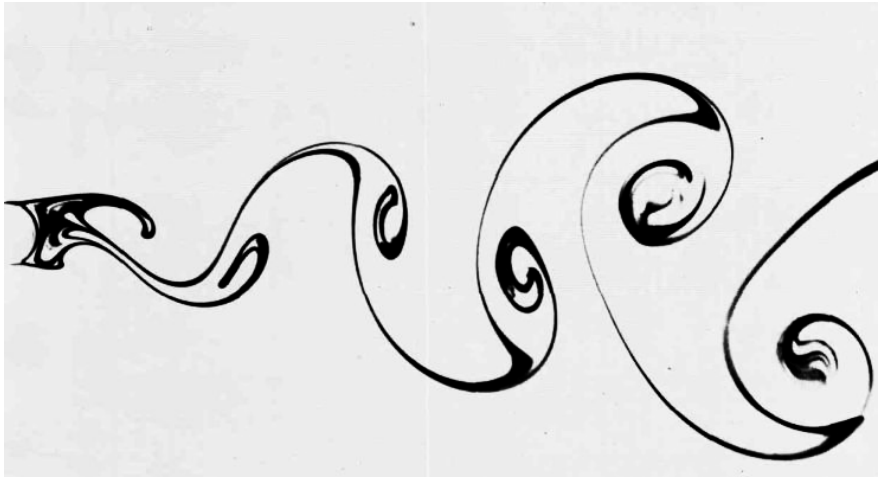
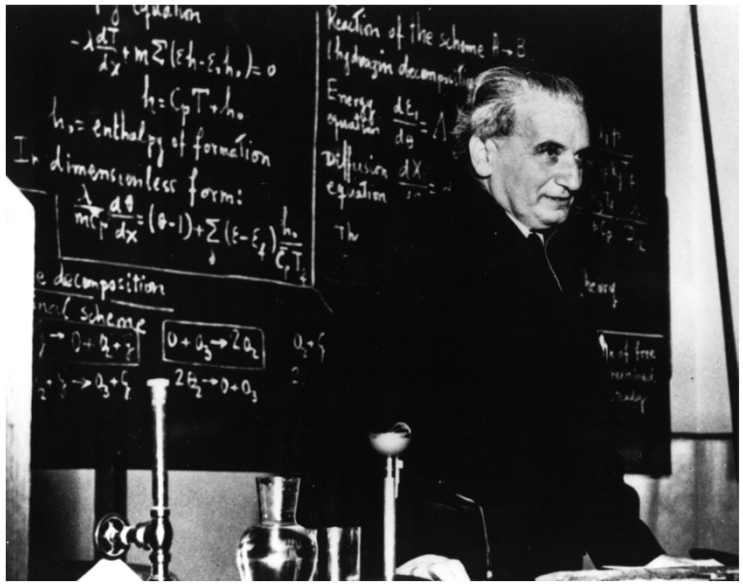
# Computer modeling of physical phenomena



Lab X – von Kármán Street

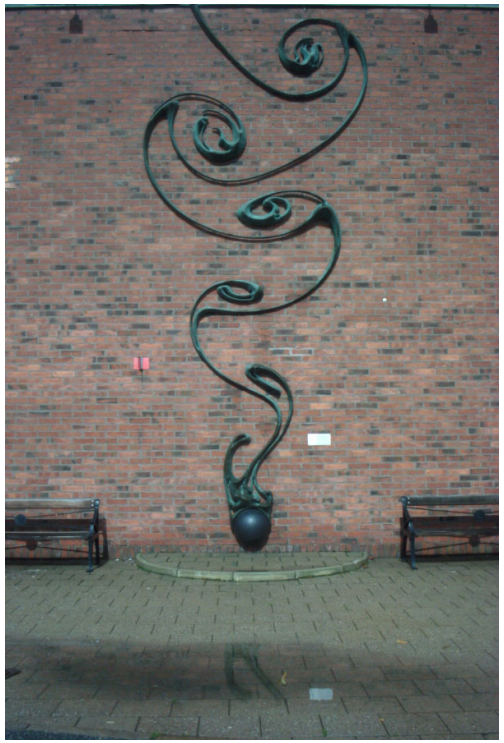
# Theodore von Kármán...



