Feedback on Problem Set 1

Ph.D. Course:
Nodal DG-FEM for solving partial differential equations

Allan P. Engsig-Karup
Scientific Computing Section
DTU Informatics
Technical University of Denmark

August 7, 2012

Feedback On Problem Set 1

Consider the linear advection equation

$$\partial_t u + a\partial_x u = 0, \quad x \in [0, L]$$

with IC and BC conditions

$$u(x,0) = \sin\left(\frac{2\pi}{L}x\right), \quad u(0,t) = -\sin\left(\frac{2\pi}{L}t\right)$$

The exact solution to this problem is given as

$$u(x,t) = f(x-at) = \sin\left(\frac{2\pi}{L}(x-at)\right)$$

where $f(\cdot)$ is an arbitrary initial function.

Feedback On Problem Set 1

```
% Driver script for solving the 1D advection equations
Globals1D;

% Order of polymomials used for approximation
N = 8;

% Generate simple mesh
[Nv, VX, K, EToV] = MeshGen1D(0.0,2.0,10);

% Initialize solver and construct grid and metric
StartUp1D;

% Set initial conditions
u = sin(x);

% advection speed
a = 2*pi;

% numerical flux (stable: 0<=alpha<=1)
alpha = 1;

% CFL constant
CFL=0.75;

% Solve Problem
FinalTime = 10;
[u] = Advec1D(u,FinalTime,a,alpha);
```

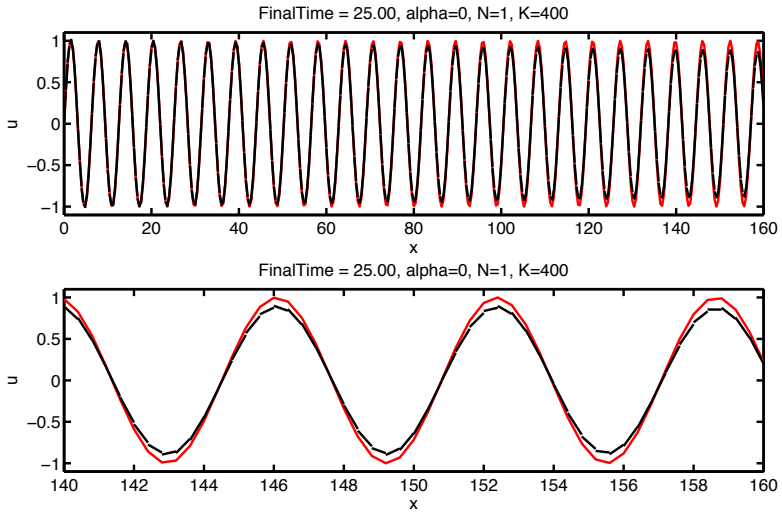

Feedback On Problem Set 1

```
function [u] = Advec1D(u, FinalTime, a, alpha)
% Purpose  : Integrate 1D advection until FinalTime starting with
%            initial the condition, u

Globals1D;
time = 0;

% Runge-Kutta residual storage
resu = zeros(Np,K);

% compute time step size
dxmin = min(abs(x(1,:)-x(2,:)));
dt = CFL*dxmin/a;
Nsteps = ceil(FinalTime/dt); dt = FinalTime/Nsteps;

% outer time step loop
for tstep=1:Nsteps
    for INTRK = 1:5
        timelocal = time + rk4c(INTRK)*dt;
        [rhsu] = AdvecRHS1D(u, timelocal, a, alpha);
        resu = rk4a(INTRK)*resu + dt*rhsu;
        u = u+rk4b(INTRK)*resu;
    end;
    % Increment time
    time = time+dt;

    % Plot solution
    plot(x,u); drawnow;
end;
return
```

