Lecture 3: Applications of Fourier Spectral Method

1 Korteweg-de Vrices (KdV) Equation

The KdV equation

$$u_t + u\,u_y + u_{yyy} = 0, \quad y \in (-\infty, \infty), \quad t > 0,$$

$$u(y, 0) = u_0(y), \qquad y \in (-\infty, \infty),$$

has an exact soliton solution

$$u(y, t) = 12\,\kappa^2 \mathrm{sech}^2(\kappa(y - y_0) - 4\kappa^3 t), \tag{1}$$

where y_0 is the center of the initial profile $u(y, 0)$, κ is a constant related to the travelling phase speed.

1.1 Discretization

1. Truncate the computational domain

$$y \in (-\pi L, \pi L), \quad u(y, t) = u(y + 2\pi L, t).$$

2. Mapping the interval $[-\pi L, \pi L]$ to $[0, 2\pi]$ by

$$x = \frac{y}{L} + \pi, \quad y = L(x - \pi), \quad x \in [0, 2\pi], \quad y \in [-\pi L, \pi L].$$

   Let $v(x, t) = u(y, t)$, $v_0(x) = u_0(y)$, then transformed KdV equation reads

$$\begin{array}{l} v_t + \dfrac{1}{L} v\,v_x + \dfrac{1}{L^3} v_{xxx} = 0, \quad x \in (0, 2\pi), \quad t > 0, \\ v(\cdot, t) \text{ periodic on } [0, 2\pi], \quad t \geqslant 0; \quad v(x, 0) = v_0(x), \quad x \in [0, 2\pi]. \end{array} \tag{2}$$

3. Solving (2) by Fourier method.

   Writting $v(x, t) = \sum_{|k| \leqslant N/2} \hat{u}_k(t) e^{i\,k\,x}$, taking the innner product of the first equation in (2) with $e^{i\,k\,x}$, and using the fact $v\,v_x = \frac{1}{2}(v^2)_x$, we obtain that

$$\frac{\mathrm{d}\hat{v}_k(t)}{\mathrm{d}t} - \frac{i\,k^3}{L^3} \hat{v}_k + \frac{i\,k}{2L} \widehat{(v^2)}_k = 0, \quad k = 0, \pm 1, \ldots, \pm N/2. \tag{3}$$

   with the initial condition

$$\hat{v}_k(0) = (v_0(x), e^{i\,k\,x}) = \frac{1}{2\pi} \int_0^{2\pi} v_0(x)\, e^{-i\,k\,x}\, \mathrm{d}x. \tag{4}$$

   We solve the ODE system (3) and (4) by a 4th order Runge-Kutta method with integrating factor. Denote $A = i\,k/L$, equation (3) has an integrating factor $g(t) = e^{A^3 t}$, and can be transformed to

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[e^{A^3 t} \hat{v}_k\right] = -\frac{A}{2} e^{A^3 t} \widehat{(v^2)}_k, \quad k = 0, \pm 1, \ldots, \pm N/2. \tag{5}$$

   Let $w_k = e^{A^3 t} \hat{v}_k$, we get

$$\frac{\mathrm{d}w_k}{\mathrm{d}t} = -\frac{A}{2} e^{A^3 t} \widehat{(v^2)}_k, \quad \hat{v}_k = w_k\, e^{-A^3 t}, \quad k = 0, \pm 1, \ldots, \pm N/2. \tag{6}$$

RK4 for equation

$$w'(t) = f(t, w)$$

reads

$$\begin{cases} w^{n+1} &= w^n + \dfrac{h_1 + 2\,h_2 + 2\,h_3 + h_4}{6}, \\ h_1 &= \delta t\, f(t^n, w^n) \\ h_2 &= \delta t\, f(t^n + \delta t/2, w^n + h_1/2), \\ h_3 &= \delta t\, f(t^n + \delta t/2, w^n + h_2/2), \\ h_4 &= \delta t\, f(t^n + \delta t, w^n + h_3). \end{cases}$$

Remark 1. In general, $\hat{v}_k(0)$ is usually calculated by FFT, which is not exactly equal to (4), the error between a discrete Fourier transform and a continuous one is controlled by the aliasing error. Similar situation happens for the nonlinear term ,

$$\widehat{(v^2)}_k = \frac{1}{2\pi} \int_0^{2\pi} v^2\, e^{-ikx}\, \mathrm{d}x.$$

Since $v \in X_N$: =span$\{e^{ikx}$: $k = 0, \pm 1, \ldots, \pm N/2\}$, so $v^2 \in X_{2N}$, and $\{v^2\ e^{-ikx},\ \ k = 0, \pm 1, \ldots, \pm N/2\} \in X_{3N}$. the discrete numerical integration formula with N point is accurate only for X_{2N}, which means if we use a discrete Fourier transform with N point to calculate $\widehat{(v^2)}_k$, then there will be some aliasling error. To get rid of the aliasing error, we at least need a quadrature with $3N/2$ points, or Fourier transform with $3N/2$ grid points. When u is very smooth, the aliasing error will not be very big, so an algorithm without anti-aliasing is acceptable.