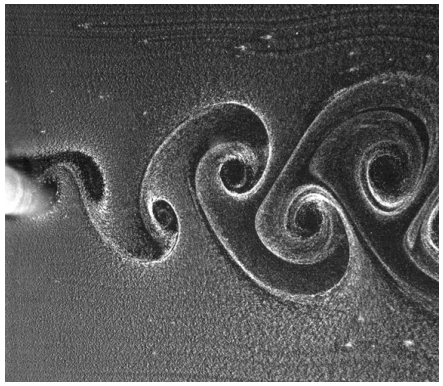

Theodore von Kármán (Tódor Kármán),
1889-1963

Judah Löwe ben Bezalel,
(1520 – 1609) and his Golem

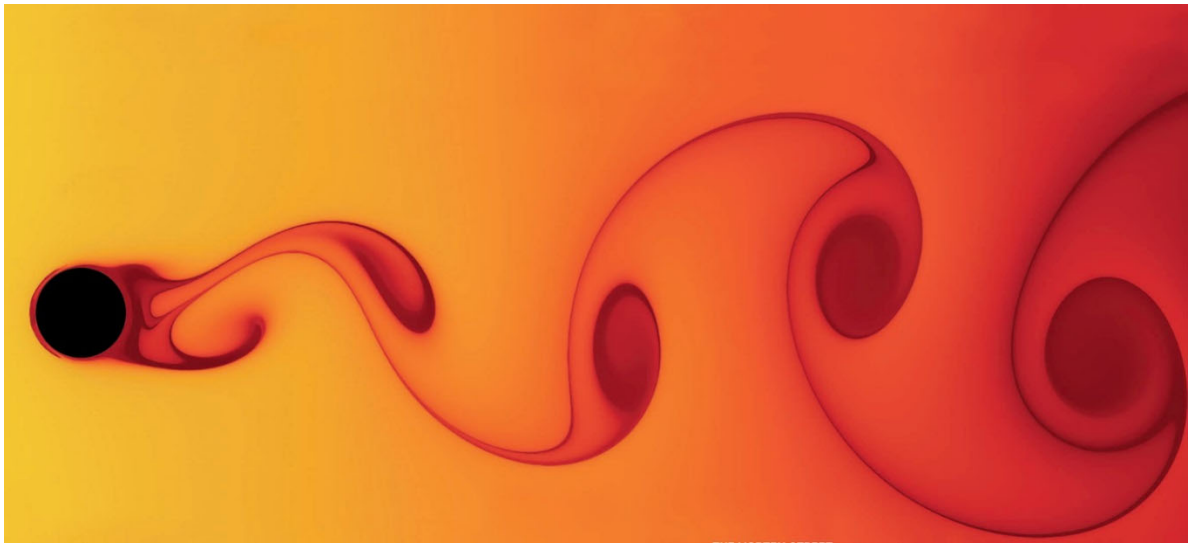
# ..and his street



KTH sculpture

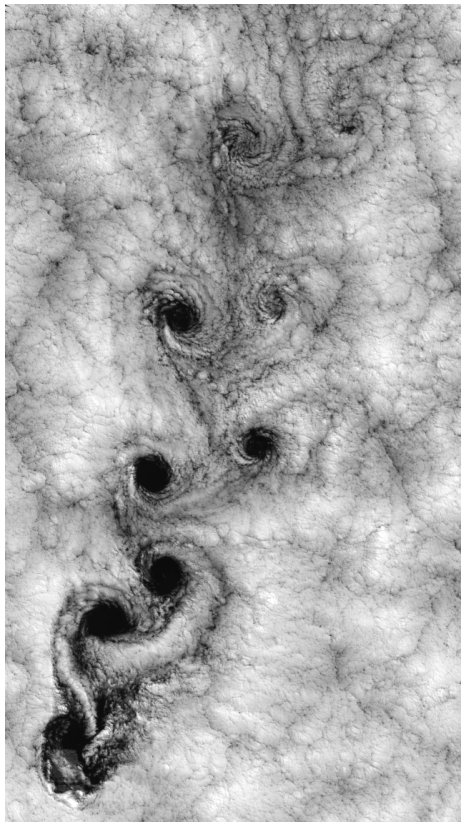


experiment: 6mm cylinder in water



