MA2507 Computing Mathematics Laboratory

1. Lorenz equation: In a standard ODE course, you learn analytic methods for solving ordinary differential equations, including first order ODEs, second and higher order ODEs, and system of ODEs. Notice the major difference between linear and nonlinear ODEs. Most nonlinear ODEs cannot be solved analytically. Most linear ODEs have "simple" behavior. If the ODE is for a function $y(t)$, then as $t \to \infty$, $y(t)$ may converge to a constant, may diverge, or may approach a periodic solution. Can ODEs have more exotic behavior? Yes, it was first observed by Edward Lorenz in 1963 in the following system of ODEs:

$$\begin{aligned} x' &= -\sigma(x-y) \\ y' &= -xz + Rx - y \\ z' &= xy - bz, \end{aligned}$$

where σ, R and b are constants. For $R = 28$, $\sigma = 10$ and $b = 8/3$, Lorenz found chaos (the mathematical term chaos was only introduced in the 70's). The solution does not converge to a constant or a periodic solution. Two solutions that are very close at $t = 0$ separate rapidly (actually exponentially) as t is increased. This is called "sensitive dependence on initial conditions", also called "butterfly effect". Lorenz is a meteorologist, he realized that if the differential equations for weather also have "sensitive dependence on initial conditions", then long-term weather forecast will never be possible. Indeed, this is now widely believed. A mathematical proof of chaos in Lorenz equation was only completed around year 2000.

We can solve the Lorenz equation using MATLAB internal program `ode45`. To do that, we must introduce a column vector $\mathbf{y} = [x, y, z]^T$, since the ODE solver only works for first order systems written as

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}) \tag{1}$$

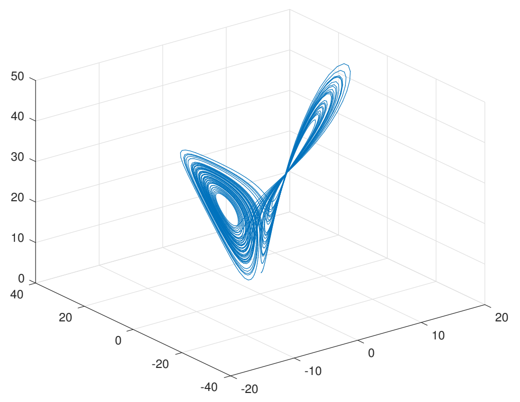
and we need a function for the right hand side of Eq. (1) as a column vector. Below is a MATLAB program.

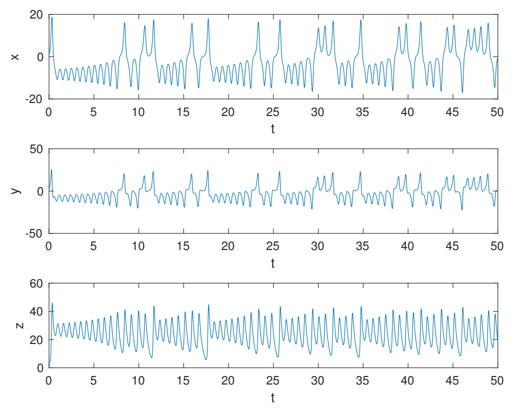
```
% main script
tspan = [0,50];                  % from t=0 to t=50
yzero = [1, 2.3, 4.1];           % initial conditions
[tt,yy] = ode45(@lorenz,tspan,yzero);
plot3(yy(:,1),yy(:,2),yy(:,3))  % 3D curves
grid

% a function for the right hand side of the ODE system, as a column vector
function k = lorenz(t,y)
s = 10; b = 8/3; r = 28;
k = [s*(y(2)-y(1)); y(1)*(r-y(3))-y(2); y(1)*y(2)-b*y(3)];
end
```

Even though t does not appear in the Lorenz equation, we must define the function as a function of t and $\mathbf{y}$. We use `tspan` to specify the starting and ending time for the solution. `yzero` is a vector of the initial conditions, that is, $x(0)$, $y(0)$ and $z(0)$. The program above gives the figure (3D curve) on next page. The solutions are given in a column vector `tt` and 3-column matrix `yy`. The vector `tt` discretizes t from $t = 0$ to $t = 50$, the columns of `yy` give the corresponding x, y and z. If we replace the last two lines by

```
subplot(3,1,1), plot(tt, yy(:,1)), xlabel('t'), ylabel('x')
subplot(3,1,2), plot(tt, yy(:,2)), xlabel('t'), ylabel('y')
subplot(3,1,3), plot(tt, yy(:,3)), xlabel('t'), ylabel('z')
```

then, we get a figure showing x, y and z as functions of t. You can also use the commands `comet`





and `comet3` to see animated plots.

2. KdV equation: The Korteweg-de Vries (KdV) equation (1895)

$$u_t + uu_x + \delta^2 u_{xxx} = 0,$$

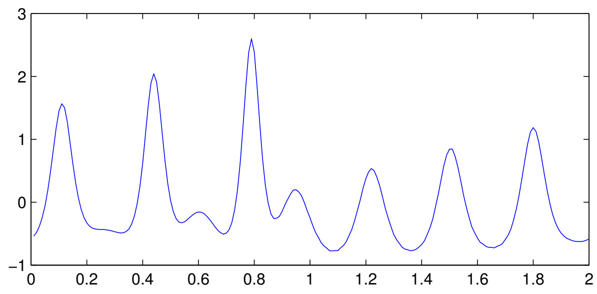
where δ is a constant, models waves in shallow water. The KdV equation has soliton solutions which were first observed by J. Scott Russell on the Edinburgh-Glasgow canal in 1834. Many mathematical properties of the KdV equation were found after the initial numerical study of Zabusky and Kruskal in 1965. They solved the KdV equation for $\delta = 0.022$ and $0 \le x \le 2$ assuming periodic boundary condition $u(x+2,t) = u(x,t)$ for all x and all t, and an initial condition $u(x,0) = \cos(\pi x)$. We will repeat this calculation with our own method. Discretizing x by $x_j = 2j/n$ for $n = 200$ for $1 \le j \le n$, the KdV equation is approximated by the following system of ODEs:

