Solving Initial Value Problems

Jake Blanchard
University of Wisconsin - Madison
Spring 2008

Example Problem

- Consider an 80 kg paratrooper falling from 600 meters.
- The trooper is accelerated by gravity, but decelerated by drag on the parachute
- This problem is from Cleve Moler's book called Numerical Computing with Matlab (my favorite Matlab book)

Governing Equation

- m=paratrooper mass (kg)
- g=acceleration of gravity (m/s^2)
- V=trooper velocity (m/s)
- Initial velocity is assumed to be zero

$$\boxed{m\frac{dV}{dt} = -mg - \frac{4}{15}V * |V|}$$

Solving ODE's Numerically

- Euler's method is the simplest approach
- Consider most general first order ODE: dy/dt=f(t,y)
- Approximate derivative as $(y_{i+1}-y_i)/dt$
- Then:

$$\frac{dy}{dt} \approx \frac{y_{i+1} - y_i}{\Delta t} = f(t, y_i)$$

$$y_{i+1} \approx y_i + \Delta t \ f(t, y_i)$$

A Problem

- Unfortunately, Euler's method is too good to be true
- It is unstable, regardless of the time step chosen
- We must choose a better approach
- The most common is 4th order Runge-Kutta

Runge-Kutta Techniques

- Runge-Kutta uses a similar, but more complicated stepping algorithm

$$k_1 = \Delta t * f(t, y_i)$$

$$k_2 = \Delta t * f(t + \frac{\Delta t}{2}, y_i + \frac{k_1}{2})$$

$$k_3 = \Delta t * f(t + \frac{\Delta t}{2}, y_i + \frac{k_2}{2})$$

$$k_4 = \Delta t * f(t + \Delta t, y_i + k_3)$$

$$y_{i+1} = y_i + \frac{k_1 + 2(k_2 + k_3) + k_4}{6}$$

Approach

- Choose a time step
- Set the initial condition
- Run a series of steps
- Adjust time step
- Continue

Preparing to Solve Numerically

- First, we put the equation in the form

$$\boxed{\frac{dy}{dt} = f(t, y)}$$

- For our example, the equation becomes:

$$\boxed{\frac{dV}{dt} = -g - \frac{4}{15}\frac{V * |V|}{m}}$$

Solving Numerically

- There are a variety of ODE solvers in Matlab
- We will use the most common: **ode45**
- We must provide:
  - a function that defines the function derived on previous slide
  - Initial value for V
  - Time range over which solution should be sought

How ode45 works

- ode45 takes two steps, one with a different error order than the other
- Then it compares results
- If they are different, time step is reduced and process is repeated
- Otherwise, time step is increased

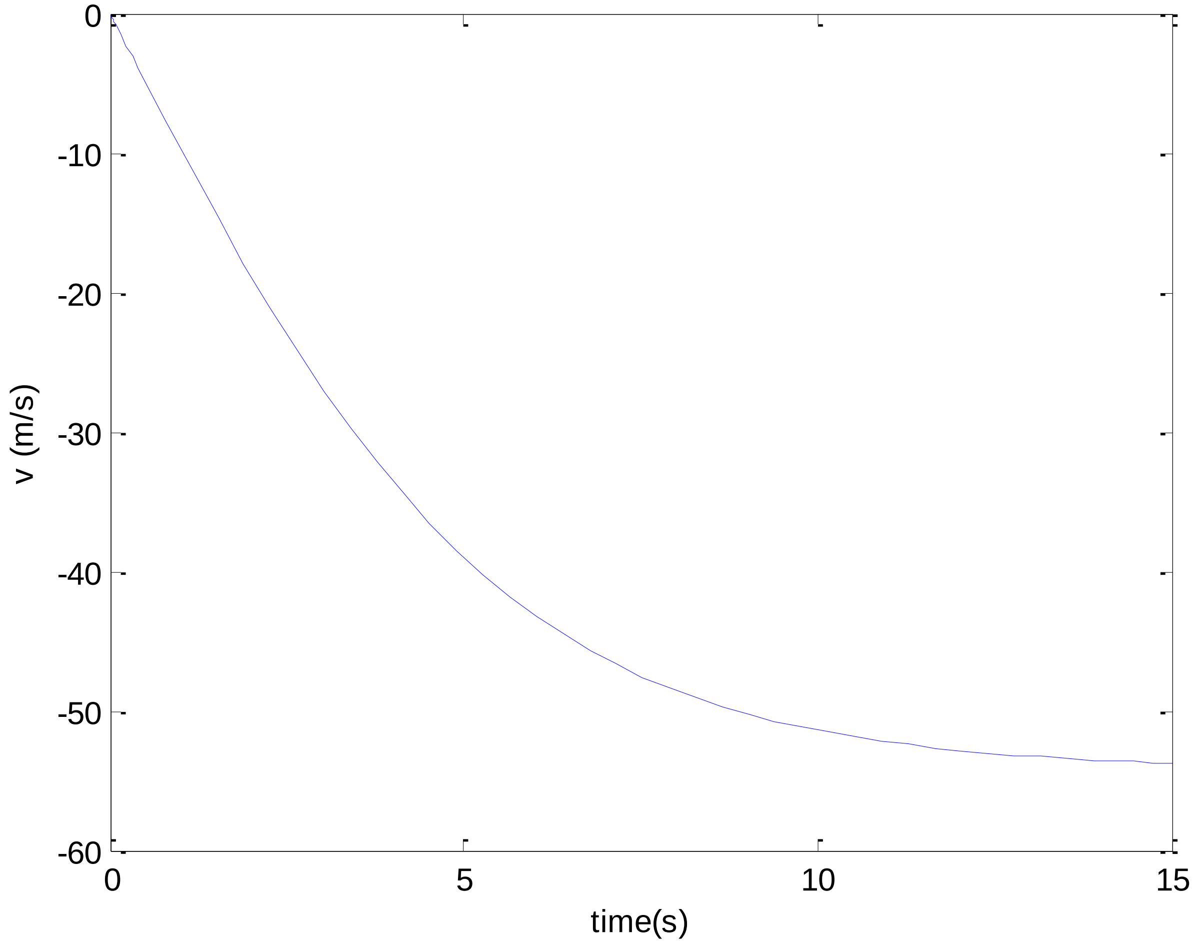
The Solution

```
clear all
timerange=[0 15]; %seconds
initialvelocity=0; %meters/second
[t,y]=ode45(@f,timerange, initialvelocity)
plot(t,y)
ylabel('velocity (m/s)')
xlabel('time(s)')
```

The Function

```
function rk=f(t,y)
mass=80;
g=9.81;
rk=-g-4/15*y.*abs(y)/mass;
```

My Solution



Practice

- Download the file **odeexample.m**
- Run it to reproduce my result
- Run again out to t=30 seconds
- Run again for an initial velocity of 10 meters/second
- Change to k=0 and run again (gravity only)

Practice

- The outbreak of an insect population can be modeled with the equation below.
- R=growth rate
- C=carrying capacity
- N=# of insects
- N_c=critical population
- Second term is due to bird predation

$$\boxed{\frac{dN}{dt} = RN\left(1-\frac{N}{C}\right) - \frac{rN^2}{N_c^2 + N^2}}$$

Parameters

- $0<t<50$ days
- $R=0.55$ /day
- $N(0)=10{,}000$
- $C=10{,}000$
- $N_c=10{,}000$
- $r=10{,}000$ /day

- What is steady state population?
- How long does it take to get there?

$$\frac{dN}{dt} = RN\left(1-\frac{N}{C}\right)-\frac{rN^2}{N_c^2+N^2}$$

- Note: this is a first order ode
- Skeleton script is in file: insects.m

Insects.m

```
function insects
clear all
tr=[0 ??];
initv=??;
[t,y]=ode45(@f, tr, initv);
plot(t,y)
ylabel('Number of Insects')
xlabel('time')
%
function rk=f(t,y)
rk= ??;
```

Practice

- Let h be the depth of water in a spherical tank
- If we open a drain at the tank bottom, the pressure at the bottom will decrease as the tank empties, so the drain rate decreases with h
- Find the time to empty the tank

Parameters

- R=5 ft; Initial height=9 ft
- 1 inch hole for drain

$$\boxed{\frac{dh}{dt} = -\frac{0.0334\sqrt{h}}{10h - h^2}}$$

- How long does it take to drain the tank?

Rockets

- A rocket's mass decreases as it burns fuel
- Find the final velocity of a rocket if:
- T=48000 N; m_0=2200 kg
- R=0.8; g=9.81 m/s^2; b=40 s

$$\boxed{\begin{aligned} m\frac{dv}{dt} &= T - mg \\ m &= m_0\left(1 - \frac{rt}{b}\right) \end{aligned}}$$

Options

- Options are available to:
  - Change relative or absolute error tolerances
  - Maximum number of steps
  - Etc.

Some Other Matlab routines

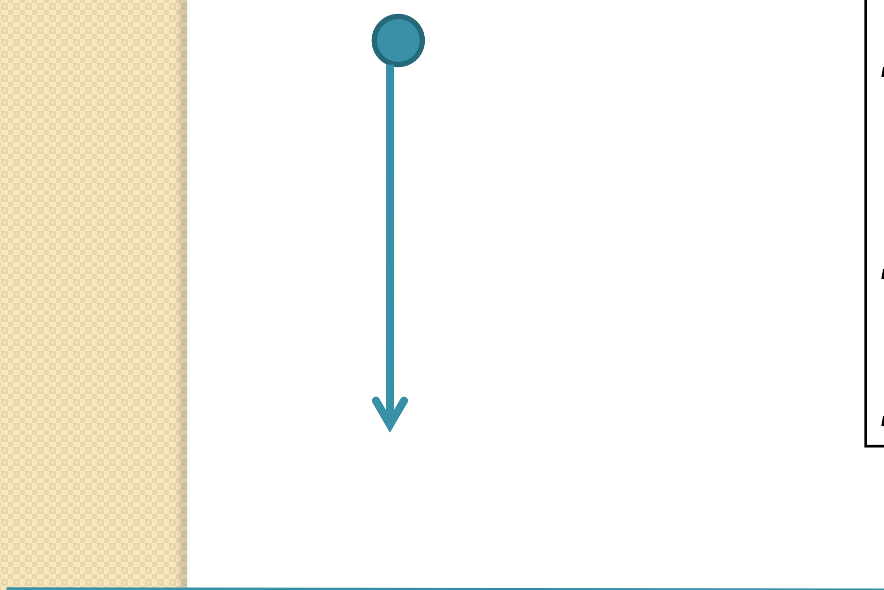
- ode23 – like ode45, but lower order
- ode15s – stiff solver
- ode23s – higher order stiff solver

Advanced IVPs

- Second order equations
- Stiff equations

Second Order Equations

- Consider a falling object with drag



$$\ddot{y} = -g - \frac{4}{15m}\dot{y}\left|\dot{y}\right|$$

$$y(0) = h$$

$$\dot{y}(0) = 0$$

Preparing for Solution

- We must break second order equation into set of first order equations
- We do this by introducing new variable (z=dy/dt)

$$z = \dot{y}$$
$$\dot{z} = \ddot{y}$$
$$\dot{z} = -\frac{4}{15m} z\left|z\right| - g$$

$$\dot{z} = -\frac{4}{15m} z\left|z\right| - g$$
$$\dot{y} = z$$
$$y(0) = h; \quad z(0) = 0$$

Solving

- Now we have to send a set of equations and a set of initial values to the ode45 routine
- We do this via vectors
- Let w be vector of solutions: w(1)=y and w(2)=z
- Let r be vector of equations: r(1)=dy/dt and r(2)=dz/dt

Function to Define Equation

$$\frac{dy}{dt} = z = w(2)$$

$$\frac{dz}{dt} = -\frac{4}{15m} w(2) * \left| w(2) \right| - g$$

```
function r=rkfalling(t,w)
...
r=zeros(2,1);
r(1)=w(2);
r(2)= -k*w(2).*abs(w(2))-g;
```

The Routines

```
tr=[0 15]; %seconds
initv=[600 0]; %start 600 m high
[t,y]=ode45(@rkfalling, tr, initv)
plot(t,y(:,1))
ylabel('x (m)')
xlabel('time(s)')
figure
plot(t,y(:,2))
ylabel('velocity (m/s)')
xlabel('time(s)')
```

Function

```
function r=rkfalling(t,w)
mass=80;
k=4/15/mass;
g=9.81;
r=zeros(2,1);
r(1)=w(2);
r(2)= -k*w(2).*abs(w(2))-g;
```

General Second Order Equations

- We can write a general second order equation as shown:
- To solve:
  - Define f
  - Set initial conditions
  - Set time range

$$\frac{d^2y}{dt^2} = f(t, y, \frac{dy}{dt})$$

$$or$$

$$\frac{dy}{dt} = z$$

$$\frac{dz}{dt} = f(t, y, z)$$

The Routines

```
tr=[0 15]; %seconds
initv=[600 0]; %start 600 m high
[t,y]=ode45(@rkfalling, tr, initv)
plot(t,y(:,1))
ylabel('x (m)')
xlabel('time(s)')
figure
plot(t,y(:,2))
ylabel('velocity (m/s)')
xlabel('time(s)')
```

```
function r=rkfalling(t,w)
mass=80;
k=4/15/mass;
g=9.81;
r=zeros(2,1);
r(1)=w(2);
r(2)= -k*w(2).*abs(w(2))-g;
```