simulations

# often observed behind islands

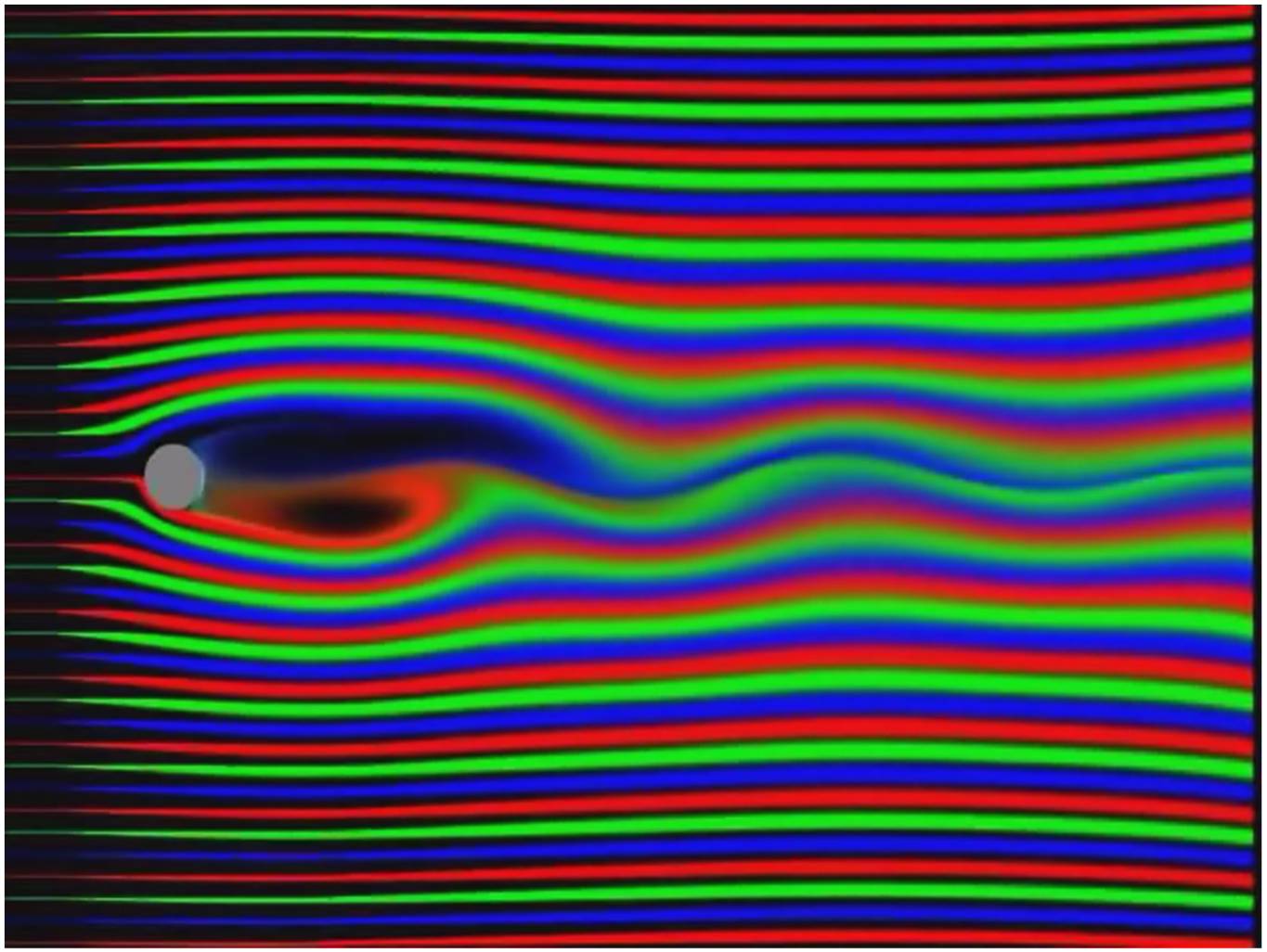


Juan Fernandez Islands

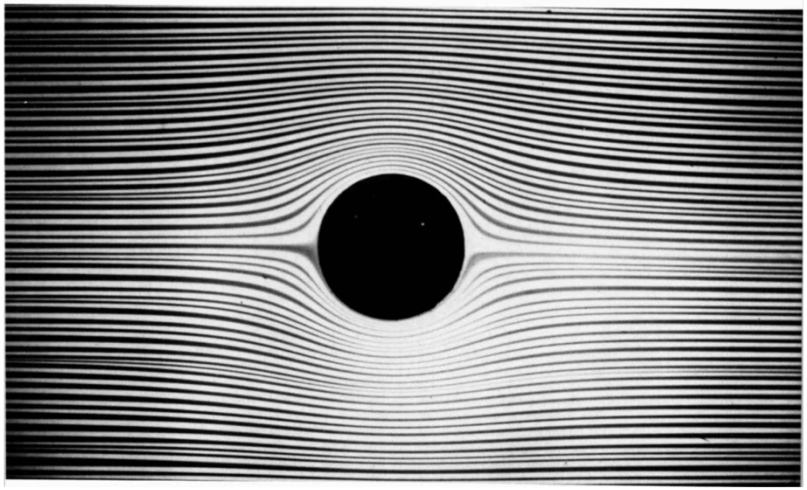
Guadalupe Islands

CAPE VERDE ISLANDS

# Vortex street video

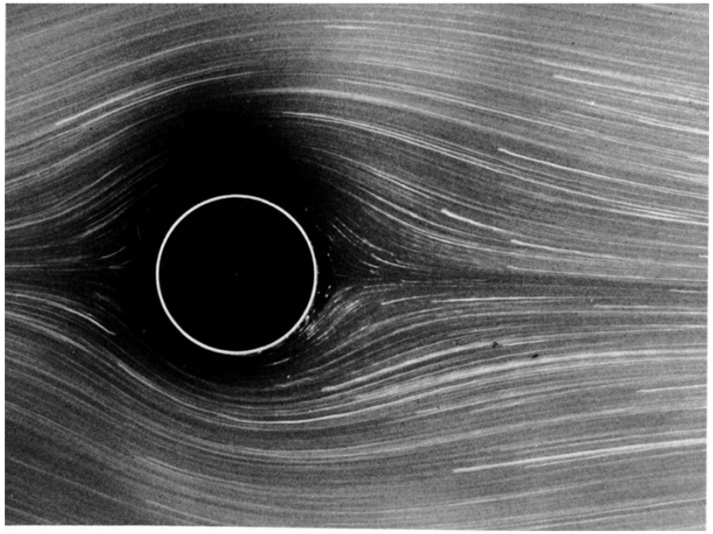


# Flow past a cylinder

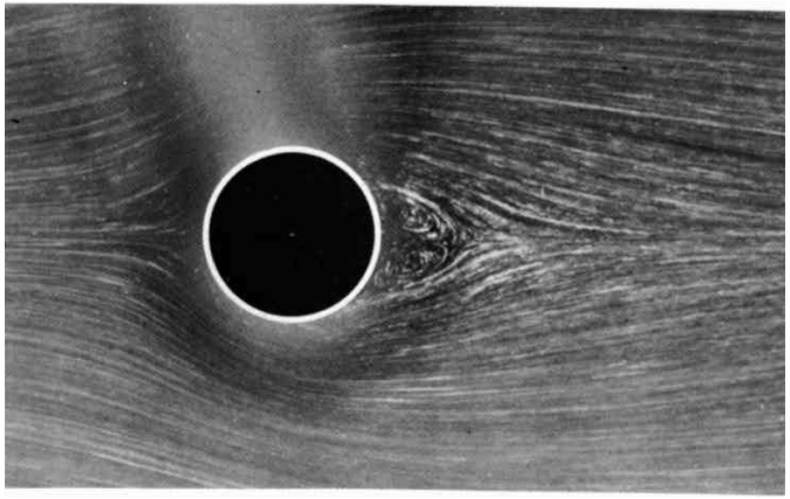


