SOLUTIONS
Problem Set 2 - Hyperbolic Equations

Handed out: March 10, 2003 — Due: March 31, 2003

Problem 1 - Solitons

1) Differentiate the one-soliton solution:

$$u(x,t) = -1/2\,\frac{v}{\left(\cosh\left(1/2\,\sqrt{v}\,(x-vt-x_0)\right)\right)^2} \tag{1}$$

$$u_x(x,t) = 1/2\,\frac{v^{3/2}\sinh\left(1/2\,\sqrt{v}\,(x-vt-x_0)\right)}{\left(\cosh\left(1/2\,\sqrt{v}\,(x-vt-x_0)\right)\right)^3} \tag{2}$$

$$u_{xx}(x,t) = -1/4\,\frac{v^2\left(2\,\left(\cosh\left(1/2\,\sqrt{v}\,(x-vt-x_0)\right)\right)^2-3\right)}{\left(\cosh\left(1/2\,\sqrt{v}\,(x-vt-x_0)\right)\right)^4} \tag{3}$$

$$u_{xxx}(x,t) = 1/2\,\frac{v^{5/2}\sinh\left(1/2\,\sqrt{v}\,(x-vt-x_0)\right)\left(\left(\cosh\left(1/2\,\sqrt{v}\,(x-vt-x_0)\right)\right)^2-3\right)}{\left(\cosh\left(1/2\,\sqrt{v}\,(x-vt-x_0)\right)\right)^5} \tag{4}$$

$$u_t(x,t) = -1/2\,\frac{v^{5/2}\sinh\left(1/2\,\sqrt{v}\,(x-vt-x_0)\right)}{\left(\cosh\left(1/2\,\sqrt{v}\,(x-vt-x_0)\right)\right)^3}. \tag{5}$$

Insert into the KdV equation and simplify to get

$$u_t(x,t) - 6u(x,t)u_x(x,t) + u_{xxx}(x,t) = 0. \tag{6}$$

2) Derive second-order discretizations in space using Taylor series, to get:

$$\frac{\partial u_i}{\partial t} = 6u_i\frac{u_{i+1}-u_{i-1}}{2\Delta x} - \frac{u_{i+2}-2u_{i+1}+2u_{i-1}-u_{i-2}}{2\Delta x^3}. \tag{7}$$

3) The discretization of the simplified problem is

$$\frac{\partial u_i}{\partial t} = -\frac{u_{i+2}-2u_{i+1}+2u_{i-1}-u_{i-2}}{2\Delta x^3}. \tag{8}$$

The eigenvalues of this difference operator are

$$\lambda_k = -i\frac{\sin(2\theta)-2\sin(\theta)}{\Delta x^3}, \qquad \theta = 2\pi k\Delta x. \tag{9}$$

These are always purely imaginary, and maximization gives a bound on the magnitude:

$$|\lambda_k| \le \frac{3\sqrt{3}}{2\Delta x^3}. \tag{10}$$

Now, to make sure that $|z_k| = |\Delta t\lambda_k| \le 2\sqrt{2}$, we have that

$$\Delta t \le \Delta x^3\frac{4\sqrt{2}}{3\sqrt{3}} \approx 1.09\Delta x^3. \tag{11}$$

4) The function `kdv.m` in Appendix A solves the problem for any of the given initial conditions. The solutions at time $t = 2$ are shown in Figure 1. Below are short comments on each of the cases:

a. The single soliton propagates to the right, with a velocity somewhat smaller than $v = 16$.

b. Although the Gaussian looks very similar to a soliton, it does not behave in the same way. It is actually closer to a two-soliton solution, but not exactly, so there are ripples.

c. The given two-soliton solution gives two perfect solitons, with different velocities and amplitudes.

d. The equation is non-linear, so it is in general not possible to add two solutions and get a new solution. In this case, the "fastest" soliton propagates, but the smaller one does not move very much, and there are some ripples.

e. When the two one-soliton solutions are well separated, they can be added. When they cross each other, the amplitude is *smaller* than the sum of the two amplitudes. Also, the velocity of the faster soliton increases slightly when they cross, and the velocity of the slower soliton decreases, which can be seen as a small phase-shift or in a plot of the characteristics. After the crossing, they separate back into the two original solitons.