1.2 Implementation

We use Matlab(octave) to implement the algorithm.

1. The initial solution $\hat{v}_k(0)$ in (4) is calculated by fft with length N. $\tilde{v} = \mathrm{fft}(v_0, N)$.

2. For given n, $\hat{v}_k(t^n)$, use RK4 to solve equation (5).

```
function [tdata udata] = KdVsolu(uex, N, tmax, dt, nplot)
   x = (2*pi/N)*(-N/2:N/2-1)';
   u = feval(uex, x, 0); v = fft(u);
   k = [0:N/2-1 0 -N/2+1:-1]'; ik3 = 1i*k.^3;
   nplt = floor((tmax/nplot)/dt); nmax = round(tmax/dt);
   udata = u; tdata = 0;
   for n = 1:nmax
      t = n*dt; g = -.5i*dt*k;
      E = exp(dt*ik3/2); E2 = E.^2;
      a = g.*fft(real( ifft( v ) ).^2);
      b = g.*fft(real( ifft(E.*(v+a/2)) ).^2); % 4th-order
      c = g.*fft(real( ifft(E.*v + b/2) ).^2); % Runge-Kutta
      d = g.*fft(real( ifft(E2.*v+E.*c) ).^2);
      v = E2.*v + (E2.*a + 2*E.*(b+c) + d)/6;
      if mod(n,nplt) == 0 u = real(ifft(v));
         udata = [udata u]; tdata = [tdata t];
      end
   end
end
```

1.3 Numerical Results

We take $\kappa = 0.3$, $y_0 = -20$ and $L = 15$ in the intial solution (1), which corresponds to

$$v(x,t) = 12\,\kappa^2 \mathrm{sech}^2(\kappa L(x - \pi - y_0/L) - 4\kappa^3 t),$$

and use tmax = 60, dt = 0.01 for various N to solve the KdV equation and verify the accuracy of the numerical solution.

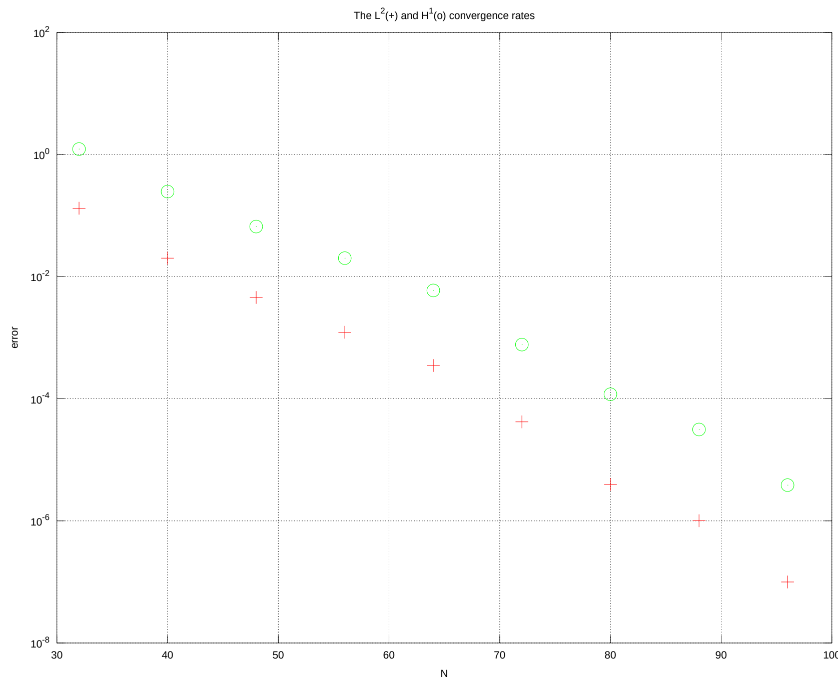

Figure 1. The convergence rate of Fourier method for KdV equation without anti-aliasing.

