Outils, modélisation et simulation en calcul numérique – Corrigé série 8

10 mai 2005

Exercice 1.
Programme Matlab :

```
clear
% General Constants
L       = 3; %system length
h       = 21; %system heigth
xi      = 0.75; %relaxation time
nu      = 1/3*(xi-1/2); %nu=cs^2(xi-1/2)
F       = [0.002, 0.]; %external force
MaxTime = 5000;
% D2Q9 Lattice constants
t0  = 4/9; t1  = 1/9; t2  = 1/36;
t   = [t0, t1, t2, t1, t2, t1, t2, t1, t2];
cx  = [ 0, +1, +1,  0, -1, -1, -1,  0, +1];
cy  = [ 0,  0, +1, +1, +1,  0, -1, -1, -1];
oppositeOf = [ 1, 6, 7, 8, 9, 2, 3, 4, 5];
% Initialization of the flow densities (rho=1, u=0)
for i=1:9
  fIn(i,:,:) = compute_feq(i, ones(1,L,h), 0, 0, cx, cy, t);
end
for timeStep = 1:MaxTime
  % Collision step
  [rho, ux, uy] = computeMacro(fIn, cx, cy);
  for i=1:9
    fEq(i,:,:) = compute_feq(i, rho, ux, uy, cx, cy, t);
  end
  for i=1:9
    fOut(i,:,:) = fIn(i,:,:) - 1/xi.*(fIn(i,:,:)-fEq(i,:,:)) ...
                  + 3*t(i)*(cx(i)*F(1)+cy(i)*F(2));
  end
  % Bounce back on upper and lower wall
  fOut(:,:,[1 h]) = fIn(oppositeOf,:,[1 h]);
  % Streaming step
  for i = 1:9
    fIn(i,:,:) = circshift(fOut(i,:,:), [0, cx(i), cy(i)]);
  end
  % Output
  if (mod(timeStep,50)==0)
    y1 = (-1/2):(h-3/2);
    u_theory1 = F(1)/2/nu*((h-2)*y1-y1.^2);
    plot(u_theory1, 'ko-');
    hold on
```

```
    y2 = 0:(h-1);
    u_theory2 = F(1)/2/nu*((h-1)*y2-y2.^2);
    plot(u_theory2, 'r.-');
    plot(squeeze(ux(:,1,:)), '*');
    legend('Theoretical curve 1', 'Theoretical curve 2', 'Simulation');
    drawnow; hold off
  end
end
```

avec les fichiers externes *compute_feq.m* pour le calcul de la fonction de distribution à l'équilibre :

```
function feq = compute_feq(i, rho, ux, uy, cx, cy,t)
  Cs2 = 1/3;
  thing = (cx(i)*ux + cy(i)*uy)/Cs2;
  feq = rho .* t(i) .* (1 + thing + 1/2*(thing.*thing)-1/2*(ux.^2+uy.^2)/Cs2);
```

et *computeMacro.m* pour le calcul de la densité ρ et du champ de vitesse $\mathbf{u}$:

```
function [rho, ux, uy] = computeMacro(fIn, cx, cy)
  rho = sum(fIn);
  ux(:,:,:) = 0;
  uy(:,:,:) = 0;
  for i=1:9
    ux = ux + fIn(i,:,:) * cx(i);
    uy = uy + fIn(i,:,:) * cy(i);
  end
  ux = ux./rho;
  uy = uy./rho;
```

La Fig. 1 représente le profil de vitesse obtenu ainsi que la comparaison avec la prédiction théorique.

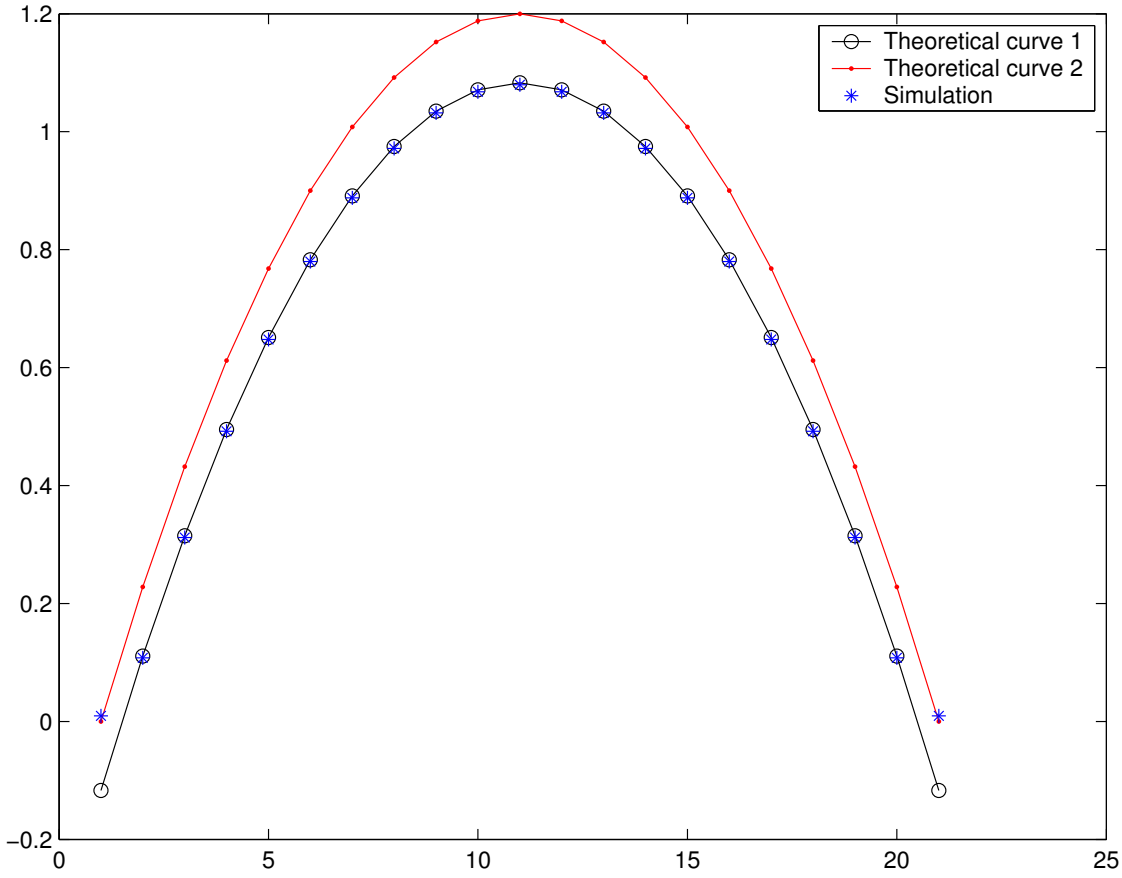

FIG. 1 – Profil des vitesses $u_x(z)$ en fonction de la hauteur z obtenu par la simulation de Boltzmann sur réseau D2Q9, ainsi que la comparaison avec les prédictions théoriques. La courbe théorique numéro 2 (non reproduite par les simulations) suppose que la taille verticale du système est égale à celle du réseau tandis que la courbe théorique numéro 1 (reproduite par les simulations) suppose la taille verticale du système inférieure de une cellule. Cela reflète le fait que le "bounce back" devient correct si on suppose que le mur se trouve entre les deux derniers plus proches sites de la paroi.