From Lect. 19, the Fourier coefficients for the spectral soln to advection equation obey:

$$\frac{d\hat{q}_n}{dt} = -i\omega_n\hat{q}_n, \quad \omega_n = k_n c, \quad k_n = \frac{2\pi}{L}[0,\ldots,\frac{N}{2}-1,-\frac{N}{2},\ldots,-1] \text{ for } n = 1,\ldots,N.$$

We can solve this set of equations using one of the previously discussed time-differencing methods. To decide which method might be optimal, suppose we use a q'th order accurate time differencing method. Fourier spectral (FS) methods can evaluate the space derivative operator very accurately. For instance, if the space derivative error decays exponentially with $N = L/\Delta x$, the overall solution error for a smooth initial condition after some finite integration time T will be

$$\varepsilon = ae^{-\alpha N} + b\Delta t^q \qquad \text{(E)}$$

where a and b are coefficients that depend on T and the initial condition. The computation will take $T/\Delta t$ timesteps, each taking O($N \log N$) flops. An efficient approach is to choose $a \exp(\alpha N)$ and $b\Delta t^q$ to be O(ε). Hence **it is most efficient to pair a FS method with an accurate time-differencing method such as RK4** for which the desired accuracy can be achieved with a relatively large timestep Δt.

Stability of FS+RK4 on the advection equation

The RK4 stability limit for oscillations is

$$\omega_{\max}\Delta t < 2.82$$

The highest frequency that must be stepped forward is:

$$\omega_{\max} = \max_n |\omega_n| = c\pi N/L = c\pi/\Delta x$$

Thus the FS+RK4 method is stable if

$$c\Delta t/\Delta x < 2.82/\pi \approx 0.9$$

Unlike for the finite difference and finite volume methods we discussed, it may not be most efficient to use a timestep close to the stability threshold. For instance, if $c = L = 1$, the desired error $\varepsilon \sim 10^{-8}$ and all the coefficients in the error formula are assumed to be O(1), we might choose

$$N \approx |\log \varepsilon| = |\log 10^{-8}| \approx 20 \Rightarrow \Delta x = N^{-1} = 0.05$$

$$\Delta t \approx \varepsilon^{1/4} = 10^{-2} \quad \Rightarrow \quad c\Delta t/\Delta x = 0.2 \text{ (much smaller than stability threshold).}$$

Plotting a Fourier spectral solution between gridpoints

Even with a coarse grid spacing, a DFT can give a remarkably accurate representation of a smooth function. It can be useful to plot that representation between grid points:

$$q(x) = N^{-1}\sum_{n=1}^{N} \hat{q}_n e^{iK_n x}$$

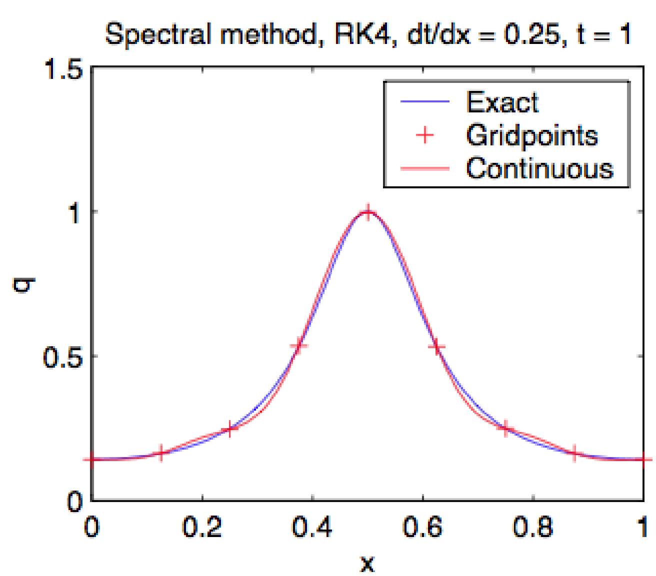
While we could just define a set of gridpoints and do this sum at each gridpoint, there is a convenient shortcut that uses the DFT. Let us define a uniform fine grid with N_f gridpoints, where N_f is a multiple of N.

$$x_i^f = (i-1)\Delta x_f, \quad \Delta x_f = L/N_f, \quad i = 1,\ldots,N_f$$

Then we can calculate the vector qf of values $q_i = q(x_i^f)$ on the fine grid as the IDFT of the vector of wavenumbers $\hat{q}_n$ padded with zeros for all of the newly added wavenumbers. This is implemented with the Matlab call `qf = interpft(q,Nf)` where q is the N-vector of coarse-grid values and qf the N_f-vector of fine grid values.