Practice

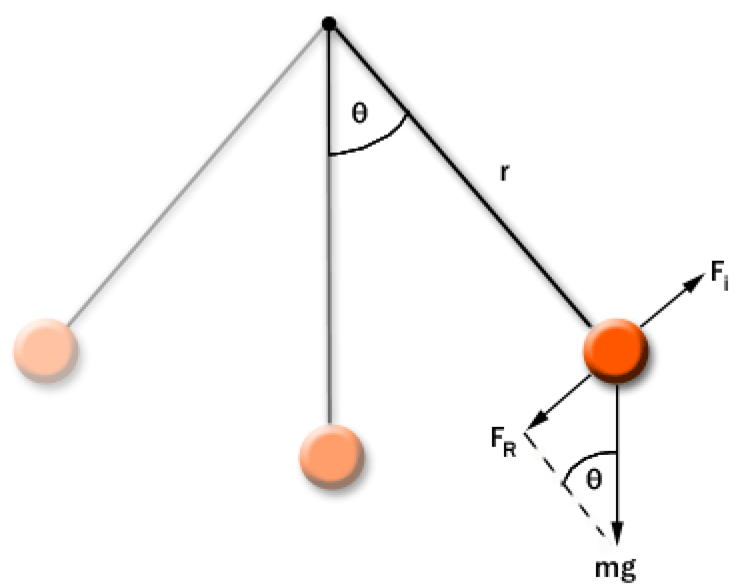
- Return to paratrooper problem.
- Download **ode2ndOrder.m**
- Run to duplicate earlier results for velocity
- Change initial velocity to 10 m/s and run again

$$\boxed{m\frac{d^2y}{dt^2} = -mg - \frac{4}{15}\frac{dy}{dt}\left|\frac{dy}{dt}\right|}$$

Practice-nonlinear pendulum

- $r=1$ m; $g=9.81$ m/s^2
- Initial angle $=\pi/8$, $\pi/2$, $\pi-0.1$

$$\boxed{\frac{d^2\theta}{dt^2} = -\frac{g}{r}\sin(\theta)}$$



Systems

- For systems of first order ODEs, just define both equations.

Practice

- Consider an ecosystem of rabbits r and foxes f. Rabbits are fox food.
- Start with 300 rabbits and 150 foxes
- α=0.01

$$\frac{dr}{dt} = 2r - \alpha rf$$

$$\frac{df}{dt} = -f + \alpha rf$$

- r=w(1)
- f=w(2)

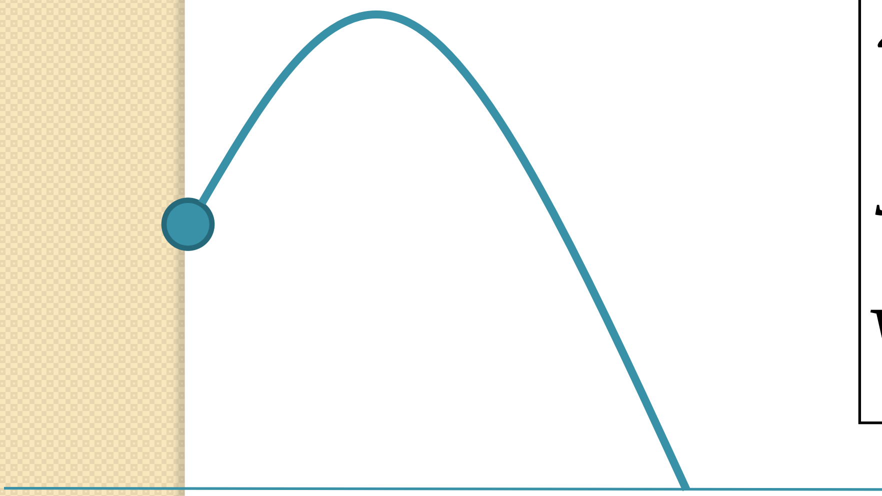
```
function z=rkfox(t,w)
alpha=0.01;
r=zeros(2,1);
z(1)=2*w(1)-alpha*w(1)*w(2);
z(2)= -w(2)+alpha*w(1)*w(2);
```

Approach

- Start with **ode2ndOrder.m**
- Modify with function from previous slide
- Put in time range ($0<t<15$) and initial conditions

Higher Order Equations

- Suppose we want to model a projectile



$$\ddot{x} = -k\,\dot{x}V$$

$$\ddot{y} = -k\,\dot{y}V - g$$

$$V = \sqrt{\dot{x}^2 + \dot{y}^2}$$

Now we need 4 1st order ODEs

$$\boxed{\begin{aligned} \dot{x} &= s \\ \dot{s} &= -k\, sV \\ \dot{y} &= z \\ \dot{z} &= -k\, zV - g \\ V &= \sqrt{s^2 + z^2} \end{aligned}}$$

The Code

```
clear all;
tspan=[0 1.1]
wnot(1)=0; wnot(2)=10;
wnot(3)=0; wnot(4)=10;
[t,y]=ode45('rkprojectile',tspan,wnot);
plot(t,y(:,1),t,y(:,3))
figure
plot(y(:,1),y(:,3))
```

The Function

```
function r=rkprojectile(t,w)
g=9.81;
x=w(1); s=w(2); y=w(3); z=w(4);
vel=sqrt(s.^2+z.^2);
r=zeros(4,1);
r(1)=s;
r(2)=-s*vel;
r(3)=z;
r(4)=-z*vel-g;
```

Questions