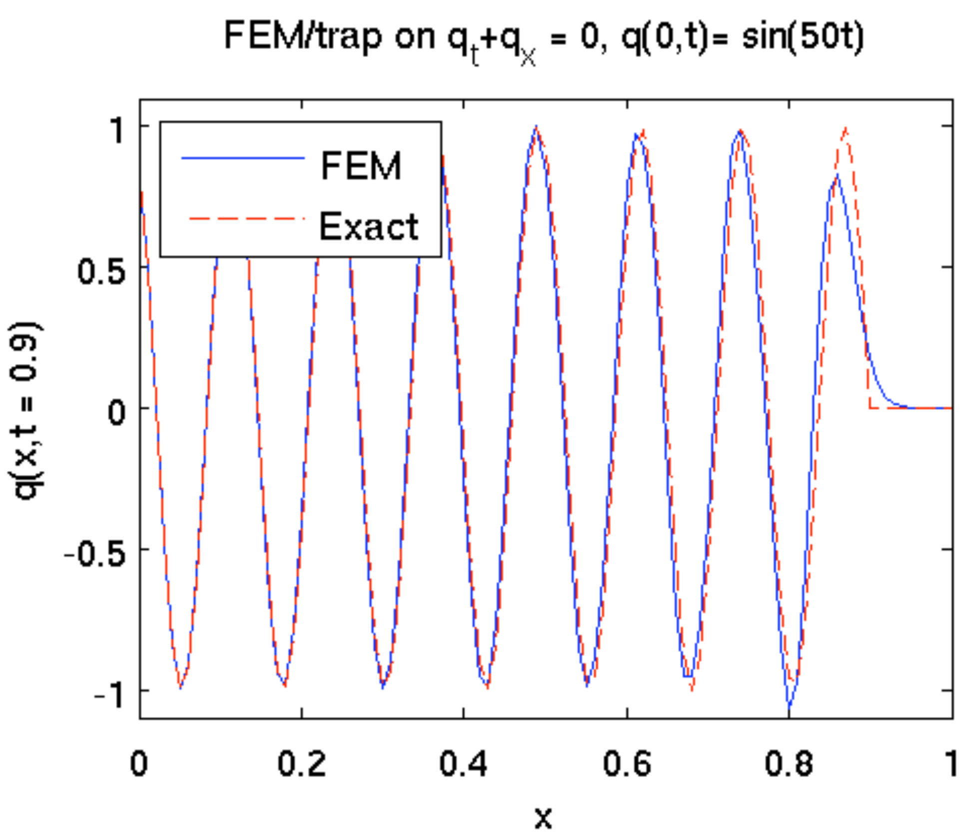
```
function [qT,qr] = advect_FEMsolve(x,c,dt,q0,ql)

%        dq/dt + c(x)*dq/dx = 0, 0<x<1, 0<t
%        q(x(1:nx),0) = q0(x),
%        q(0,t) = ql(t)

  nt = length(ql)-1; % Number of timesteps to take
  nx = length(x) - 1;  % Number of unknowns
  dx = diff(x)'; % Turns dx into a column vector,

  %  We set up the FEM in matrix notation for simplicity:
  %
  %    I*(q(:,n+1)-q(:,n))/dt + 0.5*J*(q(:,n+1)+q(:,n))= r10 (j=1)
  %                                                     0   (j>1)
  %
  %  Here I(j,n)=<phij(x),phin(x)>, J(j,n) = <phij(x),c*dphin(x)/dx>
  %  where j,n = 1,...,N. The right hand side comes from left BC:
  %  r10  = -I(1,0)*(q0(n+1)-q0(n))/dt - 0.5*J(1,0)*(q0(n+1)+q0(n));

  Ijjm1 = dx/6;
  Ijjp1 = [Ijjm1(2:nx); 0]; % Last entry handles right bdry j=nx,
  Ijj   = 2*(Ijjm1 + Ijjp1);
  I = (spdiags([Ijjp1 Ijj Ijjm1],-1:1,nx,nx))'; % Specify diags of I
                                                % to start on row 1.
  cjm1 = c(1:nx); % Note c(j+1) corresponds to index j
  cj = c(2:nx+1);
  cjp1 = [c(3:nx+1) NaN];

  Jjjm1 = -(cjm1/6 + cj/3)';
  Jjjp1 = (cj/3 + cjp1/6)'; % Note J_nx,nx+1 is not used
  Jjjp1(nx) = 0; % Takes care of right boundary
  Jjj   = -(Jjjm1 + Jjjp1);
  e = ones(nx,1);
  J = (spdiags([Jjjp1 Jjj Jjjm1],-1:1,nx,nx))'; % ...as for I

  M = I/dt +0.5*J;

  I10 = Ijjm1(1);
  J10 = Jjjm1(1);

  % Timestepping loop

  q = q0; % Initial condition

  for np1 = 1:nt
    tnp1 = np1*dt;
    rhs =   (I/dt -0.5*J)*q;
    r10  = -I10*(ql(np1+1)-ql(np1))/dt -
0.5*J10*(ql(np1)+ql(np1+1)); % left BC
    rhs(1) = rhs(1)+ r10;

    q = M\rhs; % Update q to new time level
    qr(np1+1) = q(nx); % Store q at right boundary
  end
```

```
% advect_FEM.m: Driver for advect_FEM_solve for
%       q(x,0) = 0, q(0,t) = sin(50*t).
% on uniform grid with dx = 1/nx. Solution plotted at T=0.9.

  c = 1;  % Uniform advection speed
  nx = 100;
  mu = 0.5;   % Courant number
  T = 0.9;
  L = 1;  % Domain size

  dx = L/nx;
  dt = mu*dx;
  nt = round(T/dt);
  t = dt*(0:nt);
  x = dx*(0:nx);
  q0 = zeros(nx,1);  % Zero initial condition (as column vector)
  ql = sin(50*t);  % Left BC

  [q,qr] = advect_FEMsolve(x,c,dt,q0,ql);
```



Spatial truncation error analysis for uniform grid spacing Δx

We can evaluate the local truncation error at each nodal point x_j as with a FDA, plugging the exact solution $\psi(x,t)$ into (*), noting $a_j = \psi(x_j,t) \equiv \psi_j(t)$, and Taylor-expanding:

$$LTE_j = \frac{d}{dt}\left(\frac{\psi_{j+1} + 4\psi_j + \psi_{j-1}}{6}\right) + c\left(\frac{\psi_{j+1} - \psi_{j-1}}{2\Delta x}\right)$$

$$= \frac{d}{dt}\left(\frac{6\psi_j + 2\frac{\Delta x^2}{2}\psi_{j\,xx} + 2\frac{\Delta x^4}{24}\psi_{j\,xxxx}\cdots}{6}\right) + c\left(\frac{2\Delta x\,\psi_{jx} + 2\frac{\Delta x^3}{6}\psi_{j\,xxx} + 2\frac{\Delta x^5}{120}\psi_{j\,xxxxx}\cdots}{2\Delta x}\right)$$

$$= \left(\psi_{j} + \frac{\Delta x^2}{6}\psi_{j\,xx} + \frac{\Delta x^4}{12}\psi_{j\,xxxx}\cdots\right)_t + c\left(\psi_{jx} + \frac{\Delta x^2}{6}\psi_{j\,xxx} + \frac{\Delta x^4}{120}\psi_{j\,xxxxx}\cdots\right) = O(\Delta x^4)$$

$\Rightarrow$ 4th order spatial accuracy at nodes! (but only Δx^2 between nodes and hence in L^2 norm due to linear interpolation inherent in basis functions). Can be interpreted as equivalent to "compact differencing", i.e.

$$\frac{1}{6}\left\{\left(\frac{\partial f}{\partial x}\right)_{j-1} + 4\left(\frac{\partial f}{\partial x}\right)_j + \left(\frac{\partial f}{\partial x}\right)_{j+1}\right\} = \underbrace{\delta_{2x} f_j}_{\left(\frac{f_{j+1} - f_{j-1}}{2\Delta x}\right)} + O(\Delta x^4),$$

giving a way to find $\left(\frac{\partial f}{\partial x}\right)_j$ to $O(\Delta x^4)$ accuracy using just standard centered differences using an implicit formula.

Von Neumann analysis of FEM space differencing (uniform Δx)

Look for growth rate σ of Fourier mode $\phi(x, t) = \exp(ikx + \sigma t)$:

$$\sigma\left\{\frac{e^{ik\Delta x} + 4 + e^{-ik\Delta x}}{6}\right\} + c\left\{\frac{e^{ik\Delta x} - e^{-ik\Delta x}}{2\Delta x}\right\} = 0$$

$$\sigma = -i\omega,\ \omega = ckF(k\Delta x), \quad F(\kappa) = \frac{3\sin(\kappa)}{\kappa[2+\cos(\kappa)]} = 1 - \frac{\kappa^4}{180} + O(\kappa^6).$$

For a given Δx, $\omega_{max} = \max_k \omega = \frac{c}{\Delta x}\max_\kappa \kappa F(\kappa) = \frac{c}{\Delta x}3^{-1/2}$ (max at $\kappa = 2\pi/3$). If FEM is used with a timestepping method with stability limit $\omega_{max}\Delta t = s_{max}$ then the CFL-like stability condition is

$$\mu = c\Delta t/\Delta x < 3^{-1/2}s_{max}$$

(i.e. $3^{-1/2} \approx 0.58$ for leapfrog, $2.82(3^{-1/2}) \approx 1.63$ for RK4., etc.)