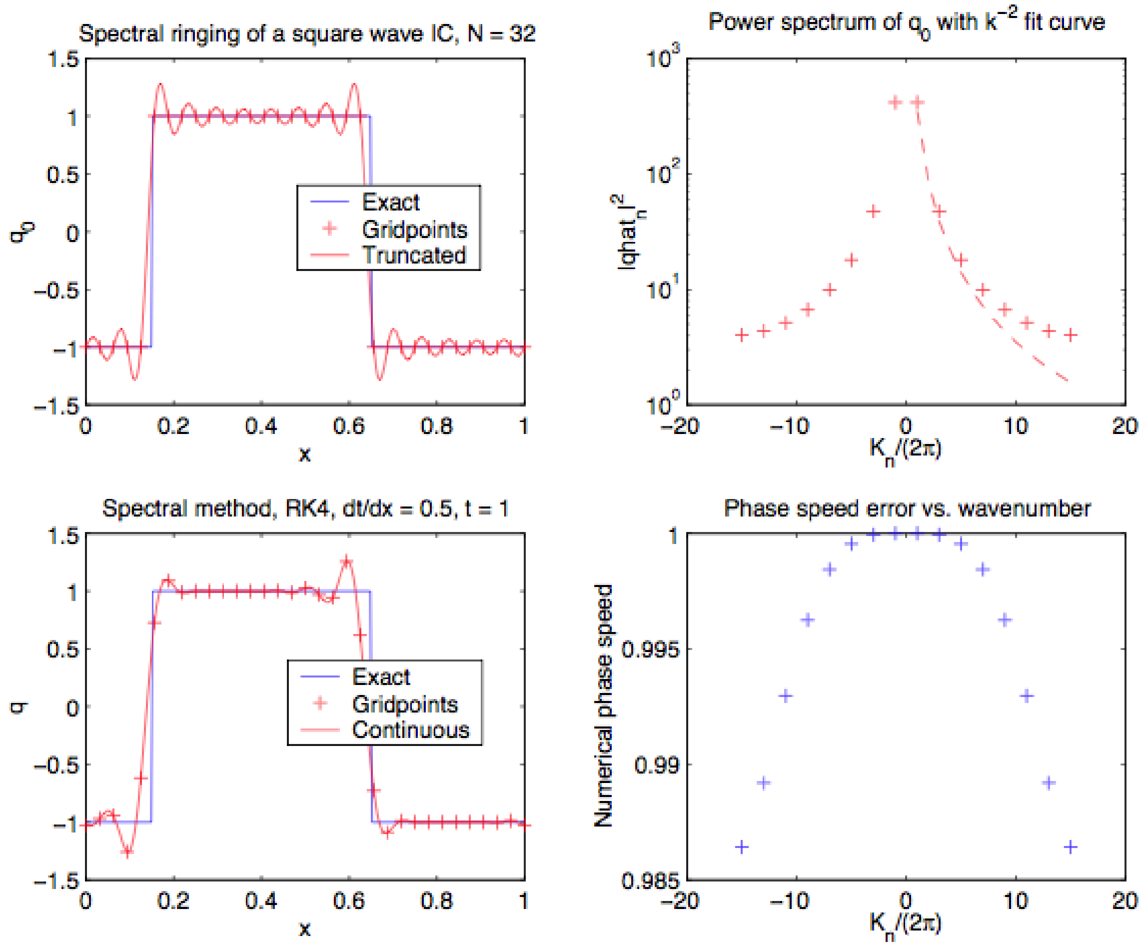
Matlab script for spectral method for advection eqn.

```
% Numerically calculate soln. to advection eqn. dq/dt + dq/dx = 0 on
% domain 0<x<1 with periodic BCs using spectral method with RK4

 N = 8; % Number of modes
 nu = 0.25;  % Courant number
 L = 1;  % Domain size
 x = L*(0:(N-1))/N;  %  x-gridpoints [1xN]
 dx = L/N;
 M = [0:(N/2-1) (-N/2):(-1)];
 k = 2*pi*M/L;  % Wavenumbers [1xN].

 q0 = 1./(4+3*cos(2*pi*x/L));  % Initial condition
 tf = 1; % Final time
 dt = nu*dx; % Timestep
 nt = round(tf/dt); % Number of timesteps to take

 qhat = fft(q0);  % Initial Fourier expansion coeffs [1xN].
 for it = 1:nt
   % March forward  dqhat/dt = -Shat using RK4
   % where Shat is DFT of S(q) = dq/dx
   d1 = -dt*1i*k.*qhat;
   d2 = -dt*1i*k.*(qhat + 0.5*d1);
   d3 = -dt*1i*k.*(qhat + 0.5*d2);
   d4 = -dt*1i*k.*(qhat + d3);
   qhat = qhat + (d1 + 2*d2 + 2*d3 + d4)/6; % New qhat
 end
 q = ifft(qhat); % Numerical q at tf [1xN]

 Nf = 256;   % Number of plotting points
 xf = (0:(Nf-1))*L/Nf;
 qf = interpft(q,Nf);  % Numerical solution on plotting grid
 q0f = 1./(4+3*cos(2*pi*xf/L));  % Initial condition
 plot([xf L],[q0f q0f(1)],'b-',[x L],[q q(1)],'r+',[xf L],[qf qf(1)],'r-')
```



