Num. #5: 1D hyperbolic PDEs system : 2nd order schemes - Correction

The programs are written with the MATLAB software.
For the exercise, the following functions are needed

- **Flux-limiters method for the transport equation** :

```
%% Flux Limiter method - Transport equation
%% Periodic boundary conditions - periodic function a
%% Phi is the function defining the flux limiter
function[ufinal]=FluxLimiter(T,dt,L,dx,uinit,a,Phi)
     %% Time discretization
    time=0:dt:T;
    Nt=length(time);
    %% Space discretization COLUMN vector
    space=(0:dx:L)';
  %% Initial datum - We calculate on N-1 points
    u=uinit(1:end-1);
  %% Computation of the velocity
    vel=a(space(1:end-1));
    %% Computations of vectors velp(i)=a(x_{i+1}) and velm(i)=a(x_{i-1})
      velp=[vel(2:end);vel(1)];
      velm=[vel(end);vel(1:end-1)];
  %% Flux Limiter 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)];
      umm=[u(end-1:end);u(1:end-2)];
      %% Flux limiters- Careful  when up=u or um=u
      Phip=zeros(size(u));Phim=zeros(size(u));
      indices1=(up==u); indices2=(up~=u);
      Phip(indices1)=0;
      teta=(u(indices2)-um(indices2))./(up(indices2)-u(indices2));
      Phip(indices2)=Phi(teta);
      indices1m=(u==um); indices2m=(u~=um);
      Phim(indices1m)=0;
      Phim(indices2m)=Phi((um(indices2m)-umm(indices2m))./(u(indices2m)-um(indices2m)));
            %% computation of flux
      Fp=vel.*(u+up)/2-abs(vel).*(up-u)/2
```

```
              +abs(vel).*(ones(size(vel))-abs(vel)*dt/dx).*(up-u).*Phip/2;
       Fm=velm.*(um+u)/2-abs(velm).*(u-um)/2
                +abs(velm).*(ones(size(velm))-abs(velm)*dt/dx).*(u-um).*Phim/2;
       u=u-dt/dx*(Fp-Fm);
   end
    ufinal=[u;u(1)];
```

- **Flux-limiters method for conservation law** :

```
%% Flux Limiter method - Conservation law
%% Periodic boundary conditions - periodic function a
%% Phi is the function defining the flux limiter
function[ufinal]=FluxLimiterNL(T,dt,L,dx,uinit,f,a,Phi)
     %% Time discretization
    time=0:dt:T;
    Nt=length(time);
    %% Space discretization COLUMN vector
    space=(0:dx:L)';
  %% Initial datum - We calculate on N-1 points
    u=uinit(1:end-1);
  %% Flux Limiter 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)];
      umm=[u(end-1:end);u(1:end-2)];
      %% Difference of functions
      Dfp=f(up)-f(u);
      Df=f(u)-f(um);
      Dfm=f(um)-f(umm);
      %% Velocities
       velp=zeros(size(u));vel=zeros(size(u));velm=zeros(size(u));
      indices1=(up==u); indices2=(up~=u);
      velp(indices1)=a(u(indices1));
      velp(indices2)=(f(up(indices2))-f(u(indices2)))./(up(indices2)-u(indices2));
      indices1m=(u==um); indices2m=(u~=um);
      vel(indices1m)=a(um(indices1m));
      vel(indices2m)=(f(u(indices2m))-f(um(indices2m)))./(u(indices2m)-um(indices2m));
      indices1mm=(um==umm); indices2mm=(um~=umm);
      velm(indices1mm)=a(umm(indices1mm));
```

```
        velm(indices2mm)=(f(um(indices2mm))-f(umm(indices2mm)))
                                        ./(um(indices2mm)-umm(indices2mm));
        %% Limiters
        Phip=zeros(size(u));Phim=zeros(size(u));
        indices1f=(f(up)==f(u)); indices2f=(f(up)~=f(u));
        Tetap=(ones(size(vel(indices2f)))-dt*vel(indices2f)/dx)./(ones(size(velp(indices2f))
                              -dt*velp(indices2f)/dx).*Df(indices2f)./Dfp(indices2f);
        Phip(indices2f)=Phi(Tetap); Phip(indices1f)=0;
        indices1mf=(f(u)==f(um)); indices2mf=(f(u)~=f(um));
        Tetam=(ones(size(velm(indices2mf)))
        -dt*velm(indices2mf)/dx)./(ones(size(vel(indices2mf)))
        -dt*vel(indices2mf)/dx).*Dfm(indices2mf)./Df(indices2mf);
        Phim(indices2mf)=Phi(Tetam); Phim(indices1mf)=0;
        %% computation of flux
        Fp=f(u)+Dfp.*(ones(size(velp))-dt*velp/dx).*Phip/2;
        Fm=f(um)+Df.*(ones(size(vel))-dt*vel/dx).*Phim/2;
        u=u-dt/dx*(Fp-Fm);
    end
     ufinal=[u;u(1)];
```

1 Transport equation

Exercise

1. Implement the flux limiters methods presented here in the case $a = 1$, with initial datum (3a). Use periodic boundary conditions, $\Delta x = 0.01$ and $\Delta t = 0.95\Delta x$.

```
     %% Space discretization
L=5;
dx=0.01;
space=(0:dx:L)';
%% Time discretization
T=1;
dt=dx*0.95;
%%/ space should be a column vector
%% Velocity of the transport equation -- a=1
a=inline('ones(size(x))');
%% Initial datum 1
uinit=exp(-(space-2).^2/0.1);
%% Initial datum 2
%space1=space(space<1);
```

```
%space2=space((space>=1)&(space<=2));
%space3=space(space>2);
%uinit=[zeros(size(space1));ones(size(space2)); zeros(size(space3))];
%%/ FLux limiter 1 : Minmod
Minmod=inline('max(zeros(size(x)),min(x,ones(size(x))))','x');
%%/ FLux limiter 2 : ROe's superbee
Roe=inline('max(zeros(size(x)),max(min(x,2*ones(size(x))),min(ones(size(x)),2*x)))','x');
%%/ FLux limiter 3 : Van Leer
VanLeer=inline('(x+abs(x))./(ones(size(x))+abs(x))','x');
%% Comparison of the different flux limiter functions
uMm=FluxLimiter(T,dt,L,dx,uinit,a,Minmod);
plot(space,uMm);
hold on;
uRoe=FluxLimiter(T,dt,L,dx,uinit,a,Roe);
plot(space,uRoe,'r');
uVL=FluxLimiter(T,dt,L,dx,uinit,a,VanLeer);
plot(space,uVL,'g');
legend('Minmod','Roe','Van Leer');
```

2. Choose one of the previous limiter functions and highlight the order of the scheme. The error will be defined as the difference between the exact solution and the solution computed.

```
      L=5;
SpaceStep=[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005];
for k=1:length(SpaceStep),
dx=SpaceStep(k);
space=(0:dx:L)';
%% Time discretization
T=1;
dt=0.95*dx;
time=0:dt:T;
    Nt=length(time);
Tsimu=dt*Nt;
%% Velocity of the transport equation -- a=1
a=inline('ones(size(x))');
%%%% Initial datum 1
uinit=exp(-(space-2).^2/0.1);
%% Initial datum 2
%space1=space(space<1);
%space2=space((space>=1)&(space<=2));
%space3=space(space>2);
```

```
%uinit=[zeros(size(space1));ones(size(space2)); zeros(size(space3))];
%%%% Exact solution
uexact=exp(-(space-2-Tsimu).^2/0.1);
%% Exact solution  2
%space1b=space(space<1);
%space2b=space((space>=1+Tsimu)&(space<=2+Tsimu));
%space3b=space(space>2+Tsimu);
%uexact=[zeros(size(space1b));ones(size(space2b)); zeros(size(space3b))];
%% Approximated solution
uMm=FluxLimiter(T,dt,L,dx,uinit,a,Minmod);
ErrorMm(k)=dx*norm(uMm-uexact,1);
end
%% Clear the figure
clf;
%% Graph of the errors
loglog(SpaceStep,ErrorMm,'b');
hold on;
loglog(SpaceStep, SpaceStep.^2,'k');
%% Legend for the graph
legend('Minmod','order 2');
```

2 Conservation law

Exercise

1. Implement the flux limiters methods presented here in the case of the Burgers equation $f(u) = \frac{u^2}{2}$ with initial data (3a) and (3b). Use periodic boundary conditions, $\Delta x = 0.01$ and $\Delta t = 0.95\Delta x$.

```
    %% Space discretization
L=5;
dx=0.1;
space=(0:dx:L)';
%% Time discretization
T=1;
dt=dx*0.95;
%%/ space should be a colmun vector
%% Functions- Burger equation
f=inline('x.^2/2');
a=inline('x');
%% Initial datum 1
uinit=exp(-(space-2).^2/0.1);
```

```
%% Initial datum 2
%space1=space(space<1);
%space2=space((space>=1)&(space<=2));
%space3=space(space>2);
%uinit=[zeros(size(space1));ones(size(space2)); zeros(size(space3))];
%%/ FLux limiter 1 : Minmod
Minmod=inline('max(zeros(size(x)),min(x,ones(size(x))))','x');
%% Comparison of the different flux limiter functions
uMm=FluxLimiterNL(T,dt,L,dx,uinit,f,a,Minmod);
plot(space,uMm);
```