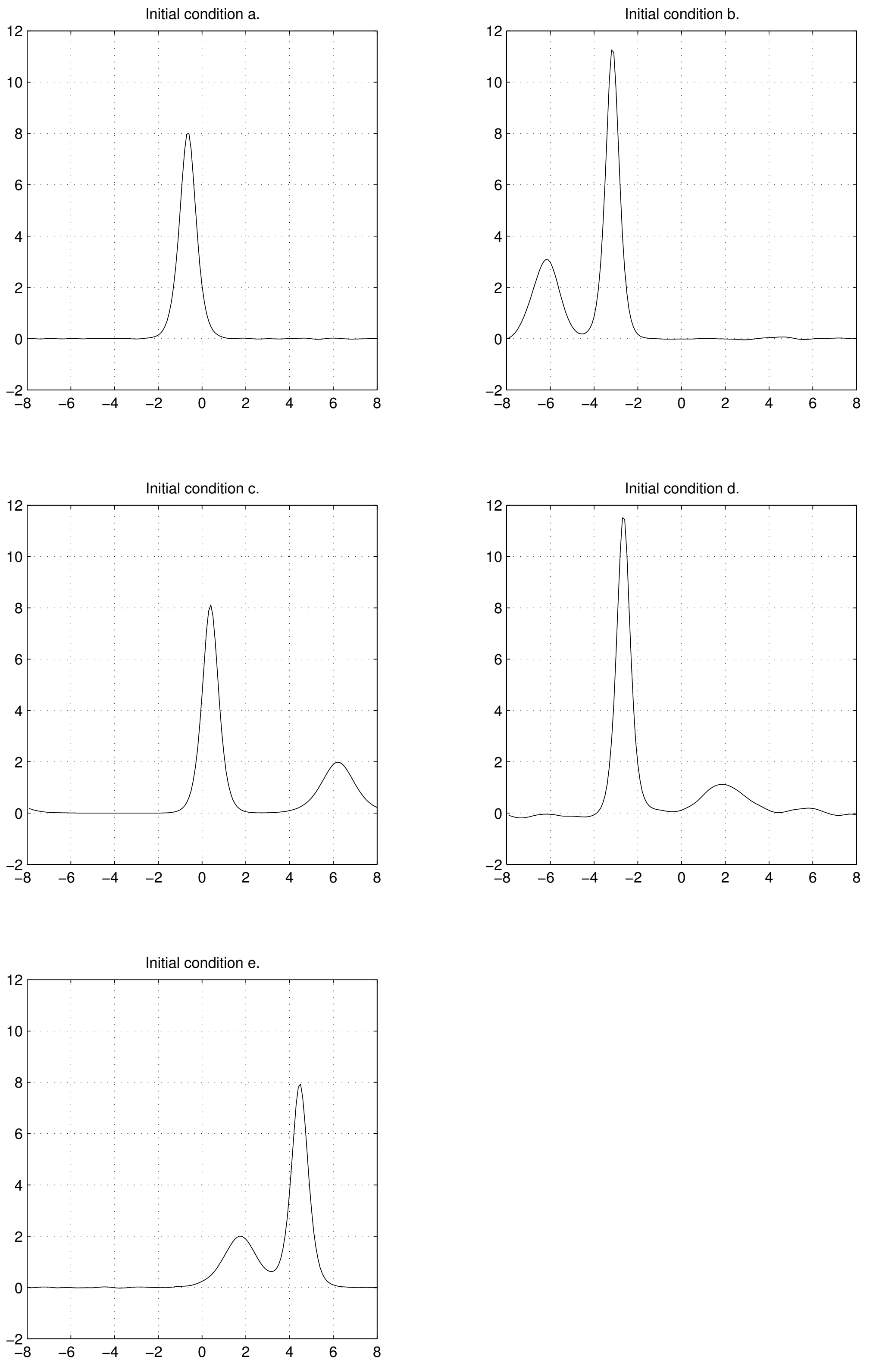


Figure 1: The solutions at time $t = 2$ for the five given initial conditions.

Problem 2 - Traffic Flow

1) The MATLAB® code for the two schemes can be found in Appendix A, `p1a.m` and `p1b.m`. Below are some comments on the implementations.