Re = $UL / \nu$

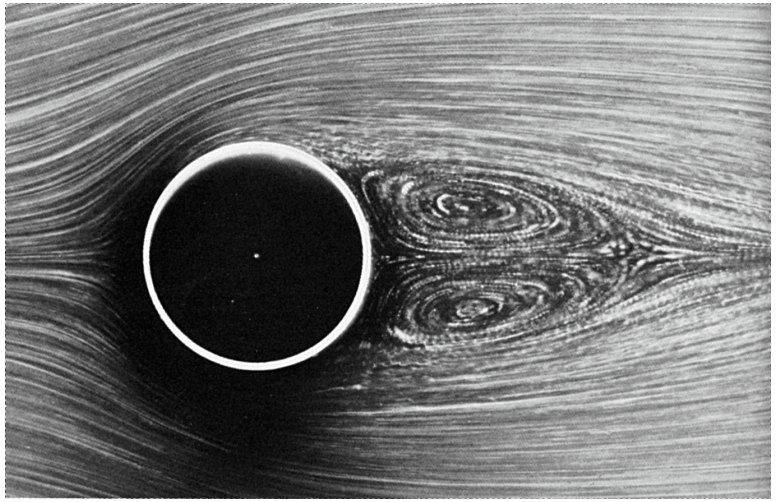
Re=0.2



Re=2.7

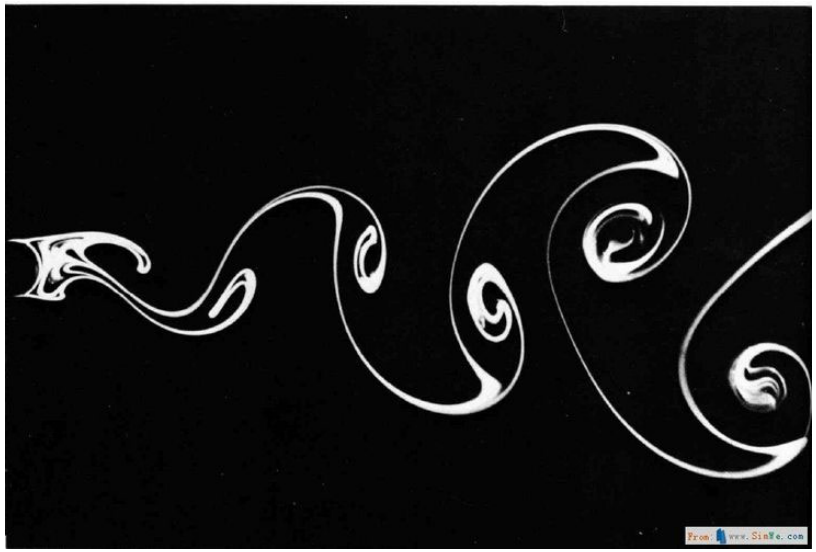


Re=13

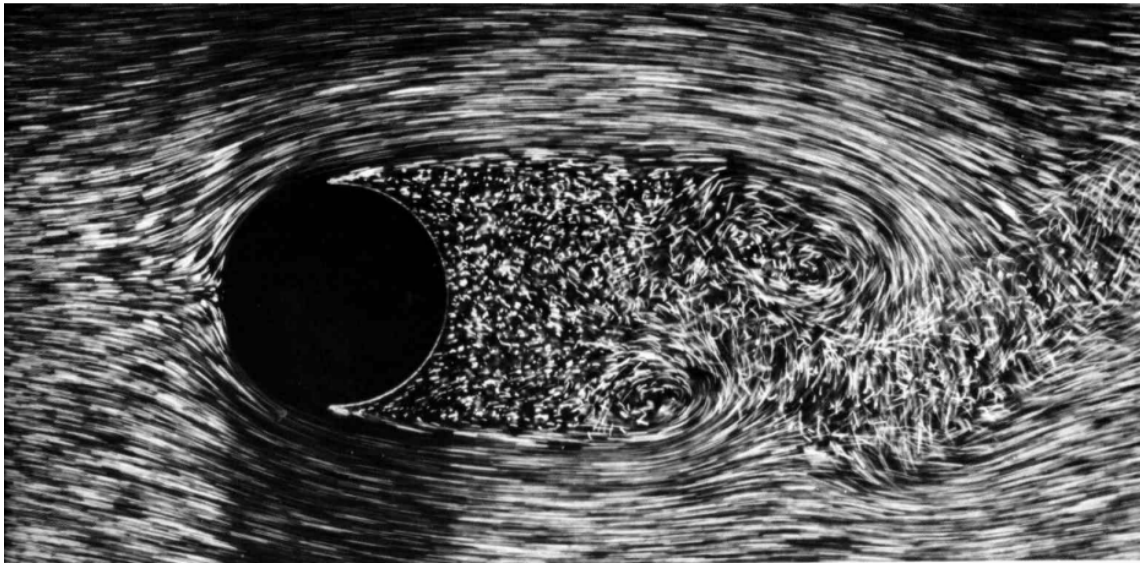


Re=26

# Flow past a cylinder

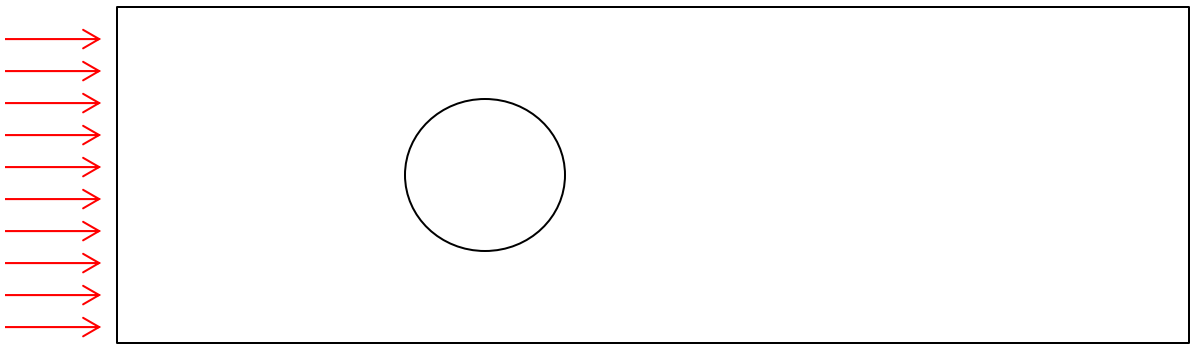


Re=140

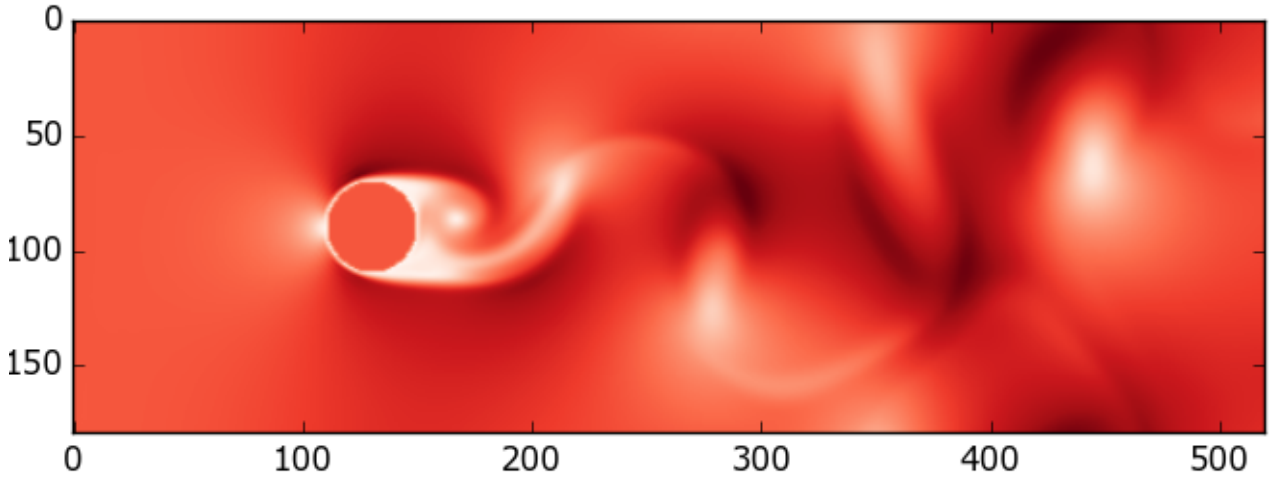


