Num. #6: Kinetic schemes - Correction

The programs are written with the MATLAB software.

For the exercise, the following functions are needed

- **Kinetic scheme for a conservation law** :

```
   %% Kinetic scheme
%% Periodic boundary conditions
function[ufinal]=kinetic(T,dt,L,dx,uinit,f)
    %% Time discretization
    time=0:dt:T;
    Nt=length(time);
 %% Initial datum - We calculate on N-1 points
    u=uinit(1:end-1);
  %% kinetic method
  for i=1:Nt
      %% Periodic boundary conditions
      %% Computation of vectors up(i)=u_{i+1}, um(i)=u_(i-1)
      up=[u(2:end);u(1)];
      um=[u(end);u(1:end-1)];
      %% Selection of indices of positive and negative values;
      posp=(up>=0); pos=(u>=0);posm=(um>=0);
      negp=(up<0); neg=(u<0);negm=(u<0);
      %% computation of flux
      Fp=zeros(size(u));
      Fp(pos)=f(u(pos));Fp(negp)=Fp(negp)+f(up(negp));
      Fm=zeros(size(u));
      Fm(posm)=f(um(posm));Fm(neg)=Fm(neg)+f(u(neg));
      u=u-dt/dx*(Fp-Fm);
  end
  ufinal=[u;u(1)];
```

- **Kinetic scheme for a system of conservation laws** :

```
%% Kinetic scheme
%% Neumann boundary conditions
function[vfinal]=kineticSystem(T,dt,L,dx,vinit,f,specrad,d)
   %% Initial datum - We calculate on N-1 points
  v=vinit(1:end-1,:);
  %%Kinetic method
 %% for i=1:Nt
```

```
Tsimul=0;
while  Tsimul<T
    %% Time step
     dt=0.95*dx/max(specrad(v));
     Tsimul=Tsimul+dt;
     %% Neumann boundary conditions
     %% Computation of vectors up(i)=u_{i+1}, um(i)=u_(i-1)
     if d==3
         vp=[v(2:end,:);v(end,1),-v(end,2),v(end,3)];
         vm=[v(1,1),-v(1,2),v(1,3);v(1:end-1,:)];
 elseif d==2
      vp=[v(2:end,:);v(end,1),-v(end,2)];
         vm=[v(1,1),-v(1,2);v(1:end-1,:)];
     end
     %%Computation of the density
     rho=v(:,1);
     rhop=vp(:,1);
     rhom=vm(:,1);
     %%Computation of the velocity
     u=v(:,2)./v(:,1);
     up=vp(:,2)./vp(:,1);
     um=vm(:,2)./vm(:,1);
     %% Computation of the temperature variable
     Temp=2*v(:,3)./v(:,1)-(v(:,2).^2)./(v(:,1).^2);
     Tempp=2*vp(:,3)./vp(:,1)-(vp(:,2).^2)./(vp(:,1).^2);
     Tempm=2*vm(:,3)./vm(:,1)-(vm(:,2).^2)./(vm(:,1).^2);
   %% Selection of indices
     ind1=(u<-sqrt(3*Temp));
     ind2=((u>=-sqrt(3*Temp))&(u<=sqrt(3*Temp)));
     ind3=(u>sqrt(3*Temp));
     ind1p=(up<-sqrt(3*Tempp));
     ind2p=((up>=-sqrt(3*Tempp))&(up<=sqrt(3*Tempp)));
     ind3p=(up>sqrt(3*Tempp));
     ind1m=(um<-sqrt(3*Tempm));
      ind2m=((um>=-sqrt(3*Tempm))&(um<=sqrt(3*Tempm)));
       ind3m=(um>sqrt(3*Tempm));
     %% computation of flux
     Fp=zeros(size(v));Fp(ind3,:)=f(v(ind3,:));
     Fp(ind2,:)=[rho(ind2).*((u(ind2)+sqrt(3*Temp(ind2))).^2)
     ./sqrt(Temp(ind2))/2/sqrt(3)/2,
     rho(ind2).*((u(ind2)+sqrt(3*Temp(ind2))).^3)./sqrt(Temp(ind2))/3/sqrt(3)/2,
```

```
      rho(ind2).*((u(ind2)+sqrt(3*Temp(ind2))).^4)./sqrt(Temp(ind2))/8/sqrt(3)/2];
      Fp(ind1p,:)=Fp(ind1p,:)+f(vp(ind1p,:));
      Fp(ind2p,:)=Fp(ind2p,:)-[rhop(ind2p).*((up(ind2p)-sqrt(3*Tempp(ind2p))).^2)
      ./sqrt(Tempp(ind2p))/2/sqrt(3)/2,
      rhop(ind2p).*((up(ind2p)-sqrt(3*Tempp(ind2p))).^3)./sqrt(Tempp(ind2p))/3/sqrt(3)/2,
      rhop(ind2p).*((up(ind2p)-sqrt(3*Tempp(ind2p))).^4)./sqrt(Tempp(ind2p))/8/sqrt(3)/2];

       Fm=zeros(size(v));Fm(ind3m,:)=f(vm(ind3m,:));
      Fm(ind2m,:)=[rhom(ind2m).*((um(ind2m)+sqrt(3*Tempm(ind2m))).^2)
      ./sqrt(Tempm(ind2m))/2/sqrt(3)/2,
      rhom(ind2m).*((um(ind2m)+sqrt(3*Tempm(ind2m))).^3)
      ./sqrt(Tempm(ind2m))/3/sqrt(3)/2,
      rhom(ind2m).*((um(ind2m)+sqrt(3*Tempm(ind2m))).^4)
      ./sqrt(Tempm(ind2m))/8/sqrt(3)/2];
      Fm(ind1,:)=Fm(ind1,:)+f(v(ind1,:));
      Fm(ind2,:)=Fm(ind2,:)-[rho(ind2).*((u(ind2)-sqrt(3*Temp(ind2))).^2)
      ./sqrt(Temp(ind2))/2/sqrt(3)/2,
      rho(ind2).*((u(ind2)-sqrt(3*Temp(ind2))).^3)./sqrt(Temp(ind2))/3/sqrt(3)/2,
      rho(ind2).*((u(ind2)-sqrt(3*Temp(ind2))).^4)./sqrt(Temp(ind2))/8/sqrt(3)/2];

      v=v-dt/dx*(Fp-Fm);
  end
   if d==3
   vfinal=[v;v(end,1), -v(end,2),v(end,3)];
elseif d==2
     vfinal=[v;v(end,1), -v(end,2)];
end
```