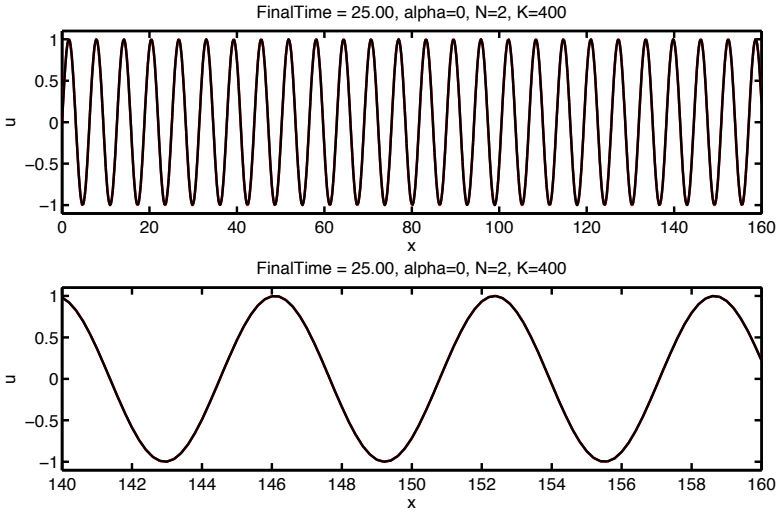
Feedback On Problem Set 1

```
function [rhsu] = AdvecRHS1D(u,time, a,alpha)

% function [rhsu] = AdvecRHS1D(u,time)
% Purpose  : Evaluate RHS flux in 1D advection

Globals1D;

% form flux differences at faces
df    = zeros(Nfp*Nfaces,K);
df(:) = 0.5*a*(u(vmapM)-u(vmapP)).*(nx(:)-(1-alpha));

% impose boundary condition at x=0
uin      = -sin(a*time);
df(mapI) = 0.5*a*(u(vmapI)-uin).*(nx(mapI)-(1-alpha));
df(mapO) = 0;

% compute right hand sides of the semi-discrete PDE
rhsu = -a*rx.*(Dr*u) + LIFT*(Fscale.*(df));
return
```