Re=2000

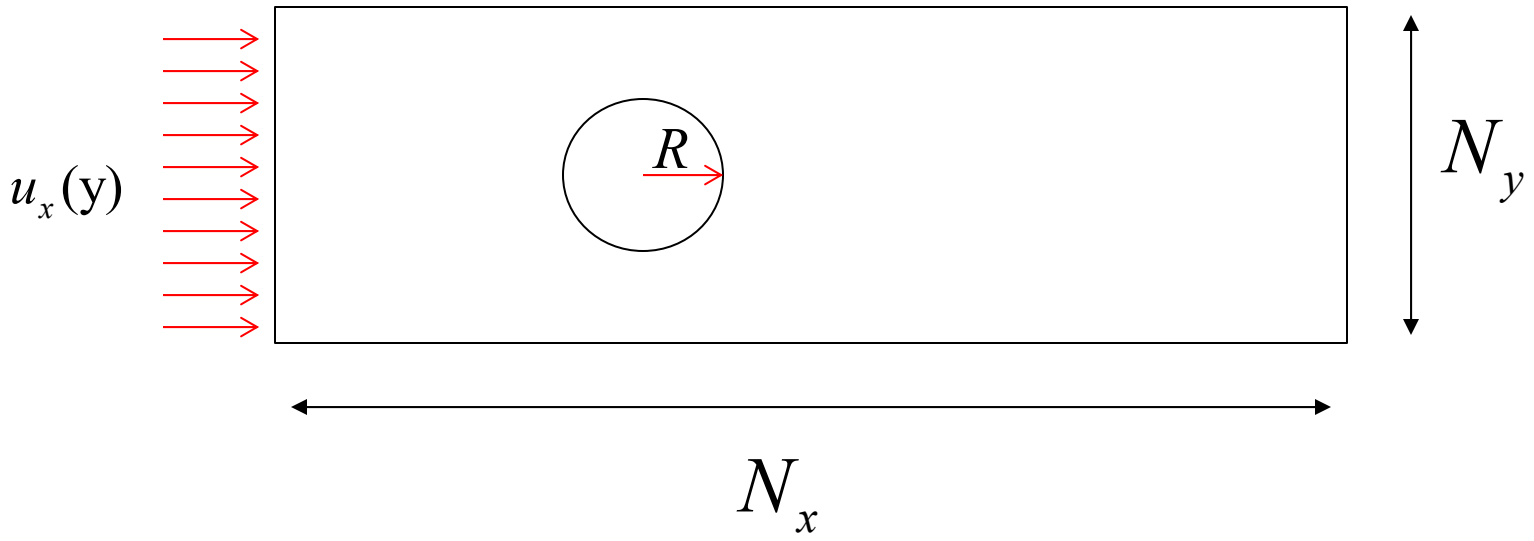
# Your task



Simulate von Kármán vortex street using LB method



# The general setup




Example parameter values:

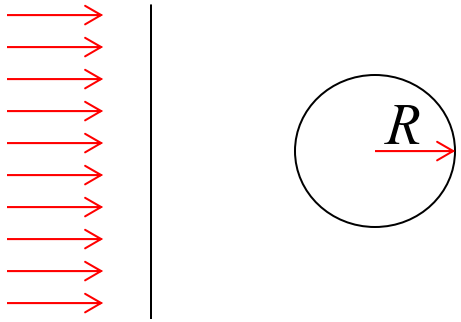
| | | | |
|---|---|---|---|
| $N_x = 520$<br>$N_y = 180$ | system size | $u_{LB} = 0.04$<br>$\mathrm{Re} = 220$ | velocity in lattice units<br>Reynolds number |
| $O_c = (\frac{N_x}{4}, \frac{N_y}{2})$ | cylinder center | $\nu_{LB} = \frac{u_{LB} R}{\mathrm{Re}}$ | viscosity |
| $R = \frac{N_y}{9}$ | cylinder radius | $\tau = 3\nu_{LB} + \frac{1}{2}$ | relaxation time |

