Num. #1: Hyperbolic PDE equation : 1D conservation law

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

- **Upwind conservative method** :

```
// Upwind method
// Periodic boundary conditions
function[ufinal]=upwind(T,dt,L,dx,uinit,f,a)
    // Time discretization
    time=0:dt:T;
    Nt=length(time);
  // Initial datum - We calculate on N-1 points
    u=uinit(1:$-1);
  // upwind method
  for i=1:Nt
      // Periodic boundary conditions
      // Computation of vectors up(i)=u_{i+1}, um(i)=u_(i-1)
      up=[u(2:$);u(1)];
      um=[u($);u(1:$-1)];
      // computation of the velocities
      vel=a((u+up)/2);
      velm=a((um+u)/2);
      // computation of flux
      Fp=zeros(u);Fm=zeros(u);
      Fp(vel>=0)=f(u(vel>=0));
      Fp(vel<0)=f(up(vel<0));
      Fm(velm>=0)=f(um(velm>=0));
      Fm(velm<0)=f(u(velm<0));
      u=u-dt/dx*(Fp-Fm);
  end
   ufinal=[u;u(1)];
   endfunction
```

- **Roe method** :

```
// Roe method
// Periodic boundary conditions
function[ufinal]=Roe(T,dt,L,dx,uinit,f,a)
    // Time discretization
    time=0:dt:T;
```

```
    Nt=length(time);
  // Initial datum - We calculate on N-1 points
  u=uinit(1:$-1);
  // Roe method
  for i=1:Nt
      // Periodic boundary conditions
      // Computation of vectors up(i)=u_{i+1}, um(i)=u_(i-1)
      up=[u(2:$);u(1)];
      um=[u($);u(1:$-1)];
      // computation of the velocities
      vel=a(u);
      indices=(u~=up);
      vel(indices)=(f(u(indices))-f(up(indices)))./(u(indices)-up(indices));
      velm=a(um);
      indicesm=(um~=u);
      velm(indicesm)=(f(um(indicesm))-f(u(indicesm)))./(um(indicesm)-u(indicesm));
       // computation of flux
       Fp=zeros(u);Fm=zeros(u);
      Fp(vel>=0)=f(u(vel>=0));
      Fp(vel<0)=f(up(vel<0));
      Fm(velm>=0)=f(um(velm>=0));
      Fm(velm<0)=f(u(velm<0));
      u=u-dt/dx*(Fp-Fm);
  end
   ufinal=[u;u(1)];
   endfunction
```

- **Engquist-Osher method** :

```
// Engquist Osher method
// Periodic boundary conditions
// equation = 'Burgers'
function[ufinal]=EngquistOsher(T,dt,L,dx,uinit,f,a)
// For Burgers equation
       deff('[y]=fpp(x)','y=x.*(x+abs(x))/4');
       deff('[y]=fmm(x)','y=x.^2/2-x.*(x+abs(x))/4');
    // Time discretization
    time=0:dt:T;
    Nt=length(time);
 // Initial datum - We calculate on N-1 points
    u=uinit(1:$-1);
```

```
  // Engquist-Osher 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)];
      // computation of flux
      Fp=fpp(u)+fmm(up);
      Fm=fpp(um)+fmm(u);
      u=u-dt/dx*(Fp-Fm);
  end
  ufinal=[u;u(1)];
  endfunction
```

- **Lax-Friedrichs method** :

```
// Lax Friedrichs method
// Periodic boundary conditions
function[ufinal]=LaxFriedrichs(T,dt,L,dx,uinit,f,a)
    // Time discretization
    time=0:dt:T;
    Nt=length(time);
 // Initial datum - We calculate on N-1 points
    u=uinit(1:$-1);
  // Lax-Friedrichs method
  for i=1:Nt
      // Periodic boundary conditions
      // Computation of vectors up(i)=u_{i+1}, um(i)=u_(i-1)
      up=[u(2:$);u(1)];
      um=[u($);u(1:$-1)];
      // computation of flux
      Fp=(f(u)+f(up))/2-dx*(up-u)/2/dt;
      Fm=(f(um)+f(u))/2-dx*(u-um)/2/dt;
      u=u-dt/dx*(Fp-Fm);
  end
  ufinal=[u;u(1)];
  endfunction
```

- **Rusanov (or Local Lax-Friedrichs) method** :

```
// Rusanov method
```

```
// Periodic boundary conditions
function[ufinal]=Rusanov(T,dt,L,dx,uinit,f,a)
    // Time discretization
    time=0:dt:T;
    Nt=length(time);
 // Initial datum - We calculate on N-1 points
    u=uinit(1:$-1);
  // Rusanov method
  for i=1:Nt
      // Periodic boundary conditions
      // Computation of vectors up(i)=u_{i+1}, um(i)=u_(i-1)
      up=[u(2:$);u(1)];
      um=[u($);u(1:$-1)];
      // velocity velp(i)=a_{i+1/2} and velm(i)=a_{i-1/2}
      vel=max(abs(a(u)),abs(a(up)));
      velm=max(abs(a(um)),abs(a(u)));
      // computation of flux
      Fp=(f(u)+f(up))/2-vel.*(up-u)/2;
      Fm=(f(um)+f(u))/2-velm.*(u-um)/2;
      u=u-dt/dx*(Fp-Fm);
  end
  ufinal=[u;u(1)];
  endfunction
```