- The computational grid is defined by

$$x_j = -2 + j\Delta x, \qquad \Delta x = \frac{4}{N}, \qquad j = 0, 1, \dots, N \tag{12}$$

  These positions will correspond to cell centers, and the density of cars ρ will be represented as averages over the cells. Note that the fluxes will be computed between these cell centers, that is, at the cell boundaries.

- No specific boundary conditions were specified, so the implementation simply keeps the initial condition at the boundary points. In practice, this means that the boundary values are used when computing the fluxes, but the values are never updated.

- The numerical flux for Roe's scheme is straightforward to compute. For Godunov's scheme, however, it is not completely trivial. The numerical flux is then given by:

$$F^G_{i+\frac{1}{2}} = \begin{cases} \min_{\rho\in[\rho_i,\rho_{i+1}]} f(q), & \text{if } \rho_i < \rho_{i+1} \\ \max_{\rho\in[\rho_i,\rho_{i+1}]} f(q), & \text{if } \rho_i > \rho_{i+1} \end{cases} \tag{13}$$

  (when $\rho_i = \rho_{i+1}$, the fluxes are equal and any of them can be used). To get an expression for the minimum and the maximum, the flux function $f(\rho)$ must be studied. In our case, it is

$$f(\rho) = \rho u_{\max}\left(1 - \frac{\rho}{\rho_{\max}}\right) \tag{14}$$

  This is a concave parabola with maximum value $\rho_{\max}u_{\max}/4$ at $\rho = \rho_{\max}/2$. The maximums and minimums can then be computed according to

$$\min_{\rho\in[\rho_i,\rho_{i+1}]} f(q) = \min(\rho_i, \rho_{i+1})$$

$$\max_{\rho\in[\rho_i,\rho_{i+1}]} f(q) = \begin{cases} \max(\rho_i, \rho_{i+1}), & \rho_i, \rho_{i+1} \text{ “on same side” of } \rho_{\max}/2 \\ \rho_{\max}u_{\max}/4, & \text{otherwise.} \end{cases}$$

The problems can now be used with the initial values specified in the problem. The solution at time $t = 2$ is shown in figure 2, using Roe's scheme (top) and Godunov's scheme (bottom). With Roe's scheme, the initial discontinuity at $x = 0$ is preserved, and the scheme is therefore not entropy satisfying. This can be seen directly from the expression for the numerical flux:

$$F^R_{i+\frac{1}{2}} = \frac{1}{2}[f(\rho_i) + f(\rho_{i+1})] - \frac{1}{2}\left|a_{i+\frac{1}{2}}\right|(\rho_{i+1} - \rho_i) \tag{15}$$

where

$$a_{i+\frac{1}{2}} = u_{\max}\left(1 - \frac{\rho_i + \rho_{i+1}}{\rho_{\max}}\right) \tag{16}$$

and

$$f(\rho_{i+1}) - f(\rho_i) = a_{i+\frac{1}{2}}(\rho_{i+1} - \rho_i) \tag{17}$$

Consider the cell directly to the left of the red light. The numerical flux at the left end of this cell is $f(\rho_L)$. But at the right end it is exactly the same:

$$\frac{1}{2}[f(\rho_i) + f(\rho_{i+1})] - \frac{1}{2}\frac{|f(\rho_{i+1}) - f(\rho_i)|}{|\rho_{i+1} - \rho_i|}(\rho_{i+1} - \rho_i) = f(\rho_i) = f(\rho_L) \tag{18}$$

The difference in the fluxes is zero, and the density in the cell will be unchanged. This is the reason that the discontinuity is preserved.

We know from the lecture notes that Godunov's scheme is entropy satisfying, and indeed we see the expected rarefaction wave from the shock.

2) This problem is implemented in `p2.m` in Appendix A. The Godunov's scheme from the previous part is used, with some additional logic for inserting a zero flux at the position of the traffic light when it is red. In each timestep, the sum in the average flow is updated:

