

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% cylinder.m: Channel flow pas a cylindrical obstacle, using a LB method
%
% Copyright (c) 2006, Jonas Latt
%
% This program is released under the GNU General Public License (GPL)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear

% GENERAL FLOW CONSTANTS
lx = 250;
ly = 51;
obst_x = lx/5+1; % position of the cylinder; (exact
obst_y = ly/2+1; % y-symmetry is avoided)
obst_r = ly/10+1; % radius of the cylinder
uMax = 0.02; % maximum velocity of Poiseuille inflow
Re = 100; % Reynolds number
nu = uMax * 2.*obst_r / Re; % kinematic viscosity
omega = 1. / (3*nu+1./2.); % relaxation parameter
maxT = 400000; % total number of iterations
tPlot = 5; % cycles

% D2Q9 LATTICE CONSTANTS
t = [4/9, 1/9,1/9,1/9,1/9, 1/36,1/36,1/36,1/36];
cx = [ 0, 1, 0, -1, 0, 1, -1, -1, 1];
cy = [ 0, 0, 1, 0, -1, 1, 1, -1, -1];
opp = [ 1, 4, 5, 2, 3, 8, 9, 6, 7];
col = [2:(ly-1)];

[y,x] = meshgrid(1:ly,1:lx);
obst = (x-obst_x).^2 + (y-obst_y).^2 <= obst_r.^2;
obst(:,[1,ly]) = 1;
bbRegion = find(obst);

% INITIAL CONDITION: (rho=0, u=0) ==> fIn(i) = t(i)
fIn = reshape( t' * ones(1,ly*lx), 9, lx, ly);

% MAIN LOOP (TIME CYCLES)
for cycle = 1:maxT

    % MACROSCOPIC VARIABLES
    rho = sum(fIn);
    ux = reshape ( ...
        (cx * reshape(fIn,9,ly*lx)), 1,ly,ly) ./rho;
    uy = reshape ( ...
        (cy * reshape(fIn,9,ly*lx)), 1,ly,ly) ./rho;

    % MACROSCOPIC (DIRICHLET) BOUNDARY CONDITIONS
    % Inlet: Poiseuille profile
    L = ly-2; y = col-1.5;
    ux(:,1,col) = 4 * uMax / (L*L) * (y.*L-y.*y);
    uy(:,1,col) = 0;

```

```

rho(:,1,col) = 1 ./ (1-ux(:,1,col)) .* ( ...
    sum(fIn([1,3,5],1,col)) + ...
    2*sum(fIn([4,7,8],1,col)) );
    % Outlet: Zero gradient on rho/ux
rho(:,lx,col) = 4/3*rho(:,lx-1,col) - ...
    1/3*rho(:,lx-2,col);
uy(:,lx,col) = 0;
ux(:,lx,col) = 4/3*ux(:,lx-1,col) - ...
    1/3*ux(:,lx-2,col);

% COLLISION STEP
for i=1:9
    cu = 3*(cx(i)*ux+cy(i)*uy);
    fEq(i,,:) = rho .* t(i) .* ...
        ( 1 + cu + 1/2*(cu.*cu) ...
          - 3/2*(ux.^2+uy.^2) );
    fOut(i,,:) = fIn(i,,:) - ...
        omega .* (fIn(i,,:)-fEq(i,,:));
end

% MICROSCOPIC BOUNDARY CONDITIONS
for i=1:9
    % Left boundary
    fOut(i,1,col) = fEq(i,1,col) + ...
        18*t(i)*cx(i)*cy(i)* ( fIn(8,1,col) - ...
        fIn(7,1,col)-fEq(8,1,col)+fEq(7,1,col) );
    % Right boundary
    fOut(i,ly,col) = fEq(i,ly,col) + ...
        18*t(i)*cx(i)*cy(i)* ( fIn(6,ly,col) - ...
        fIn(9,ly,col)-fEq(6,ly,col)+fEq(9,ly,col) );
    % Bounce back region
    fOut(i,bbRegion) = fIn(opp(i),bbRegion);
end

% STREAMING STEP
for i=1:9
    fIn(i,,:) = ...
        circshift(fOut(i,,:), [0,cx(i),cy(i)]);
end

% VISUALIZATION
if (mod(cycle,tPlot)==0)
    u = reshape(sqrt(ux.^2+uy.^2),ly,ly);
    u(bbRegion) = nan;
    imagesc(u');
    axis equal off; drawnow
end
end

```