- **Lax-Wendroff method** :

```
// Lax Wendroff method
// Periodic boundary conditions
function[ufinal]=LaxWendroff(T,dt,L,dx,uinit,f,a)
     // Time discretization
    time=0:dt:T;
    Nt=length(time);
  // Initial datum - We calculate on N-1 points
    u=uinit(1:$-1);
  // Lax-Wendroff method
  for i=1:Nt
      // Periodic boundary conditions
      // Computation of vectors up(i)=u_{i+1}, um(i)=u_(i-1)
      up=[u(2:$);u(1)];
      um=[u($);u(1:$-1)];
      // velocity velp(i)=a_{i+1/2} and velm(i)=a_{i-1/2}
```

```
      vel=a((up+u)/2);
      velm=a((u+um)/2);
      // computation of flux
      Fp=(f(u)+f(up))/2-dt*vel.*(f(up)-f(u))/2/dx;
      Fm=(f(um)+f(u))/2-dt*velm.*(f(u)-f(um))/2/dx;
      u=u-dt/dx*(Fp-Fm);
  end
   ufinal=[u;u(1)];
endfunction
```

- **Upwind non conservative method** :

```
// Upwind non-conservative method
// Periodic boundary conditions
function[ufinal]=upwindNC(T,dt,L,dx,uinit,f,a)
    // Time discretization
    time=0:dt:T;
    Nt=length(time);
  // Initial datum - We calculate on N-1 points
  u=uinit(1:$-1);
    // upwind  non conservative method
  for i=1:Nt
      // Periodic boundary conditions
      // Computation of vectors up(i)=u_{i+1}, um(i)=u_(i-1)
      up=[u(2:$);u(1)];
      um=[u($);u(1:$-1)];
      // Computation of the velocity
      vel=a(u);
     // Computation of the solution
      u=u-dt/dx*((u-um).*(vel+abs(vel))+(up-u).*(vel-abs(vel)))/2;
  end
   ufinal=[u;u(1)];
   endfunction
```

Exercise

1. Compute the functions f^+ and f^- of the Engquist-Osher flux in the case of equation (2).

2. Implement the resolution of equation (2) using the seven methods (4) presented above. We consider the interval $[0,5]$ with a space step $\Delta x = 0.01$ and periodic boundary conditions. We compute the

solution until time $T = 1$ with a time step satisfying $\Delta t = 0.95 * \Delta x$ and we will use function (5c) as an initial datum.

```
// Space discretization
L=5;
dx=0.01;
space=(0:dx:L)';
// Time discretization
T=1;
dt=0.95*dx;
// Initial datum 3
space1=space(space<1);
space2=space((space>=1)&(space<=2));
space3=space(space>2);
uinit=[zeros(space1);ones(space2); zeros(space3)];
// flux function  and derivative = Burgers
deff('[y]=f(x)','y=x.^2/2');
deff('[y]=a(x)','y=x');
// Approximated solution
uUp=upwind(T,dt,L,dx,uinit,f,a);
plot(space,uUp,'k');
uRoe=Roe(T,dt,L,dx,uinit,f,a);
plot(space,uRoe,'m');
uEO=EngquistOsher(T,dt,L,dx,uinit,f,a);
plot(space,uEO,'b');
uLF=LaxFriedrichs(T,dt,L,dx,uinit,f,a);
plot(space,uLF,'r');
uRus=Rusanov(T,dt,L,dx,uinit,f,a);
plot(space,uRus,'c');
uLW=LaxWendroff(T,dt,L,dx,uinit,f,a);
plot(space,uLW,'g');
uUpNC=upwindNC(T,dt,L,dx,uinit,f,a);
plot(space,uUpNC,'y');
legend('upwind', 'Roe', 'Engquist- Osher', 'Lax Friedrichs','Rusanov', 'Lax Wendroff',
                  'upwind Non Conservative')
```