Spectral method for scalar advection eqn. - square wave initial condition

$q(x,0) = \mathrm{sign}(0.25 - |x - 0.4|),\ 0 < x < 1$



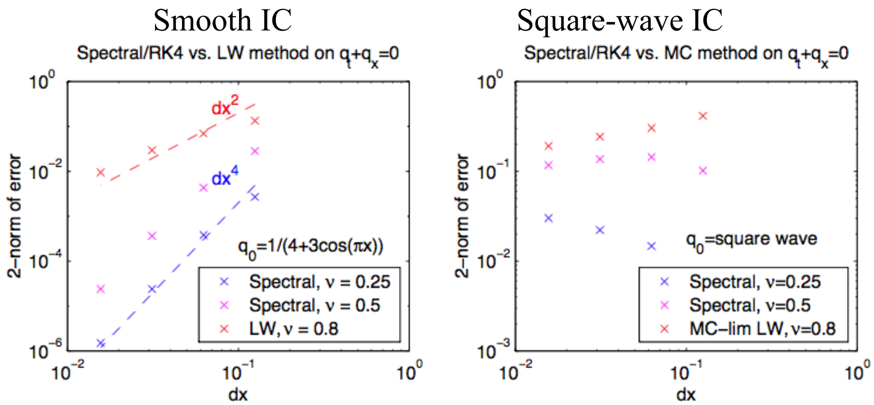
• There are 'Gibbs oscillations' near the discontinuities when $q(x,0)$ is truncated to N complex Fourier modes, with maximal overshoots of around 20%. The oscillations are compressed to a smaller region for larger N, but are not diminished in amplitude.

• The high wavenumbers now decrease much more slowly in amplitude than for smooth initial conditions.

• For this problem, if we time-differenced perfectly, the numerical solution at the grid-points would be exact at all times despite the oscillations in between. However, the RK4 time-differencing scheme creates errors in the phase speeds of each wavenumber which increase with $|\omega_n \Delta t|^4$ (where here frequency $\omega_n = K_n$). For large Δt, the phase-speed errors can be significant for the highest wavenumbers. The result is that the Gibbs oscillations start spreading to the gridpoint values as well.

• Since the numerical phase speed of high wavenumbers is too slow, the square wave doesn't propagate quite as fast as it should.

Error convergence (compared to FV methods)



L19 • We did the time differencing on the Fourier coeffs $\hat{q}_n(t)$. However, a precisely equivalent approach (called the <u>pseudospectral</u> method) would be: to work in terms of gridpoint values Q_j and set

$$\frac{dQ_j}{dt} = -S_j(\vec{Q}) \qquad S(q) = \bar{u} q_x$$

where S_j is evaluated using spectral approximations to derivatives:

$$\{\hat{q}_n(t)\} = \text{DFT}\{Q_j(t)\}$$

$$\{S_j\} = \{(\bar{u}q_x)_j\} = \bar{u} \cdot \text{IDFT}\underbrace{\{ik_n \hat{q}_n(t)\}}_{\text{spectral } q_x}$$

This is a system of coupled ODEs in the Q_j which we can solve using RK4, AB3, leapfrog, etc. For this particular problem, this involves extra work (1 DFT/1 IDFT per timestep) but can be very attractive if $S(q)$ has nonlinearities or x-dependent coeffs.