$$\frac{du_j}{dt} = -u_j \frac{u_{j+1} - u_{j-1}}{2d} - \frac{\delta^2}{2d^3}(u_{j+2} - 2u_{j+1} + 2u_{j-1} - u_{j-2}),$$

for $1 \le j \le n$, where $d = 2/n$ and $u_j(t) = u(x_j,t)$. Due to the periodic boundary condition, we need to set $u_0 = u_n$, $u_{-1} = u_{n-1}$, $u_{n+1} = u_1$ and $u_{n+2} = u_2$ in the equations for $j = 1$, 2, $n-1$ and n. The MATLAB program is given below.

```
% main script
n=200;
x =(2/n)*(1:n);
u= cos(pi*x);           % initial condition
tspan = [0, 1.5];       % time interval
[tt,uu] = ode45(@kdv, tspan, u);
m = length(tt);
for j = 1:10:m
    plot(x,uu(j,:))    % show solution every 10 time steps
    axis([0 2 -1 3])
    pause(0.01);      % show the plot only for 0.01 seconds
end
plot(x,uu(m,:))

% right hand side of the ODE system approximating thge KdV equation
% u, f  -- column vectors
function f = kdv(t,u)
  n = length(u);
  u = [u; u(1:4)];  %  extend the vector u by 4 elements periodically
  f = zeros(n+2,1); %  define a longer f
  delta = 0.022;
  d = 2/n;
  c = delta^2/(2*d^3);
  for j=3:n+2
      f(j)=-u(j)*(u(j+1)-u(j-1))/(2*d)-c*(u(j+2)-2*u(j+1)+2*u(j-1)-u(j-2));
  end
  f(1:2)=f(n+1:n+2);  % get f(1) and f(2)
  f = f(1:n);         % reset f as a vector of length n
end
```

The j loop above gives the time evolution of the solution. The final line plots the solution at $t = 1.5$.



3. Water droplet: The cross-sectional shape of a water droplet on a flat surface is given by the following ODE

$$u'' + (1-u)\left[1+(u')^2\right]^{3/2} = 0, \quad -1 < x < 1,$$

with boundary conditions

$$u(-1) = u(1) = 0,$$

where $u = u(x)$ is a function of x, the base of the droplet is assumed to be $[-1, 1]$. This is an ODE boundary value problem (BVP). To solve it in MATLAB, we must write the second order ODE as a first order system, for $\mathbf{y} = [u, u']^T$, and must also write down the two boundary conditions in a column. The first order ODE system is

$$\frac{d\mathbf{y}}{dx} = \mathbf{f}(x, \mathbf{y})$$

and we need a function for $\mathbf{f}$. The boundary conditions are

$$\mathbf{r} = \mathbf{r}(\mathbf{y}_a, \mathbf{y}_b) = \mathbf{0}$$

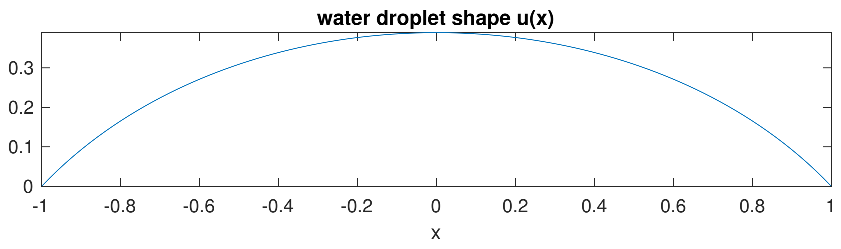
where

$$\mathbf{y}_a = \begin{bmatrix} u(-1) \\ u'(-1) \end{bmatrix}, \quad \mathbf{y}_b = \begin{bmatrix} u(1) \\ u'(1) \end{bmatrix},$$

and we need a function for $\mathbf{r}$. This is a nonlinear ODE, we also need an initial guess. The initial guess is for $\mathbf{y}$. Here, we will use $0.5\cos(\pi x/2)$ as the initial guess for $u(x)$, then the initial guess for $u'(x)$ is $-0.25\pi\sin(\pi x/2)$. We use MATLAB program `bvp4c` to solve the ODE BPV. It requires `bvpinit` to set up the initial data. Here is the propgram.

```
% main script
x = linspace(-1,1,100);     % discretize x
s0 = bvpinit(x,@myigs);     % s0 is a structure for initial data
s = bvp4c(@myde,@mybc,s0); % s is a structure for the final solution
plot(s.x, s.y(1,:))
axis equal tight
xlabel('x'), title('water droplet shape u(x)')

% function for initial guess
function y0 = myigs(x)
  y0 = [0.5*cos(pi*x/2); -0.25*pi*sin(pi*x/2)];
end

% function for the ODE written as a 1st order system
function f = myde(x,y)
  f = [y(2); (y(1)-1)*(1 + (y(2))^2)^(3/2)];
end

% function for boundary conditions, writte as a column vector
function r = mybc(ya,yb)
  r = [ya(1); yb(1)];
end
```

The program gives the following figure.


water droplet shape u(x)
x