$$\dot{q} = \frac{1}{N_T}\sum_{n=1}^{N_T} f^n \tag{19}$$

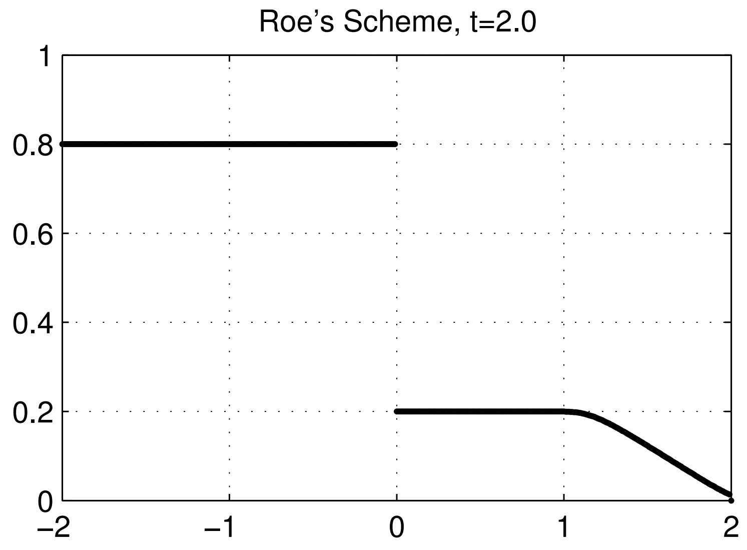
where the flux at the center is used (as described in the problem this choice does not really matter, which can be verified by trying some different positions). After one period, the average flow is printed, and the sum is set to zero.

The density of cars at three times is shown in figure 3. When the light is red, the regions with $\rho = 1$ grows to the left, and when the light is green, the cars accelerate in a similar way as in the previous part.

The average flows after each of the first three periods are:

```
Average flow = 0.19867
Average flow = 0.125
Average flow = 0.125
```

3) The modeling of two traffic lights is done in the code `p3.m` in Appendix A. The second traffic light is handled in the same way as the first, but with different position and switching times. The (converged) average flow for $k = 0, \ldots, 9$ is shown in figure 4, together with another simulation with much finer steps in τ. There are several values of τ giving the maximum capacity 0.125, for example $\tau = 0$ which corresponds to no delay between the traffic lights.



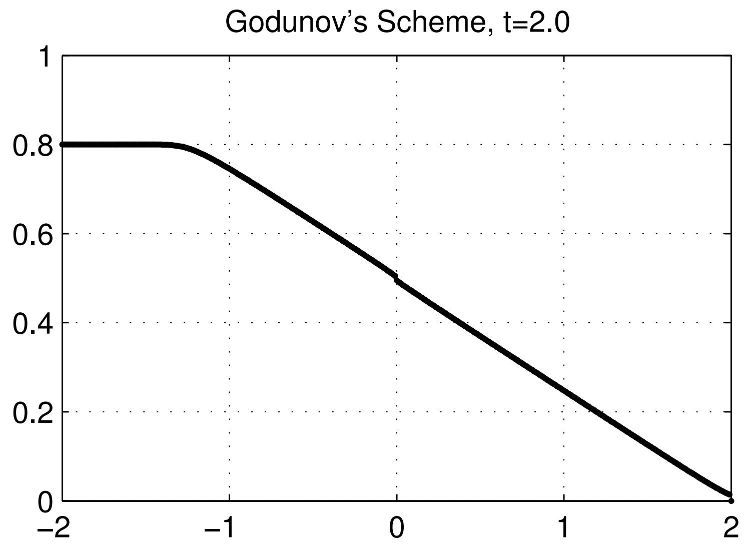


Figure 2: The solution at $t = 2$ using Roe's scheme (top) and Godunov's scheme (bottom). Roe's scheme is not entropy satisfying, since the discontinuity remains, but Godunov's scheme is.

