计算物理 第二部分
第2讲

李强　北京大学物理学院中楼411
qliphy0@pku.edu.cn， 15210033542
http://www.phy.pku.edu.cn/~qiangli/CP2017.html

2. 常微分方程

概念介绍

初值问题： 欧拉法， Runge-Kutta法

微分方程组： 电磁场中电子运动

高阶微分方程

薛定谔方程

混沌，Lorenz吸引子

刚性微分方程，Gear方法

边值问题：

1D泊松方程 差分法(三对角矩阵)，打靶法，

混合边值问题

引言

微分方程： 含有自变量、未知函数及其导数的方程
常微分方程： 未知函数只含有一个变量
偏微分方程： 未知函数含有多个变量
阶： 微分方程中未知函数的导数或偏导数的最高阶数

$$\frac{\mathrm{d}y}{dt} = a - by$$ 一阶常微分方程

$$\frac{\partial^2 u(x,t)}{\partial t^2} = a\frac{\partial^2 u(x,t)}{\partial x^2}$$ 二阶偏微分方程

n阶微分方程通解： 含有n个任意函数 → 定解条件

引言

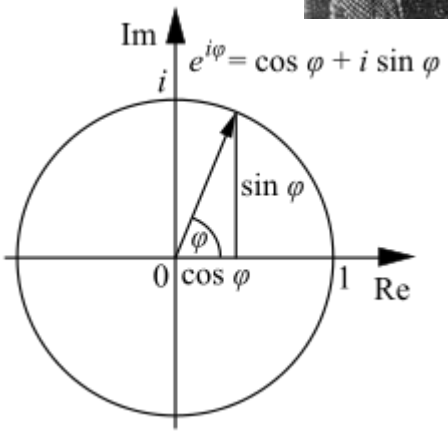
初值问题：　给出解在某一点（或初始点）的状态
边值问题：　给出解在至少两点（或边界）的状态

$$\frac{dy}{dx} = y'(x) = f(x,y), \qquad a \le x \le b$$
$$y(a) = y_0$$

$$\frac{d^2y}{dx^2} = y''(x) = f(x,y,y'), \qquad a \le x \le b$$
$$a_0 y(a) + b_0 y'(a) = r_0$$
$$a_1 y(b) + b_1 y'(b) = r_1$$

第一类边界条件：b_0=b_1=0　　Dirichlet条件
第二类边界条件：a_0=a_1=0　　Neumann条件
第三类边界条件：混合类型　　Robinns条件

莱昂哈德·欧拉（Leonhard Euler，1707年4月15日~1783年9月18日），瑞士数学家、自然科学家。1707年4月15日出生于瑞士的巴塞尔，1783年9月18日于俄国圣彼得堡去世。欧拉出生于牧师家庭，自幼受父亲的影响。13岁时入读巴塞尔大学，15岁大学毕业，16岁获得硕士学位。欧拉是18世纪数学界最杰出的人物之一，他不但为数学界作出贡献，更把整个数学推至物理的领域。他是数学史上最多产的数学家，平均每年写出八百多页的论文，还写了大量的力学、分析学、几何学、变分法等的课本，《无穷小分析引论》、《微分学原理》、《积分学原理》等都成为数学界中的经典著作。欧拉对数学的研究如此之广泛，因此在许多数学的分支中也可经常见到以他的名字命名的重要常数、公式和定理。[1] 此外欧拉还涉及建筑学、弹道学、航海学等领域。



Euler introduced and popularized several notational conventions through his numerous and widely circulated textbooks. Most notably, he introduced the concept of a function and was the first to write *f*(*x*) to denote the function *f* applied to the argument *x*. He also introduced the modern notation for the trigonometric functions, the letter *e* for the base of the natural logarithm (now also known as Euler's number), the Greek letter Σ for summations and the letter *i* to denote the imaginary unit

欧拉法　差分

$$\begin{cases} u'(t) = f(t,u), & a \le t \le b \\ u(a) = u_0 \end{cases}$$

划分为 n+1个节点：　$\{t_k\},\ k = 1,2,\dots n+1$

步长　h=dt=(b-a)/n

向前差商近似

$$u'_k = u'(t_k) \approx [u_{k+1} - u_k]/h$$
$$\Rightarrow u_{k+1} \approx u_k + h * f(t_k, u_k)$$

显式

向后差商近似

$$u'_{k+1} \approx [u_{k+1} - u_k]/h$$
$$\Rightarrow u_{k+1} \approx u_k + h * f(t_{k+1}, u_{k+1})$$

隐式

欧拉法　差分

$$\begin{cases} u'(t) = f(t,u), & a \le t \le b \\ u(a) = u_0 \end{cases}$$

两步法

$$u'_k \approx [u_{k+1} - u_{k-1}]/(2h)$$
$$\Rightarrow u_{k+1} \approx u_{k-1} + 2h * f(t_k, u_k)$$