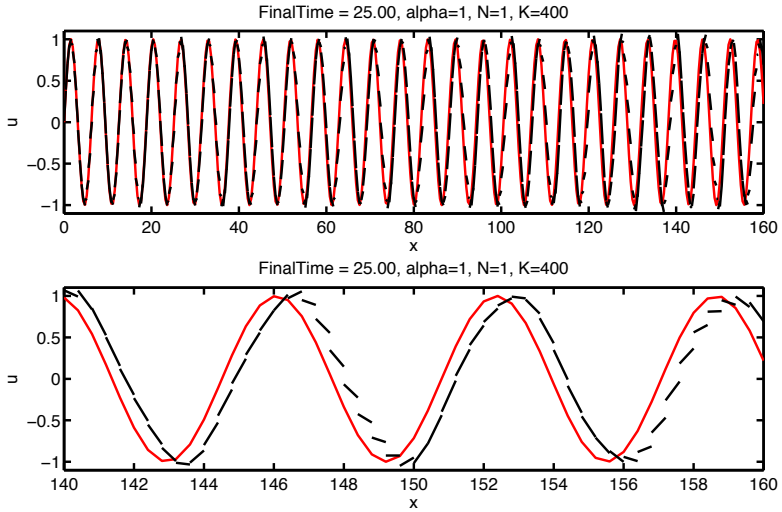

Feedback On Problem Set 1



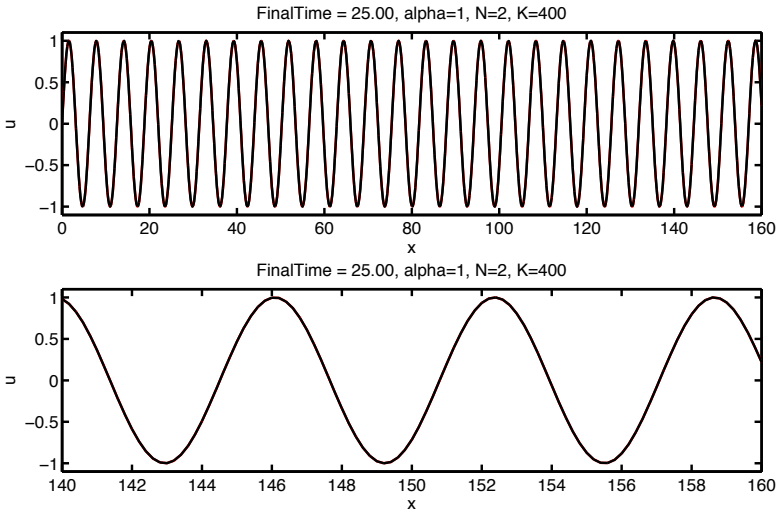
Feedback on Problem Set 1



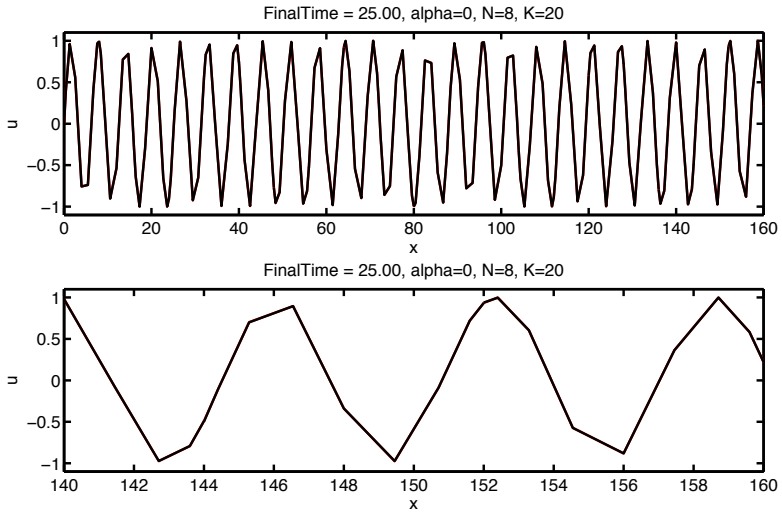
Feedback on Problem Set 1



Feedback on Problem Set 1



Feedback on Problem Set 1

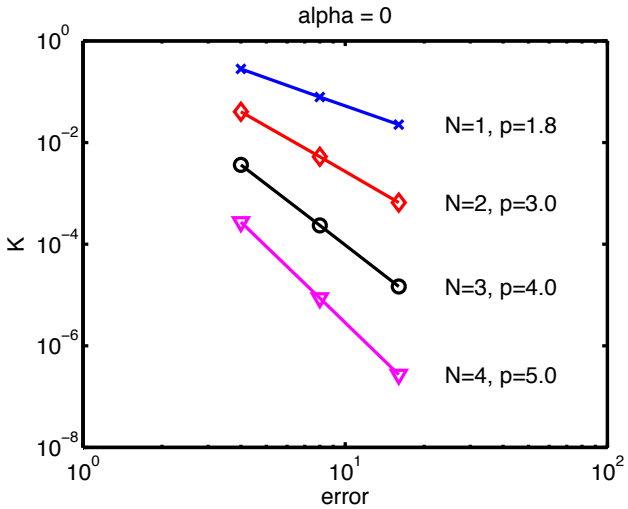


Feedback on Problem Set 1

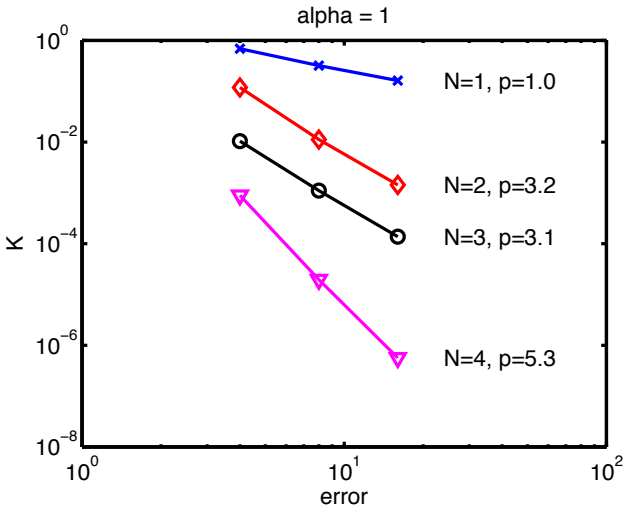
Measuring errors in the L^2 norm

```
ua = sin(x-a*time);

err = ua - u; % compute point-wise error
M = inv(V*V');
for k = 1 : K
    errL2(k) = err(:,k)'*diag(J(:,k))*M*err(:,k);
end
errL2 = sqrt(sum(errL2))
```


Feedback on Problem Set 1



Feedback on Problem Set 1



Feedback on Problem Set 1

$$\Delta t \propto \frac{h}{N^2}$$

- To increase the poly. order near the stability limit we would need to decrease the time step quadratically with 1/N.
- To reduce the size of elements h, i.e. increase K, we would need to reduce the time step linearly.