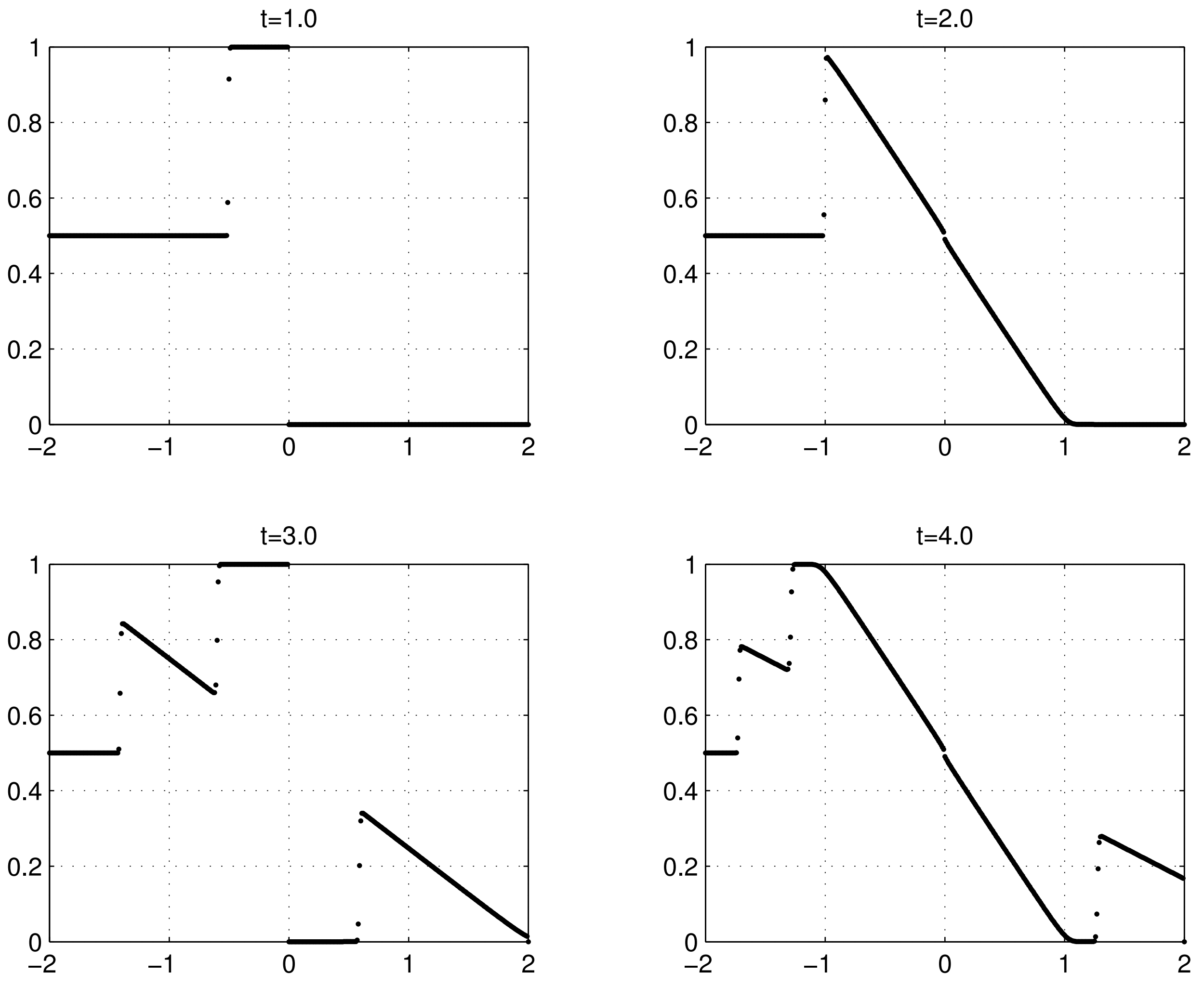


Figure 3: The density of cars at four times. The traffic light at $x = 0$ is red for the first second, green for the next second, etc.

