Chapter 8

Ordinary Differential Equations

Soon-Hyung Yook

May 16, 2017

Table of Contents I

First-Order Differential Equations with One Variable I

- The simplest type of ordinary differential equation (ODE)
- Example:

$$\frac{dx}{dt} = \frac{2x}{t}. \tag{1}$$

- Eq. (1) can be solved analytically and exactly by separating the variable.
- Another example:

$$\frac{dx}{dt} = \frac{2x}{t} + \frac{3x^2}{t^3}. \tag{2}$$

- Eq. (2) is no longer separable.
- Moreover, Eq. (2) is nonlinear.
  - Nonlinear equations can rarely be solved analytically.
  - But they can be solved **numerically**.

First-Order Differential Equations with One Variable II

- Computer don't care whether a differential equation is linear or nonlinear–the techniques used for both cases are the same.

General Form of a First-Order One-Variable ODE

$$\frac{dx}{dt} = f(x,t), \tag{3}$$

where $f(x,t)$ is a function we specify.

- Examples of $f(x,t)$:
  - in Eq. (1): $f(x,t) = \frac{2x}{t}$
  - in Eq. (2): $f(x,t) = \frac{2x}{t} + \frac{3x^2}{t^3}$
- The only dependent variable in Eq. (3) is t.

First-Order Differential Equations with One Variable III

Another form of a first-order one-variable ODE

$$\frac{dy}{dx} = f(y, x), \tag{4}$$

where $f(x, y)$ is a function we specify.

- To solve Eq. (3) or Eq. (4), we need an initial condition or boundary condition.

Euler Method

The most easiest and intuitive method.
Solve the equation:

$$\frac{dx}{dt} = f(x,t) \tag{5}$$

From the Taylor expansion of y around x

$$x(t+h) = x(t) + hx' + \frac{1}{2}h^2x'' + \cdots \tag{6}$$

Since $x' = f(x,t)$,

$$x(t+h) = x(t) + hf(x,t) + \mathcal{O}(h^2)$$

or

Euler Method (EM)

$$x_{i+1} = x_i + hf_i + \mathcal{O}(h^2) \tag{7}$$

where $f_i \equiv f(x_i, t_i)$, $x_i \equiv x(t_i)$, and $x_{i+1} \equiv x(t_i + h)$.

Euler's Method: Example I

Solve the differential equation using EM

$$\frac{dx}{dt} = -x^3 + \sin t \quad (8)$$

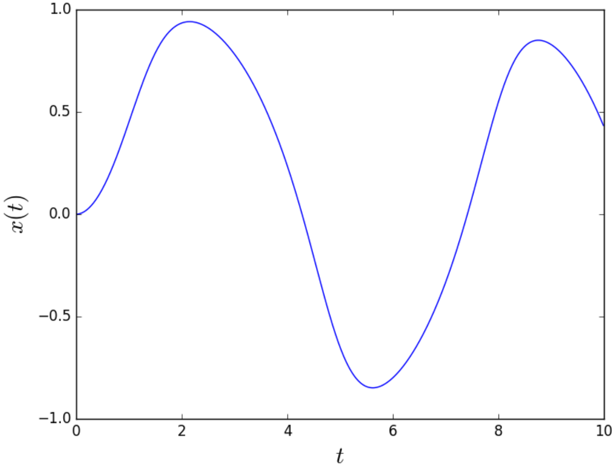
with initial condition $x = 0$ at $t = 0$. $t \in [0, 10]$.

```
from math import sin
from numpy import arange
from pylab import plot,xlabel,ylabel,show

def f(x,t):
  y=-x**3 + sin(t)
  return y

#initial conditions
t_i=0.0
x_i=0.0

t_f=10.0      # End of the interval to calculate
```

Euler's Method: Example II

```
N=1000    #number of steps
h=(t_f-t_i)/N
pt=[]
px=[]

pt.append(t_i)
px.append(x_i)
t_list=arange(t_i,t_f,h)
x=x_i
for t in t_list:
   x+=h*f(x,t)
   px.append(x)
   pt.append(t+h)

plot(pt,px)
xlabel(r"$t$",fontsize=20)
ylabel(r"$x(t)$",fontsize=20)
show()
```

Euler's Method: Example III



EM: Error Estimation

- In Eq. (7), we neglect the h^2 and all higher-order terms.
  - Leading order of error for each step $\Rightarrow h^2$.

$$\begin{aligned}\sum_{i=0}^{N-1} \frac{1}{2} h^2 \left(\frac{d^2x}{dt^2}\right)_{x=x_i, t=t_i} &= \frac{1}{2} h \sum_{i=0}^{N-1} h \left(\frac{df}{dt}\right)_{x=x_i, t=t_i} \simeq \frac{1}{2} h \int_a^b \frac{df}{dt} dt \\ &= \frac{1}{2} h \left[f(x(b), b) - f(x(a), a) \right] \end{aligned} \quad (9)$$

Therefore, the estimated error for EM is $\mathcal{O}(h)$.

Homework

Solve the differential equation

$$y' + \frac{y}{\sqrt{x^2+1}} = \frac{1}{x + \sqrt{x^2+1}}$$

with initial condition $y = 0$ when $x = 0$ using EM.

Picard Method

Eq. (5) can be expressed as

$$x_{i+1} = x_i + \int_{t_i}^{t_i+h} f(x,t)dt. \tag{10}$$

Use the trapezoidal method in "Numerical Calculus" for the integral in Eq. (10):

Picard Method

$$x_{i+1} = x_i + \frac{h}{2}(f_i + f_{i+1}) + \mathcal{O}(h^3) \tag{11}$$

Predictor-Corrector Method

1. Apply a less accurate algorithm to predict the next value x_{i+1} (Predictor)

   for example, Euler method of Eq. (7)
2. Apply a better algorithm to improve the new value (Corrector)

   for example, Picard method of Eq. (11)

Predictor-Corrector Method (PCM)

1. calculate the predictor

$$x_{i+1} = x_i + hf_i \equiv p(x_{i+1})$$

2. apply the correction by Picard method

$$x_{i+1} = x_i + \frac{h}{2}\left(f_i + f_{i+1}(t_{i+1}, p(x_{i+1}))\right).$$

Predictor-Corrector Method: Example I

Solve the differential equation using PCM

$$\frac{dx}{dt} = -x^3 + \sin t \tag{12}$$

with initial condition $x = 0$ at $t = 0$. $t \in [0, 10]$.

```
from math import sin
from numpy import arange
from pylab import plot,xlabel,ylabel,show

def f(x,t):
  y=-x**3 + sin(t)
  return y

#initial conditions
t_i=0.0
x_i=0.0

t_f=10.0      # End of the interval to calculate
```

Predictor-Corrector Method: Example II

```
N=1000    #number of steps
h=(t_f-t_i)/N
pt=[]
px=[]

pt.append(t_i)
px.append(x_i)
t_list=arange(t_i,t_f,h)
x=x_i
for t in t_list:
  p=x+h*f(x,t)
  x+=h*(f(x,t)+f(p,t+h))/2.0
  px.append(x)
  pt.append(t+h)

plot(pt,px)
xlabel(r"$t$",fontsize=20)
ylabel(r"$x(t)$",fontsize=20)
show()
```

Homework

Solve the differential equation

$$y' + \frac{y}{\sqrt{x^2+1}} = \frac{1}{x+\sqrt{x^2+1}}$$

with initial condition $y = 0$ when $x = 0$ using PCM.