Pseudospectral (PS) method for KdV eqn.

$$q_t + S(q) = 0 \qquad S(q) = 6qq_x + q_{xxx}$$

$\Rightarrow$

$$\frac{dQ_j}{dt} = -S_j$$

where to find S_j: (1) $\{\hat{q}_n\} = \mathrm{DFT}\{Q_j\}$

(2) $\{(q_x)_j\} = \mathrm{IDFT}\{iK_n\hat{q}_n\}$

$\{(q_{xxx})_j\} = \mathrm{IDFT}\{-iK_n^3\hat{q}_n\}$

(3) $S_j = 6Q_j(q_x)_j + (q_{xxx})_j$

We use our favorite time-differencing scheme to solve.

Note that for the highest wavenumbers, the $\frac{d^3}{dx^3}$ term brings in a factor $(iK_n)^3$, so we can imagine that if we worked in Fourier space,

$$\frac{d\hat{q}_n}{dt} = \text{stuff} + (iK_n)^3\hat{q}_n$$

$$= \text{stuff} + i\omega_n\hat{q}_n \quad , \quad \omega_n = -K_n^3$$

Thus for RK4 the stability limit is

$$2.82 > \max_n |\omega_n \Delta t| = \max_n |K_n^3 \Delta t| = \max_n \left|\left(\frac{\pi N}{L}\right)^3 \Delta t\right|$$

i.e. $$\Delta t < \Delta t_c = \frac{2.82}{\pi^3}\Big(\underbrace{\frac{L}{N}}_{\Delta x}\Big)^3 \approx 0.09(\Delta x)^3$$

This is a severe restriction if we have lots of modes, but there is not a good workaround (BDF, for backward differencing formula) methods are an option, but even then, if we use a much larger Δt than this we will lose accuracy. Example shown has $\Delta t = 0.05\,\Delta x^3$.

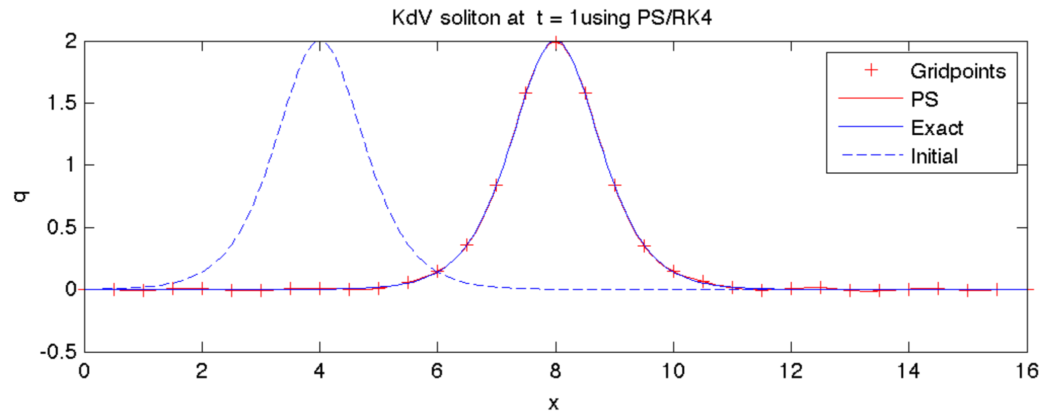
This KdV eqn has "soliton" nonlinear traveling wave solns

$$q(x,t) = a\,\mathrm{sech}^2 b(x-ct) \quad , \quad b = \left(\frac{a}{2}\right)^{\frac{1}{2}} , \quad c = -2a$$

good for testing the PS method in action. In addition, solitons interact with phase shifts as they collide; this can also be simulated.