# Inlet flow

inlet velocity

$$u_x = u_{LB}(1.0 + 10^{-4}\sin(\frac{y}{2\pi L_y}))$$

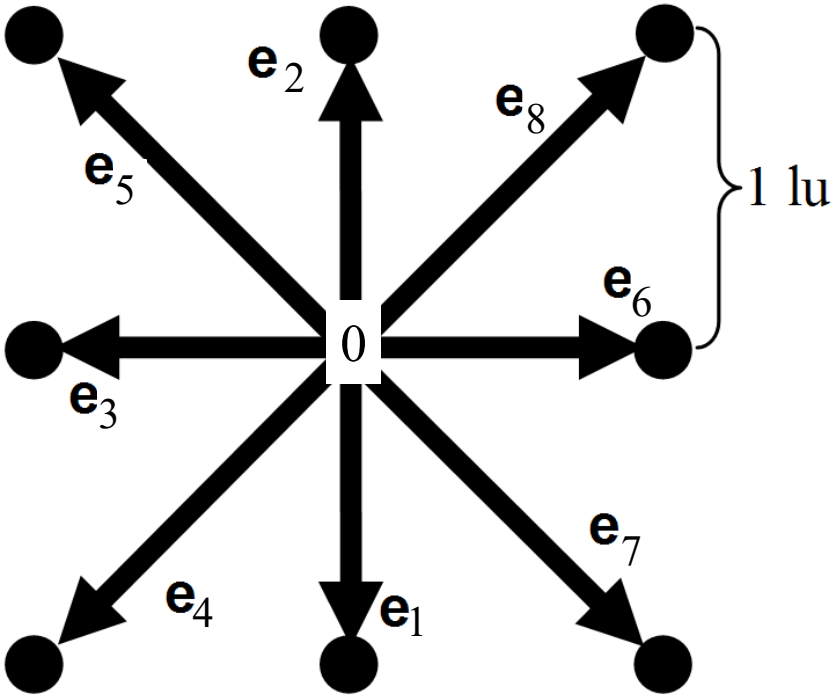
small perturbation




with $L_y = N_y - 1$

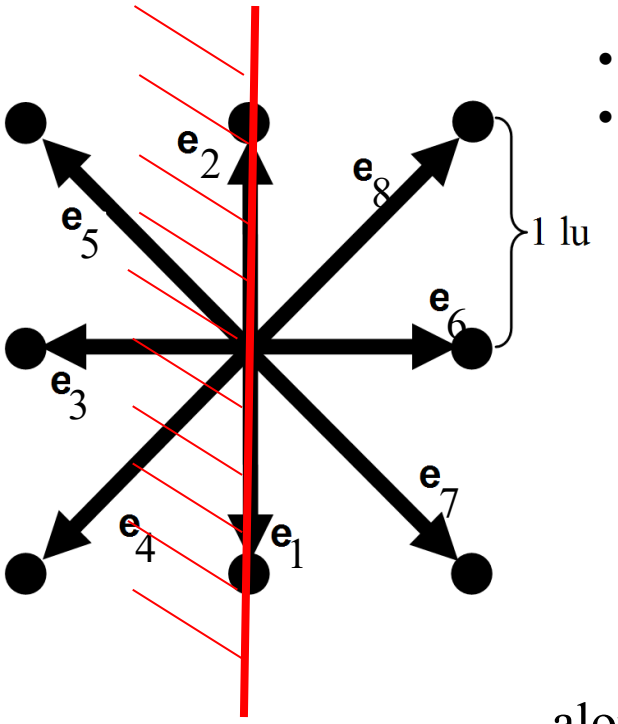
# Velocity vectors

c = np.array([(x,y) for x in [0,-1,1] for y in [0,-1,1]])




c=[[ 0,  0],[ 0, -1],[ 0,  1],[-1,  0],[-1, -1],[-1,  1],[ 1,  0],[ 1, -1],[ 1,  1]]

# Inlet boundary conditions




- the populations $f_6$, $f_7$, $f_8$ are unknown after streaming
- additionally, the density (ρ) is unknown – we need 4 eqs.

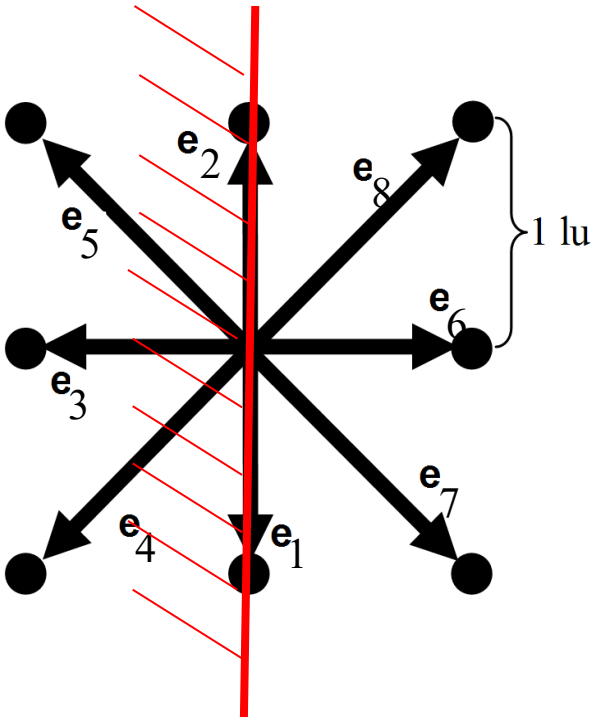
$$\rho = \sum_a f_a \qquad \rho = \sum_{a=0}^{5} f_a + f_6 + f_7 + f_8$$

$$\mathbf{u}_0 = \frac{1}{\rho}\sum_a f_a \mathbf{e}_a$$

along x: $$\rho u_0 = -f_3 - f_4 - f_5 + f_6 + f_7 + f_8$$

subtracting both: $$\rho = \frac{2\left(f_3 + f_4 + f_5\right) + \left(f_0 + f_1 + f_2\right)}{1 - u_0}$$