Runge-Kutta Method ver.1: Second Order I

- Similar to the three-point definition of derivative

Expansion $x(t+h)$ around $t + \frac{1}{2}h$:

$$x(t+h) = x(t+\frac{1}{2}h) + \frac{1}{2}h\left(\frac{dx}{dt}\right)_{t+\frac{1}{2}} + \frac{1}{8}h^2\left(\frac{d^2x}{dt^2}\right)_{t+\frac{1}{2}} + \mathcal{O}(h^3). \tag{13}$$

$$x(t) = x(t+\frac{1}{2}h) - \frac{1}{2}h\left(\frac{dx}{dt}\right)_{t+\frac{1}{2}} + \frac{1}{8}h^2\left(\frac{d^2x}{dt^2}\right)_{t+\frac{1}{2}} + \mathcal{O}(h^3). \tag{14}$$

Subtract Eq. (14) from Eq. (13), then we obtain

$$\begin{aligned} x(t+h) &= x(t) + h\left(\frac{dx}{dt}\right)_{t+\frac{1}{2}} + \mathcal{O}(h^3) \\ &= x(t) + hf\left(x\left(t+\frac{1}{2}h1\right), t+\frac{1}{2}h\right) + \mathcal{O}(h^3) \end{aligned} \tag{15}$$

Runge-Kutta Method ver.1: Second Order II

Second-Order Runge-Kutta Method (RKM) ver.1

$$k_1 = hf(x,t) \tag{16}$$

$$k_2 = hf(x + \frac{1}{2}k_1, t + \frac{1}{2}h) \tag{17}$$

$$x(t+h) = x(t) + k_2 \tag{18}$$

- Accumulated error: $\mathcal{O}(h^2)$.
- Similar to the rectangular method for numerical integration.

Runge-Kutta Method: Example I

Solve the differential equation using second-order RKM ver.1

$$\frac{dx}{dt} = -x^3 + \sin t \tag{19}$$

with initial condition $x = 0$ at $t = 0$. $t \in [0, 10]$.

```
from math import sin
from numpy import arange
from pylab import plot,xlabel,ylabel,show

def f(x,t):
  y=-x**3 + sin(t)
  return y

#initial conditions
t_i=0.0
x_i=0.0

t_f=10.0      # End of the interval to calculate

N=1000   #number of steps
h=(t_f-t_i)/N
```

Runge-Kutta Method: Example II

```
pt=[]
px=[]

pt.append(t_i)
px.append(x_i)
t_list=arange(t_i,t_f,h)
x=x_i
for t in t_list:
  k1=h*f(x,t)
  k2=h*f(x+0.5*k1,t+0.5*h)
  x+=k2
  px.append(x)
  pt.append(t+h)

plot(pt,px)
xlabel(r"$t$",fontsize=20)
ylabel(r"$x(t)$",fontsize=20)
show()
```

Homework

Solve the differential equation

$$y' + \frac{y}{\sqrt{x^2+1}} = \frac{1}{x+\sqrt{x^2+1}}$$

with initial condition $y = 0$ when $x = 0$ using second-order RM ver.1.

Runge-Kutta Method: General

To solve the differential equation:

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

we expand $y(t + \tau)$ in terms of the quantities at t with the Taylor expansion:

$$y(t + \tau) = y + \tau y' + \frac{\tau^2}{2} y'' + \frac{\tau^3}{3!} y^{(3)} + \frac{\tau^4}{4!} y^{(4)} + \cdots \quad (20)$$

Let

$$f_{yt} \equiv \frac{\partial^2 f}{\partial y \partial t}$$

and so on. Then

$$y' = f(y, t) \quad (21)$$

$$y'' = f_t + f f_y \quad (22)$$

Runge-Kutta Method: General

Higher order terms

$$y^{(3)} = f_{tt} + 2ff_{ty} + f^2 f_{yy} + ff_y^2 + f_t f_y \tag{23}$$

and

$$\begin{aligned} y^{(4)} &= f_{ttt} + 3ff_{tty} + 3f_t f_{ty} + 5ff_y f_{ty} + (2+f)ff_{tyy} + 3ff_t f_{yy} \\ &\quad +4f^2 f_y f_{yy} + f^3 f_{yyy} + f_t f_y^2 + ff_y^3 + f_{tt} f_y \end{aligned} \tag{24}$$

Runge-Kutta Method

$y(t+\tau)$ also can be written as

$$y(t+\tau) = y(t) + \alpha_1 c_1 + \alpha_2 c_2 + \cdots + \alpha_m c_m \tag{25}$$

with

$$\begin{aligned} c_1 &= \tau f(y,t), \\ c_2 &= \tau f(y+\nu_{21}c_1, t+\nu_{21}\tau), \\ c_3 &= \tau f(y+\nu_{31}c_1+\nu_{32}c_2, t+\nu_{31}\tau+\nu_{32}\tau) \\ &\vdots \end{aligned} \tag{26}$$

$$c_m = \tau f\left(y + \sum_{i=1}^{m-1} \nu_{mi} c_i, t + \tau \sum_{i=1}^{m-1} \nu_{mi}\right). \tag{27}$$

where α_i $(i = 1, 2, \cdots, m)$ and ν_{ij} $(i = 2, 3, \cdots, m$ and $j < i)$ are parameters to be determined.

Second-Order Runge-Kutta Method

If only the terms up to $\mathcal{O}(\tau^2)$ are kept in Eq. (20),

$$y(t+\tau) = y + \tau f + \frac{\tau^2}{2}(f_t + f f_y). \tag{28}$$

Truncate Eq. (25) up to the same order at $m = 2$:

$$y(t+\tau) = y(t) + \alpha_1 c_1 + \alpha_2 c_2 \tag{29}$$

From Eq. (26),

$$\begin{aligned} c_1 &= \tau f(y,t), \\ c_2 &= \tau f(y + \nu_{21} c_1, t + \nu_{21}\tau). \end{aligned} \tag{30}$$

Second-Order Runge-Kutta Method ver.2

Now expand c_2 up to $\mathcal{O}(\tau^2)$:

$$c_2 = \tau f + \nu_{21}\tau^2(f_t + f f_y) \tag{31}$$

From Eqs. (29)-(31) we obtain

$$y(t+\tau) = y(t) + (\alpha_1 + \alpha_2)\tau f + \alpha_2\tau^2\nu_{21}(f_t + f f_y). \tag{32}$$

By comparing Eq. (32) with Eq. (28), we have

$$\alpha_1 + \alpha_2 = 1, \tag{33}$$

and

$$\alpha_2\nu_{21} = \frac{1}{2}. \tag{34}$$

Two equations with three unknowns.

Second-Order Runge-Kutta Method ver.2

Choose $\alpha_1 = \frac{1}{2}$ then

$$\begin{cases} \alpha_2 = \frac{1}{2} \\ \nu_{21} = 1 \end{cases}$$

2nd-Order Runge-Kutta Method

$$y(t+\tau) = y(t) + \frac{1}{2}\tau f(y,t) + \frac{1}{2}\tau f(y+c_1, t+\tau)$$

with

$$c_1 = \tau f(y,t).$$

2nd-order RK is the same with the *Predictor-Corrector (or Modified Euler Method)*.

Fourth-Order Runge-Kutta Method

If we keep the terms in Eq. (20) and Eq. (25) up to $\mathcal{O}(\tau^4)$ we obtain the *4th-order RK method*.