2 Kuramoto–Sivashinsky (KS) Equation

$$\begin{array}{l} u_t + u_{xxxx} + u_{xx} + u\,u_x = 0, \quad x \in (-\infty, \infty), \quad t > 0, \\ u(x,t) = u(x + 2\pi L, t), \quad u_x(x,t) = u_x(x + 2\pi L, t), \quad t \geqslant 0, \\ u(x,0) = u_0(x), \quad x \in (-\infty, \infty). \end{array}$$

1. Let

$$u \approx \sum_{k=-N/2}^{N/2} \tilde{u}_k(t)\, e^{ikx/L}, \quad t > 0$$

plugging into KS equation, and pairting the result with $e^{ikx/L}$, we get

$$\tilde{u}_k'(t) + \left(\frac{k^4}{L^4} - \frac{k^2}{L^2}\right)\tilde{u}_k(t) = -\frac{1}{2L} i\,k\,\tilde{w}_k(t), \quad t > 0, \tag{7}$$

where

$$\tilde{w}_k = \frac{1}{2\pi L}\int_{-\pi L}^{\pi L} u^2(x,t)e^{-ikx/L}\,\mathrm{d}x \approx \frac{1}{N}\sum_{j=0}^{N-1} u^2(x_j,t)e^{-ikx_j/L} = (I_N(u^2), e^{ikx/L}).$$

2. Using RK4 to solve the ODE system (7) with integrating factor $g = e^{\lambda t}$, $\lambda = k^4/L^4 - k^2/L^2$.

$$(e^{\lambda t}\tilde{u}_k)' = -\frac{i\,k}{2L}\,e^{\lambda t}\tilde{w}_k,$$

or

$$\left(e^{\lambda(t-t^n)}\tilde{u}_k\right)' = -\frac{i\,k}{2L}e^{\lambda(t-t^n)}\tilde{w}_k$$

```
% p32.m - Solve Kuramoto-Sivashinsky(KS) eq.
%            u_t + uu_x + u_xx + u_xxxx = 0 on [-pi L,pi L]
%          by FFT with integrating factor.
clf; clear;
function [x tdata udata] = KSsolu(uex, L, N, tmax, dt, nplot)
  x = (2*pi*L/N)*(0:N-1)';
  u = feval(uex, x); v = fft(u);
  k = [0:N/2-1 0 -N/2+1:-1]'/L;
  nplt = floor((tmax/nplot)/dt); nmax = round(tmax/dt);
  udata = u; tdata = 0;
  for n = 1:nmax
    t = n*dt; g = -.5i*dt*k;
    E = exp(-dt*(k.^4-k.^2)/2); E2 = E.^2;
    a = g.*fft(real( ifft( v ) ).^2);
    b = g.*fft(real( ifft(E.*(v+a/2)) ).^2); % 4th-order
    c = g.*fft(real( ifft(E.*v + b/2) ).^2); % Runge-Kutta
    d = g.*fft(real( ifft(E2.*v+E.*c) ).^2);
    v = E2.*v + (E2.*a + 2*E.*(b+c) + d)/6;
    if mod(n, nplt) == 0
       u = real(ifft(v));
       udata = [udata u]; tdata = [tdata t];
    end
  end
end
L=16; u =inline('cos(x/L).*(1+sin(x/L))', 'x');
tmax=300; dt=0.05; nplot=300; N=128;
[x, tdata, udata] = KSsolu(u, L, N, tmax, dt, nplot);
[tt, xx]=meshgrid(tdata,x); imagesc(tt, xx, udata);
xlabel t, ylabel x
```

2.1 Numerical Results

We take $L = 16$ and impose the initial condition:

$$u_0(x) = \cos(x/L)(1 + \sin(x/L)). \tag{8}$$

We also set $T_{\max} = 300$, time step $\delta t = 0.05$, $N = 128$. Following figure give the numerical result.

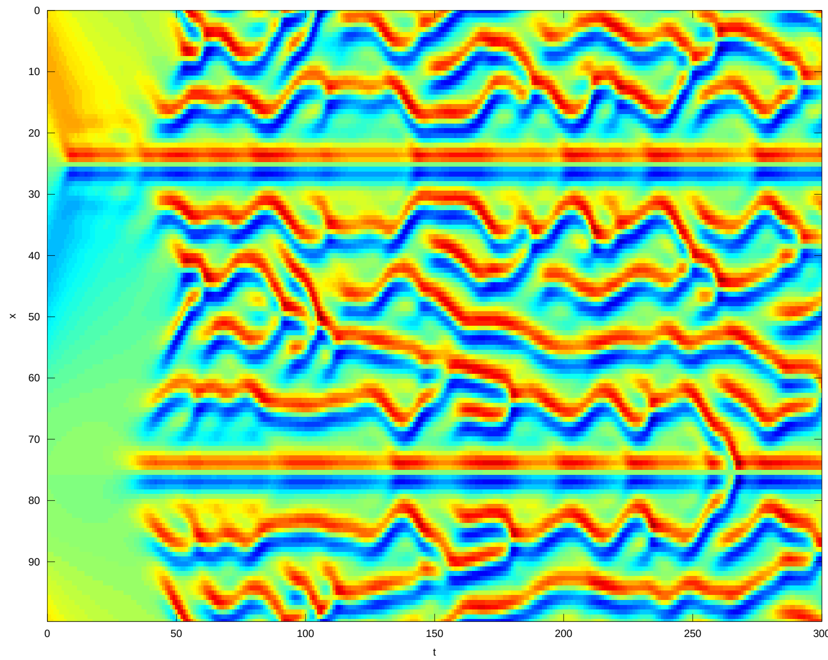

Figure 2. The numerical solution of the KS equation with initial profile (8), and $L = 16$ for $t \in [0, 300]$. The discretization parameters are: $N = 128$, $\delta t = 0.005$. This figure is generated by p32.m.

3 Allen–Cahn Equation

We consider the two-dimensional Allen–Cahn equation with periodic boundary conditions:

$$\begin{aligned} &\partial_t u - \varepsilon^2 \Delta u + u^3 - u = 0, \quad (x, y) \in \Omega = (-1, 1)^2, \quad t > 0, \\ &u(-1, y, t) = u(1, y, t), \quad u(x, -1, t) = u(x, 1, t), \quad t > 0, \\ &u(x, y, 0) = u_0(x, y), \quad (x, y) \in \Omega. \end{aligned} \tag{9}$$

3.1 Strang splitting

To get a absolutely stable scheme, we use Strang splitting. Denote by $u^n = u(\cdot, \cdot, t^n)$, $t^n = n\,h$ with h is the time step, then each time step of the first order Strang splitting scheme for Allen-Cahn equation consists three substeps:

$$\frac{\partial}{\partial t} u_1 - \varepsilon^2 \Delta u_1 = 0, \quad u_1(t = 0) = u^n \quad \Longrightarrow \quad u_1^* = u_1(h/2); \tag{10}$$

$$\frac{\partial}{\partial t} u_2 + u_2^3 - u_2 = 0, \quad u_2(t = 0) = u_1^*; \quad \Longrightarrow \quad u_2^* = u_2(h); \tag{11}$$

$$\frac{\partial}{\partial t} u_3 - \varepsilon^2 \Delta u_3 = 0, \quad u_3(t = 0) = u_2^*; \quad \Longrightarrow \quad u^{n+1} = u_3(h/2). \tag{12}$$

The first and third substeps involves solving a diffusion equation, which can be down by FFT in $O(N^2 \log_2 N)$ operations. The second step is a ODE equation for the grid values of u. which is equivalent to solve

$$u' - u = -u^3 \quad \Longrightarrow \quad u^{-3} u' - u^{-2} = -1 \quad \Longrightarrow \quad -\frac{1}{2}(u^{-2})' - u^{-2} = -1$$

By variable change $v = u^{-2}$, we get

$$v' + 2v = 2 \quad \Longrightarrow \quad v(t) = v_0\, e^{-2t} + (1 - e^{-2t})$$

so we get

$$u(t) = \pm 1 / \sqrt{u_0^{-2} e^{-2t} + (1 - e^{-2t})} = \frac{u_0}{\sqrt{e^{-2t} + (1 - e^{-2t})\, u_0^2}}.$$

We use $u_N = \sum_{k=-N/2}^{N/2} \sum_{j=-N/2}^{N/2} \tilde{u}_{k,j} e^{i k \pi x} e^{i j \pi y}$ to approximate the solution u in (9). By applying this to the first substeps (10) of Strang splitting, we get

$$\tilde{u}_{k,j}'(t) + \varepsilon^2 \pi^2 (k^2 + j^2) \tilde{u}_{k,j} = 0.$$

This also applys to the third substep. So the overall algorithm reads

1. Set up the initial values on discrete grid points: $\{\, u_{k,j}^0 = u(x_k, y_j, 0), \quad k, j = 0, \ldots, N-1 \,\}$, and transform it to spectral coefficients $\{\, \tilde{u}_{k,j}^0$: $k, j = 0, \pm 1, \ldots, \pm N/2 \,\}$by 2-dimensional FFT.

2. Let $t^n = n\, h$, for $n = 1, \ldots, n_{\max}$, do the Strange splitting

   a. calculate

   $$\tilde{u}_{k,j} = \tilde{u}_{k,j}^n\, e^{-\lambda h}, \quad \lambda = \varepsilon^2 \pi^2 (k^2 + j^2).$$

   b. Solving equation (11) in three steps.

      i. Transform $\{\tilde{u}_{k,j}\}$ to physical values $\{u_{k,j}\}$ by 2-dimensional reverse FFT, in matlab this is

      ```
      u = real( ifft2( utilde ) ),
      ```

      where `utilde` is the matrix formed by $\{\, \tilde{u}_{k,j} \,\}$, and `u` is the matrix formed by $\{\, u_{k,j} \}$;

      ii. Solving equation (11). This can be down by

      $$u_{k,j} \quad \longleftarrow \quad \frac{u_{k,j}}{\sqrt{e^{-2h} + (1 - e^{-2h}) u_{k,j}^2}};$$

      iii. Transform $\{u_{k,j}\}$ to spectral coefficients $\{\, \tilde{u}_{k,j} \,\}$ by 2-dimensional FFT in matlab: utilde=fft2(u);

   c. calculate

   $$\tilde{u}_{k,j}^{n+1} = \tilde{u}_{k,j}\, e^{-\lambda h}, \quad \lambda = \varepsilon^2 \pi^2 (k^2 + j^2).$$

3. Output the numerical solution and do other post-processing.

3.2 Numerical results

We take initial condition

$$u_0(x, y) = \begin{cases} 1, & \text{if } x^2 + y^2 \leqslant 1/4, \\ -1, & \text{otherwise.} \end{cases}$$

The snapshot of the solutions at several time steps are given in following figure. The inner $u = 1$ region shrinks, and eventually disappears.

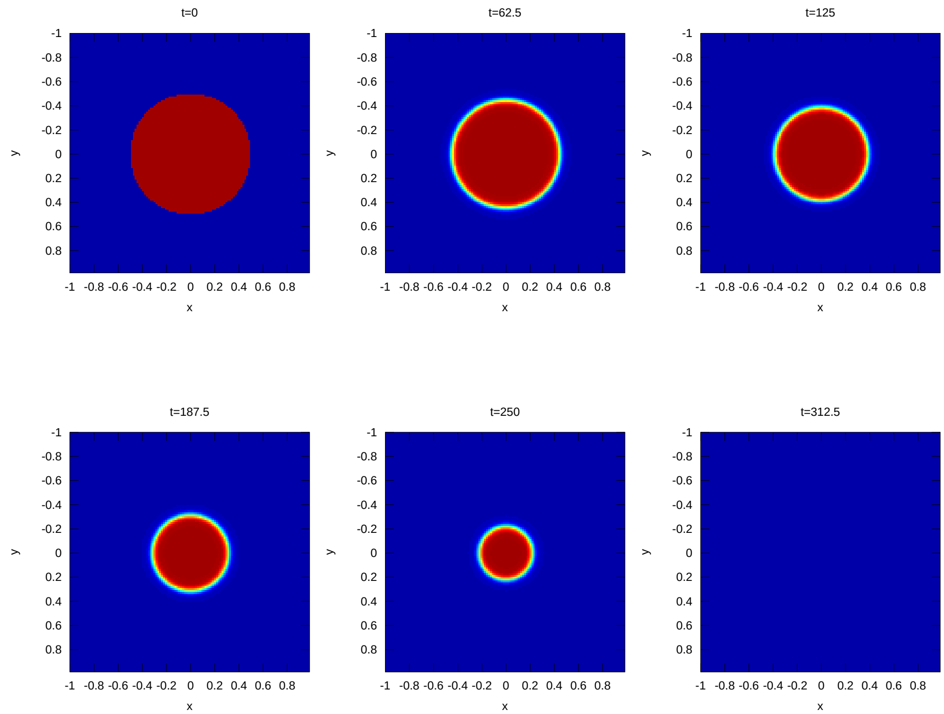

Figure 3. The solution of the Allen–Cahn equation. The solver parameters are $\varepsilon = 0.02$, $\delta t = 0.05$, $N = 128$.