# Inlet boundary conditions (2)




As for the unknown populations, $f_6$, $f_7$, $f_8$ one can simply reduce them to their equilibrium values:

$$f_6 = f_6^{eq}$$

$$f_7 = f_7^{eq}$$

$$f_8 = f_8^{eq}$$

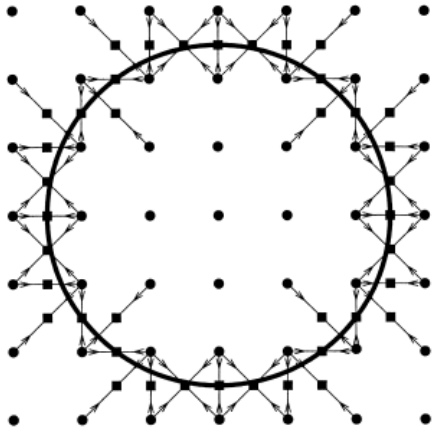
# Other boundaries

- upper & lower boundaries - periodic
- right boundary - outflow condition:

$$f_3(n_x, y) = f_3\ (n_x - 1, y)$$
$$f_4(n_x, y) = f_4\ (n_x - 1, y)$$
$$f_5(n_x, y) = f_5\ (n_x - 1, y)$$

# Bounce-back on the cylinder

c=[[ 0,  0],[ 0, -1],[ 0,  1],[-1,  0],[-1, -1],[-1,  1],[ 1,  0],[ 1, -1],[ 1,  1]]



q=9 (no. of populations)

noslip = [c.tolist().index((-c[i]).tolist()) for i in range(q)]

*[0, 2, 1, 6, 8, 7, 3, 5, 4]*

obstacle = np.fromfunction(lambda x,y: (x-cx)**2+(y-cy)**2<r**2, (nx,ny))

for i in range(q): fout[i,obstacle] = fin[noslip[i],obstacle]

(bounce back)

population matrix ($q \times N_x \times N_y$)

# Initialization

```
vel = np.fromfunction(lambda d,x,y: (1-d)*uLB*(1.0+1e-6*sin(y/ly*2*pi)),(2,nx,ny))
feq = equilibrium(1.0,vel); fin = feq.copy()
```

initialize the entire system with the inlet flow, and the populations
with the equilibrium distribution with unit density

# Collision step

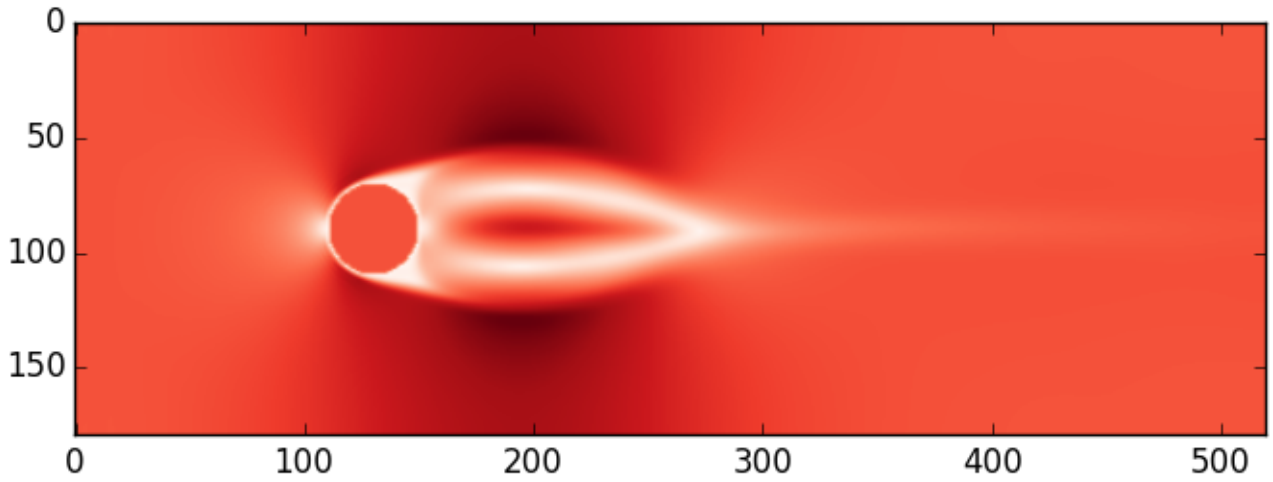
```
fout = fin - (fin - feq)/tau
```

# Streaming

```
for i in range(q):
        fin[i,:,:] = roll(roll(fout[i,:,:],c[i,0],axis=0),c[i,1],axis=1)
```

# Visualization

```
if (time%100==0):
        plt.clf(); plt.imshow(sqrt(u[0]**2+u[1]**2),cmap=cm.Reds)
        plt.savefig("vel."+str(time/100).zfill(4)+".png")
```




Run for 200000 timesteps (2000 frames)

# Extra task

- Generalize the code so that it will take any obstacle supplied by the user

For example: