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

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

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

```
   //// Flux Limiter method - Conservation law
//// Periodic boundary conditions
//// 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:$-1);
  //// Flux Limiter method
  for i=1:Nt
      //// Periodic boundary conditions
      //// Computation of vectors up(i)=u_{i+1}, um(i)=u_(i-1)
      up=[u(2:$);u(1)];
      um=[u($);u(1:$-1)];
      umm=[u($-1:$);u(1:$-2)];
      //// Difference of functions
      Dfp=f(up)-f(u);
      Df=f(u)-f(um);
      Dfm=f(um)-f(umm);
      //// Velocities
       velp=zeros(u);vel=zeros(u);velm=zeros(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(u);Phim=zeros(u);
        indices1f=(f(up)==f(u)); indices2f=(f(up)~=f(u));
        Tetap=(ones(vel(indices2f))-dt*vel(indices2f)/dx)./
        (ones(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(velm(indices2mf))-dt*velm(indices2mf)/dx)./
        (ones(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(velp)-dt*velp/dx).*Phip/2;
        Fm=f(um)+Df.*(ones(vel)-dt*vel/dx).*Phim/2;
        u=u-dt/dx*(Fp-Fm);
    end
     ufinal=[u;u(1)];
     endfunction
```

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

```
     //// Flux Limiter method - Conservation law
  //// Periodic boundary conditions
  //// 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:$-1);
    //// Flux Limiter method
    for i=1:Nt
        //// Periodic boundary conditions
        //// Computation of vectors up(i)=u_{i+1}, um(i)=u_(i-1)
        up=[u(2:$);u(1)];
        um=[u($);u(1:$-1)];
        umm=[u($-1:$);u(1:$-2)];
        //// Difference of functions
        Dfp=f(up)-f(u);
        Df=f(u)-f(um);
```

```
    Dfm=f(um)-f(umm);
    //// Velocities
     velp=zeros(u);vel=zeros(u);velm=zeros(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(u);Phim=zeros(u);
    indices1f=(f(up)==f(u)); indices2f=(f(up)~=f(u));
    Tetap=(ones(vel(indices2f))-dt*vel(indices2f)/dx)
    ./(ones(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(velm(indices2mf))-dt*velm(indices2mf)/dx)
    ./(ones(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(velp)-dt*velp/dx).*Phip/2;
    Fm=f(um)+Df.*(ones(vel)-dt*vel/dx).*Phim/2;
    u=u-dt/dx*(Fp-Fm);
end
 ufinal=[u;u(1)];
 endfunction
```

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 $\Delta t = 0.95\Delta x$.

```
// Transport equation and Conservation Law
// Second order methods
//// 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
deff('[y]=a(x)','y=1');
//// Initial datum 1
uinit=exp(-(space-2).^2/0.1);
///// FLux limiter 1 : Minmod
deff('[y]=Minmod(x)','y=max(zeros(x),min(x,ones(x)))');
///// FLux limiter 2 : Roe's superbee
deff('[y]=Roe(x)','y=max(zeros(x),max(min(x,2*ones(x)),min(ones(x),2*x)))');
///// FLux limiter 3 : Van Leer
deff('[y]=VanLeer(x)','y=(x+abs(x))./(ones(x)+abs(x))');
//// Comparison of the different flux limiter functions
uMm=FluxLimiter(T,dt,L,dx,uinit,a,Minmod);
plot(space,uMm);
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.

```
      // Transport equation and Conservation Law
// Second order methods

////////////////////// Computation of the error
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
deff('[y]=a(x)','y=1');
//////// Initial datum 1
uinit=exp(-(space-2).^2/0.1);
//////// Exact solution
uexact=exp(-(space-2-Tsimu).^2/0.1);
//// 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
plot2d(SpaceStep,ErrorMm,logflag="ll",style=1);
plot2d(SpaceStep, SpaceStep.^2,logflag="ll",style=2);
//// 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 $\Delta t = 0.95\Delta x$.

```
 // Transport equation and Conservation Law
// Second order methods
//// Space discretization
L=5;
dx=0.01;
space=(0:dx:L)';
//// Time discretization
T=1;
dt=dx*0.95;
///// space should be a colmun vector
//// Functions- Burger equation
deff('[y]=f(x)','y=x.^2/2');
deff('[y]=a(x)','y=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(space1);ones(space2); zeros(space3)];
///// FLux limiter 1 : Minmod
deff('[y]=Minmod(x)','y=max(zeros(x),min(x,ones(x)))');
////Approximated solution
uMm=FluxLimiterNL(T,dt,L,dx,uinit,f,a,Minmod);
plot(space,uMm);
```