

system = espressomd.System(box_l = 3*[box_l])
system.time_step = LB_params['tau']
system.cell_system.skin = 0.2

# choose between these two: GPU or CPU (depending on compilation features)
if espressomd.espressomd.cuda_init.gpu_available():
    lbf = lb.LBFluidGPU(**LB_params)
else:
    lbf = lb.LBFluid(**LB_params)


system.actors.add(lbf)

# create the boundary "shape"
upper_wall=shapes.Wall(normal=[0,1,0], dist=padding)
lower_wall=shapes.Wall(normal=[0,-1,0], dist=-(box_l-padding))

# from these shapes, define the LB boundary
upper_bound=lbboundaries.LBBoundary(shape=upper_wall)
lower_bound=lbboundaries.LBBoundary(shape=lower_wall)

system.lbboundaries.add(upper_bound)
system.lbboundaries.add(lower_bound)
```

```
#system.part.add(pos=0.5*system.box_l, type=0)

probe_ys = np.linspace(padding, box_l-padding, num = 200)

velocity_file = open("{}".format(dir_path), "w")

max_time=args.time
for t in range(max_time):
    system.integrator.run(int(1./system.time_step))

    pos = [0,system.box_l[1]/2.,0]
    vel = lbf.get_interpolated_velocity(pos)

    print("time:␣{}␣velocity:{}".format(system.time, vel))
    velocity_file.write("{}␣\t␣{}␣\n".format(system.time, vel[0]))

velocity_file.close()
outdir = ("./")
lbf.print_vtk_velocity("{}/velocity.vtk".format(outdir))
print("**Simulation␣Completed**␣")
```

Python code for plotting the time evolution of the velocity (script plot_time_evolution.py):

```
import numpy as np
import matplotlib.pyplot as plt
import os

#Set plot size
width = 5.787
height = width*0.6
plt.rc('figure', figsize=(width,height))

#Use LaTeX for fonts
plt.rc('font',**{'family':'serif','serif':['Computer␣Modern']})
plt.rc('text', usetex=True)

for f in [0.001, 0.01, 0.1]:

    dir_file = os.path.dirname(os.path.realpath(__file__)) + '/velocity_'\\
    + str(f).replace('.','') + '.dat'

    data = np.loadtxt(dir_file,unpack=False)

    plt.plot(data[:,0], data[:,1])
    plt.xlim((0.0, 1000.0))
    plt.ylim((0.0, f * 120))
    plt.xlabel(r'$t$')
    plt.ylabel(r'$u_x\left(y␣=␣d_y/2,t\right)$')
    plt.tight_layout()
    plt.show()
```

Python code for plotting the velocity profile (script name plot_profile.py):

```
import numpy as np
import matplotlib.pyplot as plt
import os

#Set plot size
width = 5.787
```

```
height = width*0.6
plt.rc('figure', figsize=(width,height))

#Use LaTeX for fonts
plt.rc('font',**{'family':'serif','serif':['Computer␣Modern']})
plt.rc('text', usetex=True)


def analytical_profile(y, f, eta, d):
    ret = f * (y-0.5) * (d - (y-0.5)) / (2 * eta)
    if y < 0.5:
        ret = 0.0
    elif y > 30.5:
        ret = 0.0
    return ret

analytical_profile = np.vectorize(analytical_profile)

y = np.linspace(-0.5, 31.5, 1000)

for f in [0.001, 0.01, 0.1]:

    dir_file = os.path.dirname(os.path.realpath(__file__)) \\
    + '/velocity_' + str(f).replace('.','') + '.vtk'

    data = np.loadtxt(dir_file,unpack=False, skiprows=10)
    profile = data[0:1024:32,0]

    plt.plot(y, analytical_profile(y, f, 1.0, 30.0),\\
    label=r'Analytical␣result')
    plt.plot(profile, linestyle='none', marker='o',\\
    label=r'Lattice␣Boltzmann␣method')
    plt.legend()
    plt.xlim((-0.5, 31.5))
    plt.xlabel(r'$y$')
    plt.ylabel(r'$u_x(y)$')
    plt.tight_layout()
    plt.show()
```