1 Burgers equation

Exercise

1. Give the expression of fluxes $F_{i+\frac{1}{2}}$. Implement the scheme (with periodic boundary conditions).
2. Perform a test case with the following initial datum on the interval $[0,5]$:

$$u_0(x) = \mathrm{e}^{-(x-2)^2/0.1}.$$

```
  %% Space discretization
L=5;
dx=0.01;
```

```
space=(0:dx:L)';
%% Time discretization
T=1;
dt=dx*0.95;
%%%%%%%% INITIAL DATA
%% Initial datum 1
uinit=exp(-(space-2).^2/0.1);
%% EQUATIONS
%% flux function  and derivative = Burgers
f=inline('x.^2/2');
a=inline('x');
%% kinetic scheme
uKin=kinetic(T,dt,L,dx,uinit,f);
plot(space,uKin,'b');
```

3. Highlight the CFL condition.

2 Compressible Euler system

Exercise

1. Give the expression of fluxes $F_{i+\frac{1}{2}}$. Implement the scheme.
2. Perform a test case with the following initial data on the domain $[0,1]$ with Neumann boundary conditions, as in session #4:

$$\rho(0,x) = \begin{cases} 1 & \text{if} \quad x \leqslant 0.5, \\ 0.125 & \text{if} \quad x > 0.5, \end{cases} \qquad p(0,x) = \begin{cases} 1 & \text{if} \quad x \leqslant 0.5, \\ 0.1 & \text{if} \quad x > 0.5, \end{cases} \qquad \text{and} \qquad u(0,x) = 0.$$

```
          %% Space discretization
L=1;
dx=0.01;
space=(0:dx:L)';
%% Time discretization
T=0.3;
dt=dx*0.95;
%% Function and  spectral radius of the jacobian
%%for the Euler compressible system in the (rho, j,E) variables
f=inline('[u(:,2),2*u(:,3),3*u(:,2).*u(:,3)./u(:,1)-u(:,2).^3./u(:,1).^2]','u');
specrad=inline('max(abs(u(:,2)./u(:,1)-sqrt(3*(2*u(:,3)./u(:,1)-u(:,2).^2./u(:,1).^2))),
abs(u(:,2)./u(:,1)+sqrt(3*(2*u(:,3)./u(:,1)-u(:,2).^2./u(:,1).^2))))','u');
%% Initial data
```

```
space1=space(space<0.5);
space2=space(space>=0.5);
rhoinit=[ones(size(space1));0.125*ones(size(space2))];
uinit=zeros(size(space));
pinit=[ones(size(space1));0.1*ones(size(space2))];
%% in the (rho, j,E) variables
vinit=[rhoinit,rhoinit.*uinit,(rhoinit.*uinit.^2+pinit)/2];
%% Solution of the system with kinetic scheme
v=kineticSystem(T,dt,L,dx,vinit,f,specrad,3);
plot(space,v(:,1),'k');
hold on;
plot(space,v(:,2),'r');
plot(space,v(:,3),'g');
legend('density','momentum','energy')
```