3. Compare the seven schemes in the case of the two other initial data (5a) and (5b) What is your conclusion ? Choose one of these schemes and plot the evolution of the solution with time.

```
    // Space discretization
L=5;
```

```
dx=0.01;
space=(0:dx:L)';
// Time discretization
T=1;
dt=0.95*dx;
// Initial datum 1
uinit=exp(-(space-2).^2/0.1);
// Initial datum 2
//space1=space(space<1);
//space2=space((space>=1)&(space<=3));
//space3=space(space>3);
//uinit=[zeros(space1);1-abs(space2-2); zeros(space3)];
// flux function  and derivative = Burgers
deff('[y]=f(x)','y=x.^2/2');
deff('[y]=a(x)','y=x');
// Approximated solution
uUp=upwind(T,dt,L,dx,uinit,f,a);
plot(space,uUp,'k');
uRoe=Roe(T,dt,L,dx,uinit,f,a);
plot(space,uRoe,'m');
uEO=EngquistOsher(T,dt,L,dx,uinit,f,a);
plot(space,uEO,'b');
uLF=LaxFriedrichs(T,dt,L,dx,uinit,f,a);
plot(space,uLF,'r');
uRus=Rusanov(T,dt,L,dx,uinit,f,a);
plot(space,uRus,'c');
uLW=LaxWendroff(T,dt,L,dx,uinit,f,a);
plot(space,uLW,'g');
uUpNC=upwindNC(T,dt,L,dx,uinit,f,a);
plot(space,uUpNC,'y');
legend('upwind', 'Roe', 'Engquist- Osher', 'Lax Friedrichs','Rusanov', 'Lax Wendroff',
                       'upwind Non Conservative')

figure(1)
clf;
u1=EngquistOsher(0.1,dt,L,dx,uinit,f,a);
u2=EngquistOsher(0.4,dt,L,dx,uinit,f,a);
u3=EngquistOsher(0.8,dt,L,dx,uinit,f,a);
u4=EngquistOsher(1,dt,L,dx,uinit,f,a);
plot(space,u1,'k');
plot(space,u2,'g');
```

```
plot(space,u3,'b');
plot(space,u4,'r');
```

4. Show the effect of the CFL condition on the stability of the various schemes.

```
// Space discretization
L=5;
dx=0.01;
space=(0:dx:L)';
// Time discretization
T=1;
//dt=dx*2;
dt=dx*0.95;
// dt=dx*0.5;
// Initial datum 1
uinit=exp(-(space-2).^2/0.1);
// flux function  and derivative = Burgers
deff('[y]=f(x)','y=x.^2/2');
deff('[y]=a(x)','y=x');
// Approximated solution
uUp=upwind(T,dt,L,dx,uinit,f,a);
plot(space,uUp,'k');
uRoe=Roe(T,dt,L,dx,uinit,f,a);
plot(space,uRoe,'m');
uEO=EngquistOsher(T,dt,L,dx,uinit,f,a);
plot(space,uEO,'b');
uLF=LaxFriedrichs(T,dt,L,dx,uinit,f,a);
plot(space,uLF,'r');
uRus=Rusanov(T,dt,L,dx,uinit,f,a);
plot(space,uRus,'c');
uLW=LaxWendroff(T,dt,L,dx,uinit,f,a);
plot(space,uLW,'g');
uUpNC=upwindNC(T,dt,L,dx,uinit,f,a);
plot(space,uUpNC,'y');
legend('upwind', 'Roe', 'Engquist- Osher', 'Lax Friedrichs','Rusanov', 'Lax Wendroff',
                         'upwind Non Conservative')
```

5. Compare upwind conservative and upwind non-conservative scheme for equation (2) with initial datum (5c). What do you notice ?

```
// Space discretization
```

```
L=5;
dx=0.01;
space=(0:dx:L)';
// Time discretization
T=1;
dt=dx*0.95;
// Initial datum 3
space1=space(space<1);
space2=space((space>=1)&(space<=2));
space3=space(space>2);
uinit=[zeros(space1);ones(space2); zeros(space3)];
// flux function  and derivative = Burgers
deff('[y]=f(x)','y=x.^2/2');
deff('[y]=a(x)','y=x');
// Approximated solution
uUp=upwind(T,dt,L,dx,uinit,f,a);
plot(space,uUp,'k');
uUpNC=upwindNC(T,dt,L,dx,uinit,f,a);
plot(space,uUpNC,'y');
legend('upwind','upwind Non Conservative')
```

6. What are the results of Roe scheme with equation (2) and initial datum (5d). What is your interpretation ? Do the other schemes have the same drawback ?

```
// Space discretization
L=5;
dx=0.01;
space=(0:dx:L)';
// Time discretization
T=1;
dt=dx*0.95;
////// Initial datum 4
space1=space(space<1);
space2=space((space>=1)&(space<=2));
space3=space(space>2);
uinit=[-ones(space1);ones(space2); -ones(space3)];
//// flux function 1 and derivative = Burgers
deff('[y]=f(x)','y=x.^2/2');
deff('[y]=a(x)','y=x');
// Approximated solution
uUp=upwind(T,dt,L,dx,uinit,f,a);
```

```
plot(space,uUp,'k');
uRoe=Roe(T,dt,L,dx,uinit,f,a);
plot(space,uRoe,'m');
uEO=EngquistOsher(T,dt,L,dx,uinit,f,a);
plot(space,uEO,'b');
uLF=LaxFriedrichs(T,dt,L,dx,uinit,f,a);
plot(space,uLF,'r');
uRus=Rusanov(T,dt,L,dx,uinit,f,a);
plot(space,uRus,'c');
uLW=LaxWendroff(T,dt,L,dx,uinit,f,a);
plot(space,uLW,'g');
uUpNC=upwindNC(T,dt,L,dx,uinit,f,a);
plot(space,uUpNC,'y');
legend('upwind', 'Roe', 'Engquist- Osher', 'Lax Friedrichs','Rusanov', 'Lax Wendroff',
          'upwind Non Conservative')
```