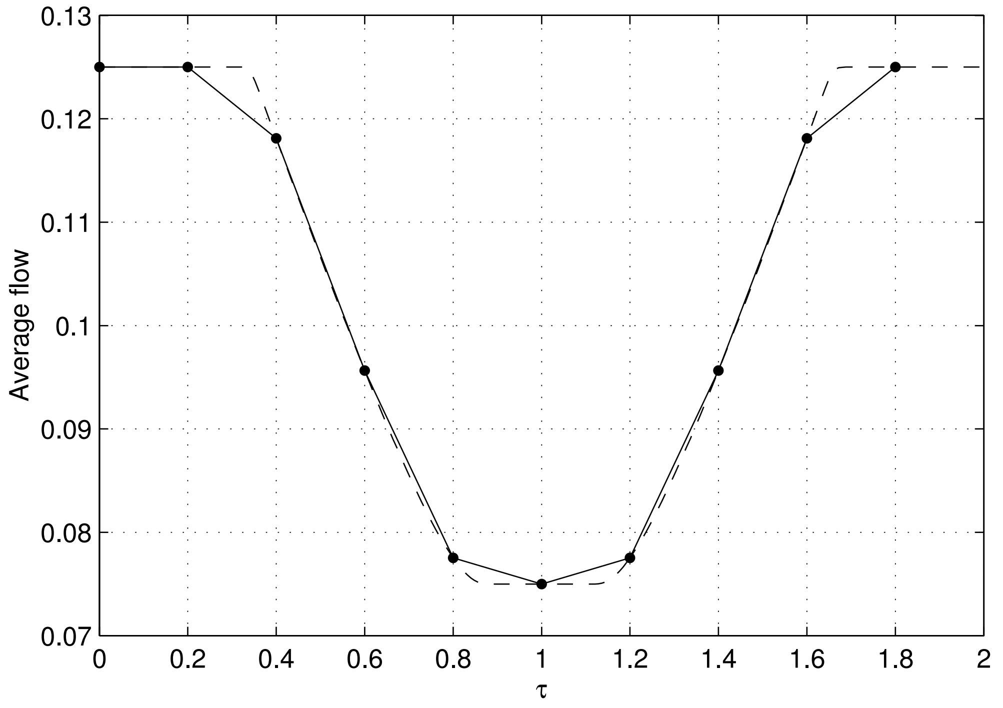


Figure 4: The average flow for different delays τ between the two traffic lights. The dashed curve shows the same quantity as the solid curve, but with a finer resolution.

Appendix A: Code for Part #1

kdv.m

```
function kdv(initcond)

% Space coordinates
dx=0.1;
x=(-8+dx:dx:8)';
nx=length(x);

% Time steps
k=dx^3;
nsteps=2.0/k;

% Initial condition
switch initcond
 case 1
  u=onesoliton(x,16,0);
 case 2
  u=-8*exp(-x.^2);
 case 3
  u=-6./cosh(x).^2;
 case 4
  u=onesoliton(x,16,0)+onesoliton(x,4,0);
 case 5
  u=onesoliton(x,16,4)+onesoliton(x,4,-4);
 otherwise
  error('Incorrect initial condition.');
end

set(gcf,'doublebuffer','on');
for ii=1:nsteps
  % Runge-Kutta step
  k1=k*kdvequ(u,dx);
  k2=k*kdvequ(u+k1/2,dx);
  k3=k*kdvequ(u+k2/2,dx);
  k4=k*kdvequ(u+k3,dx);
  u=u+k1/6+k2/3+k3/3+k4/6;

  % Animate every 10th step
  if mod(ii,10)==0
    plot(x,-u);
    axis([-8,8,-2,12])
    drawnow;
  end
end

function u=onesoliton(x,v,x0)
% One-soliton solution
u=-v/2./cosh(.5*sqrt(v)*(x-x0)).^2;

function dudt=kdvequ(u,dx)
% KdV equation: dudt = 6*u*dudx - d^3u/dx^3
u = [u(end-1:end); u; u(1:2)];
dudt = 6*(u(3:end-2)).*(u(4:end-1)-u(2:end-3))/2/dx - ...
       (u(5:end)-2*u(4:end-1)+2*u(2:end-3)-u(1:end-4))/2/dx^3;
```

Appendix B: Code for Part #2

p1a.m

```
% Problem 1a
% Roe's Scheme

% Parameters
rhomax=1.0;
rhoL=0.8;
umax=1.0;
dx=4/400;
dt=0.8*dx/umax;
nt=2.0/dt;

% Grid and initial conditions
x=-2:dx:2;
rho=rhoL*(x<0.0);

% Main loop
for it=1:nt
  % Flux
  f=rho*umax.*(1-rho/rhomax);
  a=umax*(1-(rho(1:end-1)+rho(2:end))/rhomax);
  F=1/2*(f(1:end-1)+f(2:end))-1/2*abs(a).*(rho(2:end)-rho(1:end-1));

  % Update solution
  rho(2:end-1)=rho(2:end-1)-dt/dx*(F(2:end)-F(1:end-1));

  % Plot
  plot(x,rho,'.')
  axis([-2,2,0,1])
  grid on
  drawnow
end
```