Feedback on Problem Set 1

Weak formulation

```
close all, clear all, clc
% Driver script for solving the 1D advection equations
Globals1D;

% Order of polymomials and number of elements used for approximation
N = 1;
K = 6;

% Generate simple mesh
[Nv, VX, K, EToV] = MeshGen1D(0.0,2*pi,K);

% Initialize solver and construct grid and metric
StartUp1D;
S = inv(V*V')*Dr; % stiffness matrix
Drw = V*V'*S'; % weak derivative operator matrix (unscaled)

% Set initial conditions
u = sin(x);

% advection speed
a = 2*pi;

% numerical flux (stable: 0<=alpha<=1)
alpha = 0;

% Solve Problem
FinalTime = 2*pi;
[u,time] = Advec1Dweak(u,FinalTime,a,alpha);
```


Feedback on Problem Set 1

Weak formulation

```
function [u,time] = Advec1Dweak(u, FinalTime, a, alpha)
% function [u] = Advec1D(u, FinalTime)
% Purpose  : Integrate 1D advection until FinalTime starting with
%            initial the condition, u
%            based on wek formulation.

Globals1D;
time = 0;

% Runge-Kutta residual storage
resu = zeros(Np,K);

% compute time step size
xmin = min(abs(x(1,:)-x(2,:)));
CFL=0.125*0.75; dt   = CFL/(2*pi)*xmin; dt = .5*dt;
Nsteps = ceil(FinalTime/dt); dt = FinalTime/Nsteps;

% outer time step loop
for tstep=1:Nsteps
    for INTRK = 1:5
        timelocal = time + rk4c(INTRK)*dt;
        [rhsu] = AdvecRHS1Dweak(u, timelocal, a, alpha);
        resu = rk4a(INTRK)*resu + dt*rhsu;
        u = u+rk4b(INTRK)*resu;
    end;
    % Increment time
    time = time+dt;
    if mod(tstep,10)==0
    plot(x,u,'b.-')
    axis([0 max(x(:)) -1.25 1.25])
    drawnow
    end
end
return
```

Feedback on Problem Set 1

Weak formulation

```
function [rhsu] = AdvecRHS1Dweak(u,time,a,alpha)
% function [rhsu] = AdvecRHS1Dweak(u,time)
% Purpose  : Evaluate RHS flux in 1D advection
Globals1D;

% form numerical flux at faces
df    = zeros(Nfp*Nfaces,K);
df(:) = 0.5*a*nx(:).*(u(vmapM)+u(vmapP)); % central

% impose boundary condition at x=0
uin      = -sin(a*time);
df(mapI) = 0.5*a*nx(mapI).*(u(vmapI)+uin); % central
df(mapO) = 0.5*a*nx(mapO).*(u(vmapO)*2); % central

% compute right hand sides of the semi-discrete PDE
rhsu = +a*rx.*(Drw*u) - LIFT*(Fscale.*(df));
return
```


Feedback on Problem Set 1

The "energy method" can be used to show that PDEs are well-posed and solutions are stable (finite "energy").

Consider the linear advection equation

$$u_t + au_x = 0, \quad x \in [-1, 1], \quad a > 0$$

subject to initial conditions $u(x, 0) = f(x)$ and periodic boundary conditions $u(-1, t) = u(1, t)$. The problem is well-posed if the "energy" of the solution can be bounded by the initial data

$$||u(\cdot, t)|| \leq Ce^{\alpha T}||f(\cdot)||, \quad C, \alpha > 0$$

We can try and show that this is the case by considering the L^2 norm $||u||^2 = \int_{-1}^{1} u^2 dx$

$$\frac{\partial}{\partial t}||u||^2 = 2\int_{-1}^{1} u_t u dx = -2\int_{-1}^{1} au_x u dx$$
$$= -\int_{-1}^{1} a(u^2)_x dx = -a\left(u(1)^2 - u(-1)^2\right)$$