4th-Order Runge-Kutta Method

$$y(t+\tau) = y(t) + \frac{1}{6}\left(c_1 + 2c_2 + 2c_3 + c_4\right), \tag{35}$$

with

$$\begin{aligned} c_1 &= \tau f(y,t), \\ c_2 &= \tau f\left(y + \frac{c_1}{2}, t + \frac{\tau}{2}\right), \\ c_3 &= \tau f\left(y + \frac{c_2}{2}, t + \frac{\tau}{2}\right), \\ c_4 &= \tau f(y + c_3, t + \tau) \end{aligned} \tag{36}$$

Runge-Kutta Method: Example I

Solve the differential equation using fourth-order RKM.

$$\frac{dx}{dt} = -x^3 + \sin t \tag{37}$$

with initial condition $x = 0$ at $t = 0$. $t \in [0, 10]$.

```
from math import sin
from numpy import arange
from pylab import plot,xlabel,ylabel,show

def f(x,t):
  y=-x**3 + sin(t)
  return y

#initial conditions
t_i=0.0
x_i=0.0

t_f=10.0     # End of the interval to calculate

N=1000   #number of steps
h=(t_f-t_i)/N
```

Runge-Kutta Method: Example II

```
pt=[]
px=[]

pt.append(t_i)
px.append(x_i)
t_list=arange(t_i,t_f,h)
x=x_i
for t in t_list:
  k1=h*f(x,t)
  k2=h*f(x+0.5*k1,t+0.5*h)
  k3=h*f(x+0.5*k2,t+0.5*h)
  k4=h*f(x+k3,t+h)
  x+=(k1+2.0*k2+2.0*k3+k4)/6.0
  px.append(x)
  pt.append(t+h)

plot(pt,px)
xlabel(r"$t$",fontsize=20)
ylabel(r"$x(t)$",fontsize=20)
show()
```

Homework

Solve the differential equation

$$y' + \frac{y}{\sqrt{x^2+1}} = \frac{1}{x+\sqrt{x^2+1}}$$

with initial condition $y = 0$ when $x = 0$ using second-order fourth-order RM.

Differential Equations with More than One Variable

- Many physics problems have more than one variable.
- **Simultaneous differential equations**
  - The derivative of each variable can depend on
    - any of the variables
    - or all of the variables
    - the independent variable t as well.

Example:

$$\frac{dx}{dt} = xy - x, \qquad \frac{dy}{dt} = y - xy + \sin^2 \omega t. \tag{38}$$

- Note that there is still only one *independent* variable t.
- The Eq. (38) is ordinary differential equation, not partial differential equation.

Differential Equations with More than One Variable

General Form

$$\frac{dx}{dt} = f_x(x, y, t), \qquad \frac{dy}{dt} = f_y(x, y, t), \tag{39}$$

where f_x and f_y are general, possibly nonlinear, functions of x, y, and t.

Vector Form

For an arbitrary number of variables,

$$\frac{d\mathbf{r}}{dt} = \mathbf{f}(\mathbf{r}, t), \tag{40}$$

where $\mathbf{r} = (x, y, \cdots)$ and $\mathbf{f}(\mathbf{r}, t) = (f_x(\mathbf{r}, t), f_y(\mathbf{r}, t), \cdots)$.

Euler's Method

Taylor expansion of a vector $\mathbf{r}$:

$$\mathbf{r}(t+h) = \mathbf{r}(t) + h\frac{d\mathbf{r}}{dt} + \mathcal{O}(h^2) = \mathbf{r}(t) + h\mathbf{f}(\mathbf{r},t) + \mathcal{O}(h^2). \quad (41)$$

Neglecting the terms of order h^2 and higher,

Euler's Method:

$$\mathbf{r}(t+h) = \mathbf{r}(t) + h\mathbf{f}(\mathbf{r},t)dt \quad (42)$$

Fourth-Order Runge-Kutta Method

Fourth-Order Runge-Kutta Method

$$
\begin{aligned}
\mathbf{k}_1 &= h\mathbf{f}(\mathbf{r}, t) \\
\mathbf{k}_2 &= h\mathbf{f}(\mathbf{r} + \frac{1}{2}\mathbf{k}_1, t + \frac{1}{2}h) \\
\mathbf{k}_3 &= h\mathbf{f}(\mathbf{r} + \frac{1}{2}\mathbf{k}_2, t + \frac{1}{2}h) \\
\mathbf{k}_4 &= h\mathbf{f}(\mathbf{r} + \mathbf{k}_3, t + h) \\
\mathbf{r}(t+h) &= \mathbf{r}(t) + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \qquad (43)
\end{aligned}
$$

Example: I

Example 8.5:

Solve

$$\frac{dx}{dt} = xy - x, \qquad \frac{dy}{dt} = y - xy + \sin^2 \omega t, \tag{44}$$

from $t = 0$ to $t = 10$ with $\omega = 1$ and initial condition $x = y = 1$ at $t = 0$.

```
from math import sin
from numpy import arange, array
from pylab import plot,xlabel,ylabel,show

def f(r,t):
  x=r[0]
  y=r[1]
  fx=x*y-x
  fy=y-x*y+sin(t)**2
  return array([fx,fy],float)

#initial conditions
t_i,t_f=0.0,10.0
```

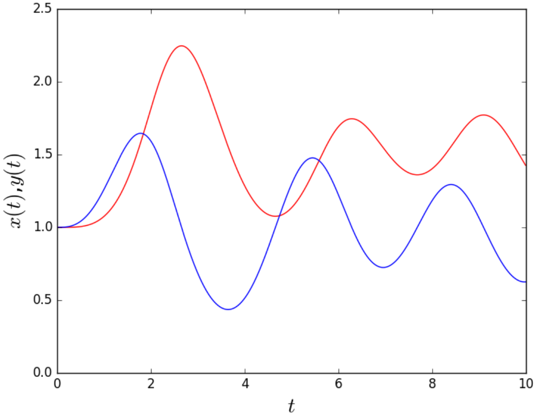
Example: II

```
x_i, y_i=1.0,1.0

N=1000    #number of steps
h=(t_f-t_i)/N
pt=[]
px=[]
py=[]

pt.append(t_i)
px.append(x_i)
py.append(y_i)
t_list=arange(t_i,t_f,h)
x=x_i

r=array([x_i,y_i],float)

for t in t_list:
  k1=h*f(r,t)
  k2=h*f(r+0.5*k1,t+0.5*h)
  k3=h*f(r+0.5*k2,t+0.5*h)
  k4=h*f(r+k3,t+h)
  r+=(k1+2.0*k2+2.0*k3+k4)/6.0
  px.append(r[0])
  py.append(r[1])
  pt.append(t+h)
```

Example: III

```
plot(pt,px,'r')
plot(pt,py,'b')
xlabel(r"$t$",fontsize=20)
ylabel(r"$x(t)$,$y(t)$",fontsize=20)
show()
```

Example: IV



Homeworks:

Exercises: 8.2 and 8.3

Second-Order Differential Equations

- Most equations in physics textbooks are second-order differential equations.
- Once we know how to solve the first-order ODE, solving the second-order ODE is easy.
- Solving the second-order ODE requires just the following trick.

Consider a case where there is only one dependent variable

$$\frac{d^2x}{dt^2} = f\left(x, \frac{dx}{dt}, t\right). \tag{45}$$

Here $f\left(x, \frac{dx}{dt}, t\right)$ can be any arbitrary function, including a nonlinear one.

- Example:

$$\frac{d^2x}{dt^2} = \frac{1}{x}\left(\frac{dx}{dt}\right)^2 + 2\frac{dx}{dt} - x^3 e^{-4t}. \tag{46}$$

Trick for the Second-Order Differential Equations

Trick for the Second-Order ODE

- Define a new quantity:

$$\frac{dx}{dt} \equiv y \tag{47}$$

- Then Eq. (45) can be rewritten as:

$$\frac{dy}{dt} = f(x, y, t). \tag{48}$$

- Now the second-order ODE becomes two first-order ODEs.

Higher-Order ODEs

Similar trick for higher-order ODEs

For example for a third-order ODE:

$$\frac{d^3x}{dt^3} = f\left(x, \frac{dx}{dt}, \frac{d^2x}{dt^2}, t\right). \tag{49}$$

Define two additional variables, y and z by

$$\frac{dx}{dt} \equiv y, \qquad \frac{dy}{dt} \equiv z \tag{50}$$

Then Eq. (49) becomes

$$\frac{dx}{dt} = f\left(x, y, z, t\right). \tag{51}$$

Now we have three first-order ODEs, Eqs. (50) and (51).

Generalization to equations more than one dependent variables

- The generalization is straightforward.

ODE with more than one dependent variables

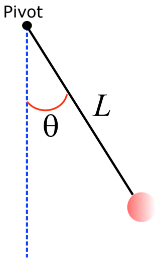
A set of simultaneous second-order ODEs can be written in vector form:

$$\frac{d^2\mathbf{r}}{dt} = \mathbf{f}\left(\mathbf{r}, \frac{d\mathbf{r}}{dt}, t\right). \tag{52}$$

Eq. (52) is equivalent to the first-order ODEs:

$$\frac{d\mathbf{r}}{dt} = \mathbf{s}, \qquad \frac{d\mathbf{s}}{dt} = \mathbf{f}\left(\mathbf{r}, \mathbf{s}, t\right). \tag{53}$$

Example 8.6: The Nonlinear Pendulum



- θ: the angle of displacement of the arm from the vertical
- m: the mass of the bob
- L: length of the arm

- Newton's law:

$$mL\frac{d^2\theta}{dt^2} = -mg\sin\theta, \tag{54}$$

or equivalently,

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin\theta, \tag{55}$$

Divide two first-order ODEs

Divide Eq. (55) into two first-order ODEs:

$$\frac{d\theta}{dt} = \omega, \tag{56}$$

and

$$\frac{d\omega}{dt} = -\frac{g}{L}\sin\theta \tag{57}$$

Using EM I

Let $\mathbf{r} = (\theta, \omega)$.

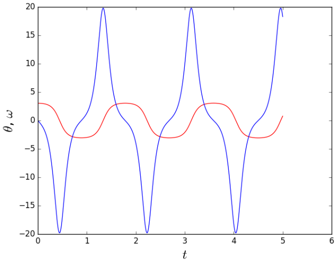
```
from math import sin, pi
from numpy import arange, array
from pylab import plot, xlabel, ylabel, show

g=9.81
l=0.1

def f(r,t):
  theta=r[0]
  omega=r[1]
  f_theta=omega
  f_omega=-(g/l)*sin(theta)
  #f_omega=-(g/l)*theta
  return array([f_theta,f_omega],float)

#initial conditions
t_i=0.0
theta_i=pi-0.1
omega_i=0.0

t_f=5.0     # End of the interval to calculate

h=0.00001   #number of steps
pt=[]
```

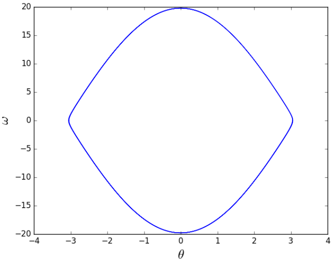
Using EM II

```
ptheta=[]
pomega=[]
r=[theta_i,omega_i]

theta=theta_i
omega=omega_i
t=t_i

pt.append(t)
ptheta.append(theta)
pomega.append(omega)

while t<=t_f:
    r+=h*f(r,t)
    t+=h
    ptheta.append(r[0])
    pomega.append(r[1])
    pt.append(t)

plot(pt,ptheta,'r')
plot(pt,pomega,'b')
xlabel(r"$t$",fontsize=20)
ylabel(r"$\theta$, $\omega$",fontsize=20)
show()
```

Using EM III

```
plot(ptheta,pomega)
xlabel(r"$\theta$",fontsize=20)
ylabel(r"$\omega$",fontsize=20)
show()
```





Using 2nd-Order RKM I

```
from math import sin, pi
from numpy import arange, array
from pylab import plot, xlabel, ylabel, show

g=9.81
l=0.1

def f(r,t):
  theta=r[0]
  omega=r[1]
  f_theta=omega
  f_omega=-(g/l)*sin(theta)
  #f_omega=-(g/l)*theta
  return array([f_theta, f_omega], float)

#initial conditions
t_i=0.0
theta_i=pi-0.1
omega_i=0.0

t_f=5.0     # End of the interval to calculate

h=0.0001   #number of steps
pt=[]
ptheta=[]
```

Using 2nd-Order RKM II

```
pomega=[]
r=[theta_i ,omega_i]

theta=theta_i
omega=omega_i
t=t_i

pt.append(t)
ptheta.append(theta)
pomega.append(omega)

while t<=t_f:
  k1=h*f(r,t)
  k2=h*f(r+0.5*k1,t+0.5*h)
  r+=k2
  t+=h
  ptheta.append(r[0])
  pomega.append(r[1])
  pt.append(t)

plot(pt,ptheta,'r')
plot(pt,pomega,'b')
xlabel(r"$t$",fontsize=20)
ylabel(r"$\theta$, $\omega$",fontsize=20)
show()
```

Using 2nd-Order RKM III

```
plot(ptheta,pomega)
xlabel(r"$\theta$",fontsize=20)
ylabel(r"$\omega$",fontsize=20)
show()
```

Using 4th-Order RKM I

```
from math import sin, pi
from numpy import arange, array
from pylab import plot, xlabel, ylabel, show

g=9.81
l=0.1

def f(r,t):
    theta=r[0]
    omega=r[1]
    f_theta=omega
    f_omega=-(g/l)*sin(theta)
    #f_omega=-(g/l)*theta
    return array([f_theta, f_omega], float)

#initial conditions
t_i=0.0
theta_i=pi-0.1
omega_i=0.0

t_f=5.0      # End of the interval to calculate

h=0.0001    #number of steps
pt=[]
ptheta=[]
```

Using 4th-Order RKM II

```
pomega=[]
r=[theta_i ,omega_i]

theta=theta_i
omega=omega_i
t=t_i

pt.append(t)
ptheta.append(theta)
pomega.append(omega)

while t<=t_f:
  k1=h*f(r,t)
  k2=h*f(r+0.5*k1,t+0.5*h)
  k3=h*f(r+0.5*k2,t+0.5*h)
  k4=h*f(r+k3,t+h)
  r+=(k1+2*k2+2*k3+k4)/6.0
  t+=h
  ptheta.append(r[0])
  pomega.append(r[1])
  pt.append(t)

plot(pt,ptheta,'r')
plot(pt,pomega,'b')
xlabel(r"$t$",fontsize=20)
```

Using 4th-Order RKM III

```
ylabel(r"$\theta$, $\omega$", fontsize=20)
show()


plot(ptheta, pomega)
xlabel(r"$\theta$", fontsize=20)
ylabel(r"$\omega$", fontsize=20)
show()
```

Homework

Damped Harmonic Montion: Solve the second order differential equation:

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0$$

using EM, 2nd-order RKM, and 4th-order RKM. There are three different types of damping:

1. $c^2 - 4mk > 0$ overdamping
2. $c^2 - 4mk = 0$ critical damping
3. $c^2 - 4mk < 0$ underdamping

Plot x vs. t for each case.

Revisit Newton's Equation of motion

Newton's equation of motion in one-dimensional space

$$\frac{d^2x}{dt^2} = \frac{F(x,v,t)}{m} \equiv f(x,v,t) \tag{58}$$

Disassemble Eq. (58) into two steps:

$$\frac{dv}{dt} = f(x,v,t) \tag{59}$$

and

$$\frac{dx}{dt} = v \tag{60}$$

Revisit Newton's Equation of motion

By using the 4th-order Runge-Kutta method

$$v_{i+1} = v_i + \frac{1}{6}(c_1 + 2c_2 + 2c_3 + c_4), \tag{61}$$

where

$$\begin{aligned} c_1 &= \tau f(x_i, v_i, t_i) \\ c_2 &= \tau f\left(x_i + \frac{q_1}{2}, v_i + \frac{c_1}{2}, t_i + \frac{\tau}{2}\right) \\ c_3 &= \tau f\left(x_i + \frac{q_2}{2}, v_i + \frac{c_2}{2}, t_i + \frac{\tau}{2}\right) \\ c_4 &= \tau f(x_i + q_3, v_i + c_3, t + \tau) \end{aligned} \tag{62}$$

Revisit Newton's Equation of motion

And from Eq. (60)

$$x_{i+1} = x_i + \frac{1}{6}(q_1 + 2q_2 + 2q_3 + q_4), \tag{63}$$

where

$$\begin{aligned} q_1 &= \tau v_i \\ q_2 &= \tau\left(v_i + \frac{c_1}{2}\right) \\ q_3 &= \tau\left(v_i + \frac{c_2}{2}\right) \\ q_4 &= \tau(v_i + c_3) \end{aligned} \tag{64}$$

From Eq. (63) and Eq. (64)

$$x_{i+1} = x_i + \frac{1}{6}\left[\tau v_i + 2\tau\left(v_i + \frac{c_1}{2}\right) + 2\tau\left(v_i + \frac{c_2}{2}\right) + \tau(v_i + c_3)\right] \tag{65}$$

Or

$$x_{i+1} = x_i + \tau v_i + \frac{\tau}{6}\left[c_1 + c_2 + c_3\right] \tag{66}$$

Revisit Newton's Equation of motion

Therefore, we only need to calculate c_i's!

Newton's Equation of Motion

$$x_{i+1} = x_i + \tau v_i + \frac{\tau}{6}\left(c_1 + c_2 + c_3\right)$$

$$v_{i+1} = v_i + \frac{1}{6}(c_1 + 2c_2 + 2c_3 + c_4)$$

where

$$\begin{aligned} c_1 &= \tau f(x_i, v_i, t_i) \\ c_2 &= \tau f\left(x_i + \frac{\tau v_i}{2}, v_i + \frac{c_1}{2}, t_i + \frac{\tau}{2}\right) \\ c_3 &= \tau f\left(x_i + \frac{\tau v_i}{2} + \frac{\tau c_1}{4}, v_i + \frac{c_2}{2}, t_i + \frac{\tau}{2}\right) \\ c_4 &= \tau f\left(x_i + \tau v_i + \frac{\tau c_2}{2}, v_i + c_3, t + \tau\right) \end{aligned}$$

Revisit Example 8.6: The Nonlinear Pendulum I

Divide two first-order ODEs

Divide Eq. (55) into two first-order ODEs:

$$\frac{d\theta}{dt} = \omega, \tag{67}$$

and

$$\frac{d\omega}{dt} = -\frac{g}{L}\sin\theta \tag{68}$$

```
from math import sin, pi
from numpy import arange, array
from pylab import plot, xlabel, ylabel, show

g=9.81
l=0.1

def f(theta, t):
  y=-(g/l)*sin(theta)
  #f_omega=-(g/l)*theta
```

Revisit Example 8.6: The Nonlinear Pendulum II

```
    return y

#initial conditions
t_i=0.0
theta_i=pi-0.1
omega_i=0.0

t_f=5.0      # End of the interval to calculate

h=0.0001    #number of steps
pt=[]
ptheta=[]
pomega=[]

theta=theta_i
omega=omega_i
t=t_i

pt.append(t)
ptheta.append(theta)
pomega.append(omega)

while t<=t_f:
    k1=h*f(theta,t)
    k2=h*f(theta+0.5*h*omega,t+0.5*h)
```

Revisit Example 8.6: The Nonlinear Pendulum III

```
    k3=h*f(theta+0.5*h*omega+h*k1/4.0,t+0.5*h)
    k4=h*f(theta+h*omega+h*k2*0.5,t+h)
    theta+=h*omega+h*(k1+k2+k3)/6.0
    omega+=(k1+2*k2+2*k3+k4)/6.0
    t+=h
    ptheta.append(theta)
    pomega.append(omega)
    pt.append(t)

plot(pt,ptheta,'r')
plot(pt,pomega,'b')
xlabel(r"$t$",fontsize=20)
ylabel(r"$\theta$, $\omega$",fontsize=20)
show()


plot(ptheta,pomega)
xlabel(r"$\theta$",fontsize=20)
ylabel(r"$\omega$",fontsize=20)
show()
```

Example: Van der Pol Oscillator

Van der Pol Oscillator

$$\frac{d^2x}{dt^2} = \mu(x_0^2 - x^2)\frac{dx}{dt} - \omega^2 x$$

or

$$\frac{d^2x}{dt^2} = f(t, x, v) = \mu(x_0^2 - x^2)v - \omega^2 x$$

x_0, μ, ω are given constants.

From the previous page:

$$\begin{aligned}
x_{i+1} &= x_i + \tau v_i + \tfrac{\tau}{6}\left(c_1 + c_2 + c_3\right)\\
v_{i+1} &= v_i + \tfrac{1}{6}(c_1 + 2c_2 + 2c_3 + c_4)\\
c_1 &= \tau\left[\mu(x_0^2 - x_i^2)v_i - \omega^2 x_i\right]\\
c_2 &= \tau\left[\mu\left(x_0^2 - \left(x_i + \tfrac{\tau v_i}{2}\right)^2\right)\left(v_i + \tfrac{c_1}{2}\right) - \omega^2\left(x_i + \tfrac{\tau v_i}{2}\right)\right]\\
c_3 &= \tau\left[\mu\left(x_0^2 - \left(x_i + \tfrac{\tau v_i}{2} + \tfrac{\tau c_1}{4}\right)^2\right)\left(v_i + \tfrac{c_2}{2}\right) - \omega^2\left(x_i + \tfrac{\tau v_i}{2} + \tfrac{\tau c_1}{4}\right)\right]\\
c_4 &= \tau\left[\mu\left(x_0^2 - \left(x_i + \tau v_i + \tfrac{\tau c_2}{2}\right)^2\right)\left(v_i + c_3\right) - \omega^2\left(x_i + \tau v_i + \tfrac{\tau c_2}{2}\right)\right]
\end{aligned}$$

Example: Van der Pol Oscillator I

Van der Pol Oscillator

$$\frac{d^2x}{dt^2} = \mu(x_0^2 - x^2)\frac{dx}{dt} - \omega^2 x$$

or

$$\frac{d^2x}{dt^2} = f(t, x, v) = \mu(x_0^2 - x^2)v - \omega^2 x$$

$x_0 = 1$, $\mu = 1$, $\omega = 1$ are given constants.

```
from math import sin, pi
from numpy import arange, array
from pylab import plot, xlabel, ylabel, show

def f(x,v,t):
  omega=1.0
  mu=1.0
  x_0=1.0
  y=mu*(x_0**2-x**2)*v-omega**2*x
  return y

#initial conditions
```

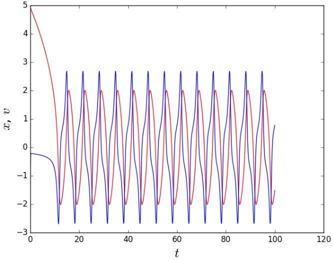
Example: Van der Pol Oscillator II

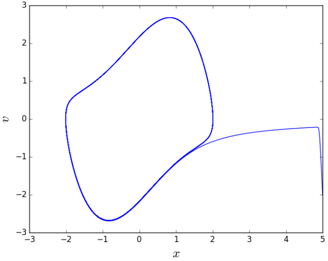
```
t_i =0.0
x_i =5.0
v_i =-2.0

t_f =100.0      # End of the interval to calculate

h=0.001    #number of steps
pt=[]
px=[]
pv=[]

x=x_i
v=v_i
t=t_i

pt.append(t)
px.append(x)
pv.append(v)

while t<=t_f:
  k1=h*f(x,v,t)
  k2=h*f(x+0.5*h*v,v+0.5*k1,t+0.5*h)
  k3=h*f(x+0.5*h*v+h*k1/4.0,v+0.5*k2,t+0.5*h)
  k4=h*f(x+h*v+h*k2*0.5,v+k3,t+h)
  x+=h*v+h*(k1+k2+k3)/6.0
```

Example: Van der Pol Oscillator III

```
    v+=(k1+2*k2+2*k3+k4)/6.0
    t+=h
    px.append(x)
    pv.append(v)
    pt.append(t)

plot(pt,px,'r')
plot(pt,pv,'b')
xlabel(r"$t$",fontsize=20)
ylabel(r"$x$, $v$",fontsize=20)
show()


plot(px,pv)
xlabel(r"$x$",fontsize=20)
ylabel(r"$v$",fontsize=20)
show()
```

Example: Van der Pol Oscillator IV





Revisit: Exmaple 8.6 Simple Pendulum I

With Animation!

```
"""
===========================
The simple pendulum problem
===========================

This animation illustrates the double pendulum problem.
"""

# Double pendulum formula translated from the C code at
# http://www.physics.usyd.edu.au/~wheat/dpend_html/solve_dpend.c

from numpy import sin, cos, pi
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

G = 9.81  # acceleration due to gravity, in m/s^2
L = .10  # length of pendulum 1 in m
M = 1.0  # mass of pendulum 1 in kg

# create a time array from 0..100 sampled at 0.05 second steps
dt = 0.01
#t = np.arange(0.0, 20, dt)
```

Revisit: Exmaple 8.6 Simple Pendulum II

```
# th1 and th2 are the initial angles (degrees)
# w10 and w20 are the initial angular velocities (degrees per second)
theta_i=pi-0.1
w1 = 0.0

# initial state
state = [theta_i, w1]

# integrate your ODE using scipy.integrate.
def f(theta,t):
  y=-(G/L)*sin(theta)
  #f_omega=-(g/l)*theta
  return y

#initial conditions
t_i=0.0
theta_i=pi-0.1
omega_i=0.0

t_f=5.0      # End of the interval to calculate

#h=0.0001    #number of steps
h=dt
pt=[]
ptheta=[]
```

Revisit: Exmaple 8.6 Simple Pendulum III

```
pomega=[]

theta=theta_i
omega=omega_i
t=t_i

pt.append(t)
ptheta.append(theta)
pomega.append(omega)

while t<=t_f:
  k1=h*f(theta,t)
  k2=h*f(theta+0.5*h*omega,t+0.5*h)
  k3=h*f(theta+0.5*h*omega+h*k1/4.0,t+0.5*h)
  k4=h*f(theta+h*omega+h*k2*0.5,t+h)
  theta+=h*omega+h*(k1+k2+k3)/6.0
  omega+=(k1+2*k2+2*k3+k4)/6.0
  t+=h
  ptheta.append(theta)
  pomega.append(omega)
  pt.append(t)

x = L*sin(ptheta)
y = -L*cos(ptheta)
```

Revisit: Exmaple 8.6 Simple Pendulum IV

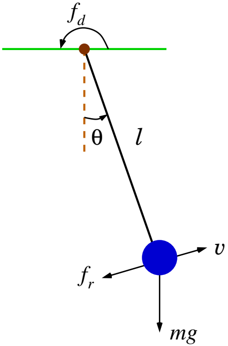
```
fig = plt.figure()
ax = fig.add_subplot(111, autoscale_on=False, xlim=(-0.2, 0.2), ylim=(-0.2, 0.2
ax.grid()

line, = ax.plot([], [], 'o-', lw=2)
time_template = 'time = %.1fs'
time_text = ax.text(0.05, 0.9, '', transform=ax.transAxes)


def init():
    line.set_data([], [])
    time_text.set_text('')
    return line, time_text


def animate(i):
    thisx = [0, x[i]]
    thisy = [0, y[i]]

    line.set_data(thisx, thisy)
    time_text.set_text(time_template % (i*dt))
    return line, time_text

ani = animation.FuncAnimation(fig, animate, np.arange(1, len(y)),
```

Revisit: Exmaple 8.6 Simple Pendulum V

```
                              interval=25, blit=True, init_func=init)

# ani.save('double_pendulum.mp4', fps=15)
plt.show()
```

Homework

Driven Pendulum



A point mass m is attached to the lower end of massless rod of length l. The pendulum is confined to a vertical plane, acted on by a driving force f_d and a resistive force f_r (see the figure). The motion of the pendulum is described by Newton's equation along the tangential direction of the circular motion of the mass,

$$ma_t = -mg\sin\theta + f_d + f_r,$$

where $a_t = l d^2\theta/dt^2$. If the driving force is periodic as $f_d(t) = f_0\cos\omega_0 t$ and $f_r = -\kappa v = -\kappa l d\theta/dt$ then the equation of motion becomes

$$l\frac{d^2\theta}{dt^2} = -mg\sin\theta - \kappa l\frac{d\theta}{dt} + f_0\cos\omega_0 t. \tag{69}$$

If we rewrite Eq. (69) in a dimensionless form with $\sqrt{l/g}$ chosen as the unit of time, we obtain

$$\frac{d^2\theta}{dt^2} + q\frac{d\theta}{dt} + \sin\theta = b\cos\omega_0 t, \tag{70}$$

where $q = \kappa/m$, $b = f_0/ml$, and ω_0 is the angular frequency of the driving force. Solve Eq. (70) numerically and plot the trajectory in phase space when (1) $(\omega_0, q, b) = (2/3, 0.5.0.9)$ and (2)$(\omega_0, q, b) = (2/3, 0.5.1.15)$

Boundary Value Problems

For a second-order differential equation,

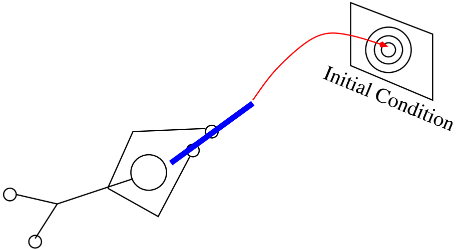
$$y'' = f(x, y, y'),$$

there are four possible boundary condition sets:

1. $y(x_0) = y_0$ and $y(x_1) = y_1$
2. $y(x_0) = y_0$ and $y'(x_1) = v_1$
3. $y'(x_0) = v_0$ and $y(x_1) = y_1$
4. $y'(x_0) = v_0$ and $y'(x_1) = v_1$

- Shooting Method
- Relaxation Method

Shooting Method



Basic idea: change the given boundary condition into the corresponding initial condition through a trial-and-error method.

Prerequisite: **Secant Method or Bisection Method** to find a root of equation and the basic **algorithm(s) for ODE**.

Shooting Method

Convert a single second-order differential equation

$$\frac{d^2y_1}{dx^2} = f(x, y_1, y_1')$$

into two first-order differential equations:

$$y_1' \equiv \frac{dy_1}{dx} = y_2$$

and

$$\frac{dy_2}{dx} = f(x, y_1, y_2)$$

with boundary condition, for example, $y_1(x_i) = u_0$ and $y_1(x_f) = u_1$, where x_i and x_f are the location of boundary.

Shooting Method

How to change the given boundary condition into the initial condition?

- $y_1(x_i)$ is given
- **guess** $y_1'(x_i) = y_2(x_i) \equiv \alpha$.
  - Here the parameter α will be adjusted to satisfy $y_1(x_f) = u_1$.
  - For this we will use the **secant method (or bisection method)**.
- Let us define a function of α as

$$g(\alpha) \equiv u_\alpha(x_f) - u_1,$$

  - $u_\alpha(x_f)$ is the boundary condition obtained with the assumption that $y_2(x_i) = \alpha$
    - $u_\alpha(x_f)$ is calculated by the usual algorithm for initial value problem (for example, by applying Runge-Kutta method) with assumed initial value $y_2(x_i) = \alpha$.
  - u_1 is the true boundary condition.
- Using the secant method, find the value α which satisfy

$$g(\alpha) = 0$$

Example 8.8: Vertical Position of a Thrown Ball (Bisection Method) I

Solve the differential equation

$$\frac{d^2x}{dt^2} = -g \tag{71}$$

with the b.c. $x = 0$ at time $t = 0$ and $t = 10$. Rewrite Eq. (71) as

$$\frac{dx}{dt} = v, \qquad \frac{dv}{dt} = -g \tag{72}$$

Example 8.8: Vertical Position of a Thrown Ball (Bisection Method) II

```
from numpy import array,arange
from matplotlib import pyplot as plt

g=9.81
t_i,t_f=0.0,10.0
N=1000                    # number of steps for RKM
h=(t_f-t_i)/N

tolerance=1e-10

def f(r):
  x=r[0]
  v=r[1]
  fx=v
  fy=-g
  return array([fx,fy],float)

def RK4(r):
  k1=h*f(r)
  k2=h*f(r+0.5*k1)
  k3=h*f(r+0.5*k2)
  k4=h*f(r+k3)
```

Example 8.8: Vertical Position of a Thrown Ball (Bisection Method) III

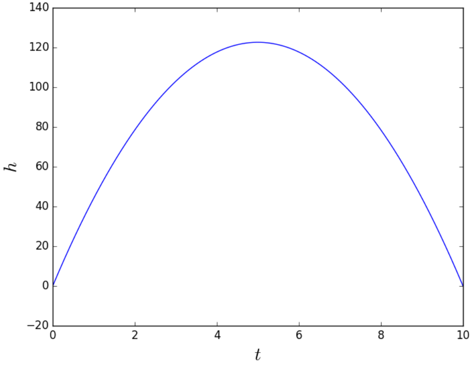
```
    r+=(k1+2*k2+2*k3+k4)/6

    return r

def height(v):
    r=array([0.0,v],float)
    for t in arange(t_i,t_f,h):
        r=RK4(r)
    return r[0]

def bisection(v1,v2):
    h1=height(v1)
    h2=height(v2)
    while abs(h2-h1)>tolerance:
        v_m=(v1+v2)/2
        h_m=height(v_m)
        if h1*h_m>0:
            v1=v_m
            h1=h_m
        else:
            v2=v_m
            h2=h_m
```

Example 8.8: Vertical Position of a Thrown Ball (Bisection Method) IV

```
    v=(v1+v2)/2
    return v

v1=0.01
v2=1000.0

v=bisection(v1,v2)

print("The required initial velocity is",v,"m/s");

px=[]
pt=[]
px.append(0.0)
pt.append(0.0)
r=array([0.0,v],float)
for t in arange(t_i,t_f,h):
    r=RK4(r)
    px.append(r[0])
    pt.append(t+h)

plt.plot(pt,px)
plt.xlabel(r'$t$',fontsize=20)
plt.ylabel(r'$h$',fontsize=20)
```

Example 8.8: Vertical Position of a Thrown Ball (Bisection Method) V

```
plt.show()
```



Example 8.8: Vertical Position of a Thrown Ball (Secant Method) I

To solve the differential equation (72) using bisection method,

- We need two guessed v_i's.
- The true v_i should be located between two estimated v_i's.

To avoid such ambiguity we can also use secant method!

```
from numpy import array, arange
from matplotlib import pyplot as plt

g=9.81
t_i, t_f=0.0,10.0
N=1000              # number of steps for RKM
h=(t_f-t_i)/N

tolerance=1e-15

def f(r):
  x=r[0]
  v=r[1]
  fx=v
  fy=-g
  return array([fx,fy],float)
```

Example 8.8: Vertical Position of a Thrown Ball (Secant Method) II

```
def RK4(r):
  k1=h*f(r)
  k2=h*f(r+0.5*k1)
  k3=h*f(r+0.5*k2)
  k4=h*f(r+k3)
  r+=(k1+2*k2+2*k3+k4)/6

  return r

def gg(x,v,x_f):
  r=array([x,v],float)
  for t in arange(t_i,t_f,h):
    r=RK4(r)
  return (r[0]-x_f)

def secant(r,dv):
  x=r[0]
  v=r[1]
  v1=v+dv
  r1=array([x,v],float)
  while abs(dv)>=tolerance:
    d=gg(x,v1,0.0)-gg(x,v,0.0)
```

Example 8.8: Vertical Position of a Thrown Ball (Secant Method) III

```
    v2=v1-gg(x,v1,0.0)*(v1-v)/d
    v=v1
    v1=v2
    dv=v1-v
  return v

v1=0.01
x_i=0.0
r=array([x_i,v1],float)
v=secant(r,10.0)


print("The required initial velocity is",v,"m/s");

px=[]
pt=[]
px.append(0.0)
pt.append(0.0)
r=array([0.0,v],float)
for t in arange(t_i,t_f,h):
  r=RK4(r)
  px.append(r[0])
  pt.append(t+h)
```

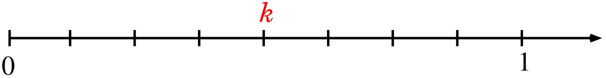
Example 8.8: Vertical Position of a Thrown Ball (Secant Method) IV

```
plt.plot(pt,px)
plt.xlabel(r'$t$',fontsize=20)
plt.ylabel(r'$h$',fontsize=20)
plt.show()
```

Relaxation Method

$$\frac{d^2y}{dx^2} = f(x,y) \Rightarrow \frac{d^2y}{dx^2} - f(x,y) = 0 \tag{73}$$

(1) Divide the given interval into discrete N intervals (discretization).



(2) Use the definition of the numerical second order derivative to rewrite Eq. (73) as

$$\frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} - f(x_k, y_k) = 0 \tag{74}$$

at $x = x_k$. Eq. (74) becomes

$$y_{k+1} - 2y_k + y_{k-1} - h^2 f(x_k, y_k) = 0 \tag{75}$$

or equivalently

$$y_k = \frac{y_{k+1} + y_{k-1} - h^2 f(x_k, y_k)}{2} \tag{76}$$

Relaxation Method

(3) Using Eq. (76), keeping the boundary condition, iteratively calculate y_k as

$$y_k^{(n+1)} = \frac{y_{k+1}^{(n)} + y_{k-1}^{(n)} - h^2 f(x_k, y_k^{(n)})}{2}, \tag{77}$$

for all k. Here $y_k^{(n)}$ is the value of y_k at nth iteration.

Relaxation Method

Relaxation Method

$$y_k^{(n+1)} = \frac{y_{k+1}^{(n)} + y_{k-1}^{(n)} - h^2 f(x_k, y_k^{(n)})}{2}$$

Successive Over relaxation Method

$$y_k^{(n+1)} = w\left(\frac{y_{k+1}^{(n)} + y_{k-1}^{(n)} - h^2 f(x_k, y_k^{(n)})}{2}\right) + (1-w)y_k^{(n)}$$

where w is called as *over relaxation parameter* and $w \in [0, 2]$. Usually $w > 1$ is used to speed up the slow converging process and $w < 1$ is frequently used to establish convergence of diverging iterative process or speed up the convergence of an overshooting process.

Example 8.8: Vertical Position of a Thrown Ball (Relaxation Method) I

```
from numpy import array,arange,ones,copy,max
from matplotlib import pyplot as plt

g=9.81
t_i,t_f=0.0,10.0
N=100    #number of interval in time
h=(t_f-t_i)/N

def f():
  return (-g)

t=list(arange(t_i,t_f,h))
t.append(t[len(t)-1]+h)
t[0]=0.0
leng_t=len(t)
x=list(ones(leng_t,float))
i=1
for i in range(1,leng_t-1):
  x[i]=20.0
x[0]=0.0
x[leng_t-1]=0.0
xtmp=copy(x)
```

Example 8.8: Vertical Position of a Thrown Ball (Relaxation Method) II

```
w=0.8
tolerance=1e-6
delta=1.0
while delta>tolerance:
    for i in range(leng_t):
        if i==0 or i==leng_t-1:
            xtmp[i]=x[i]
        else:
            xtmp[i]=(x[i+1]+x[i-1]-f()*h**2)/2.0
    delta=max(abs(x-xtmp))
    x,xtmp=xtmp,x
#   xtmp=copy(x)

plt.plot(t,x)
plt.xlabel(r'$t$',fontsize=20)
plt.ylabel(r'$h$',fontsize=20)
plt.show()
```

Eigenvalue Problems

Time-independent Shrödinger Equation:

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V(x)\psi(x) = E\psi(x). \tag{78}$$

Infinite square potential:

$$V(x) = \begin{cases} 0 & \text{for } 0 < x < L \\ \infty & \text{otherwise,} \end{cases} \tag{79}$$

where L is the width of the well.

- The probability of finding the particle in the region with $V(x) = \infty$.
  - Corresponding boundary conditions: $\psi(x = 0) = 0$ and $\psi(x = L) = 0$.

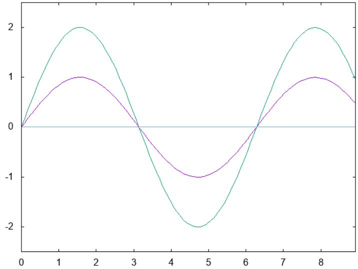
Rewrite Shrödinger Equation

Since Eq. (78) is second-order, rewrite Eq. (78) as:

$$\frac{d\psi}{dx} = \phi, \qquad \frac{d\phi}{dx} = \frac{2m}{\hbar^2}\left[V(x) - E\right]\psi. \tag{80}$$

- To calculate a solution, we need two initial conditions:
  - one for each ψ and ϕ.
  - We already know $\psi(x = 0) = 0$.
  - So we guess an initial value for $\phi(x = 0)$, then try to calculate the solution from $x = 0$ to $x = L$.

Problem!!



Problem!!

- By changing the $\phi(x=0)$ can not find a condition to satisfy b.c. $\psi(x=L)=0$!!
- Because the equation is linear
  - For example, if we double the $\phi(x=0)$, then ψ becomes double and does not satisfy the other b.c. at $x=L$ (see the green line in the figure).

How to resolve the problem?

Change the Energy, E

Instead of changing $\phi(0)$, change E to find the value for $\psi = 0$ at $x = L$.

Unknown b.c. on $\phi = d\psi/dx$

Than does'n matter!

- In this case, the value of ϕ only affects the amplitude of ψ.
- The correct amplitude of ψ can be determined by the normalization condition

$$\int |\psi|^2 \, dx = 1 \tag{81}$$

- i.e., just by dividing ψ by $\int |\psi|^2 \, dx$ numerically.

Ex.8.9: Ground State Energy in a Square Well and Wave Function I

L is given by the Bohr radius, $a_0 = 5.292 \times 10^{-11}$.

```
from numpy import array, arange,dot,sqrt
from matplotlib import pyplot as plt

# Constants
m=9.1094e-31    # mass of electron
hbar=1.0546e-34
e=1.6022e-19
L=5.2918e-11    # Bohr radius
N=1000
h=L/N

# Potential Function
def V(x):
  return 0.0

def f(r,x,E):
  psi=r[0]
  phi=r[1]
  fpsi=phi
  fphi=(2*m/hbar**2)*(V(x)-E)*psi
  return array([fpsi,fphi],float)
```

Ex.8.9: Ground State Energy in a Square Well and Wave Function II

```
# Calculate the wavefucntion for a particular Energy

def RK4(r,x,E):
  k1=h*f(r,x,E)
  k2=h*f(r+0.5*k1,x+0.5*h,E)
  k3=h*f(r+0.5*k2,x+0.5*h,E)
  k4=h*f(r+k3,x+h,E)
  r+=(k1+2*k2+2*k3+k4)/6
  return r

def solve(E):
  psi=0.0
  phi=1.0
  r=array([psi,phi],float)

  for x in arange(0,L,h):
    r=RK4(r,x,E)
  return r[0]

# Main program to find the energy using the secant method
E1=0.0
E2=e
```

Ex.8.9: Ground State Energy in a Square Well and Wave Function III

```
psi2=solve(E1)

tolarence=e/1000
while abs(E2-E1)>tolarence:
  psi1,psi2=psi2,solve(E2)
  E1,E2=E2,E2-psi2*(E2-E1)/(psi2-psi1)

print("E=",E2/e,"eV")

# Calculate the psi

ppsi=[]
pphi=[]
px=[]

ppsi.append(0.0)
pphi.append(1.0)
px.append(0.0)

r=array([ppsi[0],pphi[0]],float)

for x in arange(0,L,h):
  r=RK4(r,x,E2)
```

Ex.8.9: Ground State Energy in a Square Well and Wave Function IV

```
    ppsi.append(r[0])
    pphi.append(r[1])
    px.append(x+h)

# Normalize psi
integ=0.0
for i in range(len(px)):
    integ+=h*ppsi[i]**2
norm_ppsi=ppsi/sqrt(integ)


plt.plot(px,norm_ppsi)
plt.xlim(0,L)
plt.show()
```

Ex.8.9: Ground State Energy in a Square Well and Wave Function V