p1b.m

```
% Problem 1b
% Godunov's Scheme

% Parameters
rhomax=1.0;
rhoL=0.8;
umax=1.0;
dx=4/400;
dt=0.8*dx/umax;
nt=2.0/dt;

% Grid and initial conditions
x=-2:dx:2;
rho=rhoL*(x<0.0);

% Main loop
for it=1:nt
  % Flux
  f=rho*umax.*(1-rho/rhomax);
  minf=min(f(1:end-1),f(2:end));
  maxf=max(f(1:end-1),f(2:end));
  maxf((rho(1:end-1)-0.5).*(rho(2:end)-0.5)<0)=rhomax*umax/4;
  F=minf.*(rho(1:end-1)<rho(2:end))+ ...
    maxf.*(rho(1:end-1)>=rho(2:end));

  % Update solution
  rho(2:end-1)=rho(2:end-1)-dt/dx*(F(2:end)-F(1:end-1));

  % Plot
  plot(x,rho,'.')
  axis([-2,2,0,1])
  grid on
  drawnow
end
```

p2.m

```
% Problem 2

% Parameters
rhomax=1.0;
rhoL=rhomax/2;
umax=1.0;
dx=4/400;
dt=0.8*dx/umax;
nt=10.0/dt;

% Grid and initial conditions
x=-2:dx:2;
rho=rhoL*(x<0.0);

% Period, number of timesteps
T=2.0/dt;

% Average flow and initial state
qdot=0.0;
red=1;

% Main loop
for it=1:nt
  % Flux
  f=rho*umax.*(1-rho/rhomax);
  minf=min(f(1:end-1),f(2:end));
  maxf=max(f(1:end-1),f(2:end));
  maxf((rho(1:end-1)-0.5).*(rho(2:end)-0.5)<0)=rhomax*umax/4;
  F=minf.*(rho(1:end-1)<rho(2:end))+ ...
    maxf.*(rho(1:end-1)>=rho(2:end));

  % Red light
  if red
    F(size(F,2)/2)=0;
  end

  % Update solution
  rho(2:end-1)=rho(2:end-1)-dt/dx*(F(2:end)-F(1:end-1));
  % Update average flow
  qdot=qdot+f(size(F,2)/2);

  % Switch state?
  if mod(it,T/2)==0
    if red==0
      disp(['Average flow = ',num2str(qdot/T)]);
      qdot=0.0;
      red=1;
    else
      red=0;
    end
  end

  % Plot
  plot(x,rho,'.')
  axis([-2,2,0,1])
  grid on
  drawnow
end
```

p3.m

```
% Problem 3

% Parameters
rhomax=1.0;
rhoL=rhomax/2;;
umax=1.0;
dx=4/400;
dt=0.8*dx/umax;
nt=10.0/dt;

% Period, number of timesteps
T=2.0/dt;

% Test cases
for k=0:9
  disp(' ')
  disp(['k = ',int2str(k)])
  disp('-----')

  % tau, number of timesteps
  tau=k/10*T;

  % Grid and initial conditions
  x=-2:dx:2;
  rho=rhoL*(x<0.0);

  % Average flow and initial states
  qdot=0.0;
  red1=1;
  red2=tau>T/2|tau==0;

  % Main loop
  for it=1:nt
    % Flux
    f=rho*umax.*(1-rho/rhomax);
    minf=min(f(1:end-1),f(2:end));
    maxf=max(f(1:end-1),f(2:end));
    maxf((rho(1:end-1)-0.5).*(rho(2:end)-0.5)<0)=rhomax*umax/4;
    F=minf.*(rho(1:end-1)<rho(2:end))+ ...
      maxf.*(rho(1:end-1)>=rho(2:end));

    % Red lights
    if red1
      F(size(F,2)/2)=0;
    end
    if red2
      F(size(F,2)/2+.15/dx)=0;
    end

    % Update solution
    rho(2:end-1)=rho(2:end-1)-dt/dx*(F(2:end)-F(1:end-1));

    % Update average flow
    qdot=qdot+f(size(F,2)/2);

    % Switch states?
    if mod(it,T/2)==0
      if red1==0
        disp(['Average flow = ',num2str(qdot/T)]);
        qdot=0.0;
        red1=1;
      else
        red1=0;
      end
    end
    if mod(it-tau,T/2)==0
      if red2==0
        red2=1;
      else
        red2=0;
      end
    end

    % Plot
    plot(x,rho,'.')
    axis([-2,2,0,1])
    grid on
    drawnow
  end
end
```