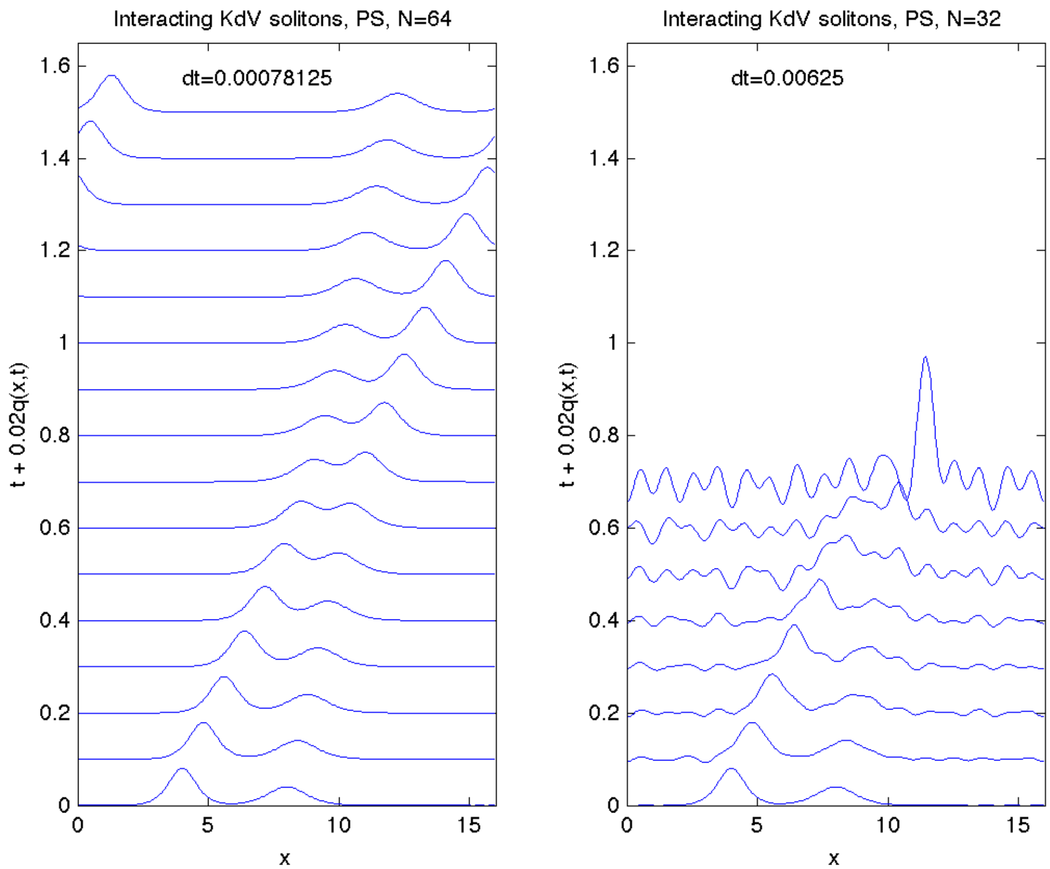
Pseudospectral method for KdV soliton
Matlab script ps_KdV_RK4.m

```
N = 32; % Number of Fourier modes
L = 16;  % Domain size
C = .05; % Nondimensional timestep parameter(gives dt = 0.05 for L/N=1)
dt = C*(L/N)^3; % Timestep limit from qxxx term
x = L*(0:(N-1))/N;  %  x-gridpoints [1xN]
a =  2; % soliton amplitude
b =  sqrt(a/2); % inverse of soliton width
xm = 0.25*L; % initial soliton center point
q = a*sech(b*(x-xm)).^2;
for it = 1:nt
  d1 = -dt*S_KdV(q,L);
  d2 = -dt*S_KdV(q + 0.5*d1,L);
  d3 = -dt*S_KdV(q + 0.5*d2,L);
  d4 = -dt*S_KdV(q + d3,L);
  q = q + (d1 + 2*d2 + 2*d3 + d4)/6; % q marched forward dt
end

function S = S_KdV(q,L)
  N = length(q);
  qhat = fft(q);
  M = [0:(N/2-1) (-N/2):(-1)];
  k = 2*pi*M/L;  % Wavenumbers [1xN].
  qx = real(ifft(1i*k.*qhat));
  qxxx = real(ifft(-1i*k.^3.*qhat));
  S = 6*q.*qx + qxxx;
```



- Note slight dispersive ripples for $11 < x < 15$ due to under-resolution of IC
- Solution stable for $C = 0.10$ but not $C = 0.11$ (compare to theoretical limit $C_{max} = 0.09$)
- Max Courant number $6q_{max}\Delta t/\Delta x = 12(.00625)/(0.5) = 0.15 < \nu_{max}=2.82/\pi = 0.9$ so the dispersive term, not the nonlinear term, is what limits timestep. For a larger-amplitude soliton we could run into CFL problems with this timestep (as well as resolution problems with this number of modes.)

Pseudospectral two-soliton solution
Matlab script ps_KdV_2soliton.m



- $N = 32$ develops nonlinear instabilities due to underresolution of interacting solitons.
- This instability persists for a much smaller Δt, so not due to CFL or linear dispersion
- Neither individual soliton is numerically unstable with $N = 32$ at this Δt.