显式

u_2需要特别处理

$$u_2 \approx u_1 + hf(t_1, u_1)$$

积分近似

$$u_{k+1} \approx u_k + \int_{t_k}^{t_{k+1}} f(t,u)dt$$

隐式

$$\Rightarrow \quad u_{k+1} \approx u_k + \frac{h}{2}[f(t_k,u_k) + f(t_{k+1},u_{k+1})]$$

欧拉法　差分

$$\begin{cases} u'(t) = f(t,u), & a \le t \le b \\ u(a) = u_0 \end{cases}$$

积分近似：　预测校准

$$\bar{u}_{k+1} \approx u_k + h * f(t_k, u_k)$$

$$u_{k+1} \approx u_k + \frac{h}{2}[f(t_k, u_k) + f(t_{k+1}, \bar{u}_{k+1})]$$

$$= u_k + \frac{h}{2}[f(t_k, u_k) + f(t_{k+1}, u_k + h * f(t_k, u_k))]$$

欧拉法　差分

$$\begin{aligned} u'(t) &= f(t,u), \qquad a \le t \le b \\ u(a) &= u_0 \end{aligned}$$

高阶展开

$$u(t_{k+1}) = u(t_k) + hu'(t_k) + \frac{1}{2}h^2u''(t_k) + \ldots$$

$$= u(t_k) + hf(t_k,u_k) + \frac{1}{2}h^2f'(t_k,u_k) + \ldots$$

例：

$$\frac{dy}{dt} = -y, \quad y(0)=1, \quad t:[0,10]$$

$$y' = -y \quad \Rightarrow \quad y'' = -y' = y$$

$$y_{k+1} = y_k + h * (-y_k) + \frac{h^2}{2}y_k$$

初值问题：欧拉法

An ODE Example: dy/dt=-y, with y(0)=1. x:[0-10]
Explicit Method

```
 Program main
      IMPLICIT NONE
      Real*8 nstep
      Real*8 xfinal, xini
      Real*8 xstep
      Real*8 y(10000)
      Integer i
      character*50 file_name_out
      file_name_out = 'plot.gnu'
      open( unit=16, file=file_name_out,
access="sequential",
     *      form='formatted',
status="unknown" )
 10   format( E15.7, E15.7, E15.7 )
```

```
      nstep=50.
      xfinal=10.
      xini=0.
      y(1)=1.0
      xstep=(xfinal-xini)/nstep
      write(unit=16, fmt=10) 0., y(1), y(1)
      do i=2,nstep+1
      y(i)=y(i-1)+(-1.)*xstep*y(i-1)
      write(unit=16, fmt=10)(i-1.)*xstep, y(i),
y(1)*dexp(-(i-1.)*xstep)
      enddo
      close( unit=16 )

      end
```

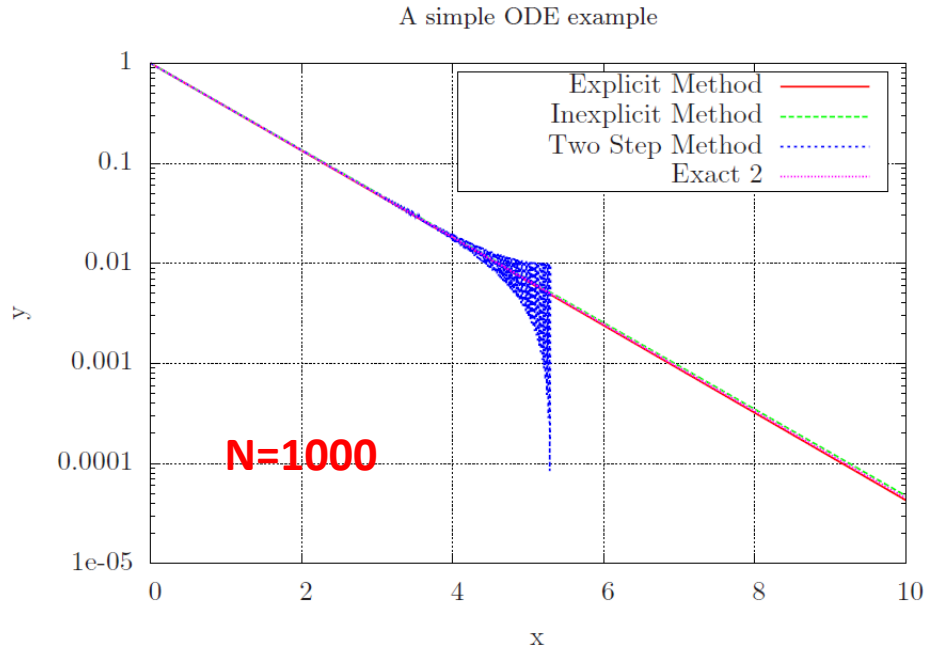
初值问题：欧拉法2，3,4

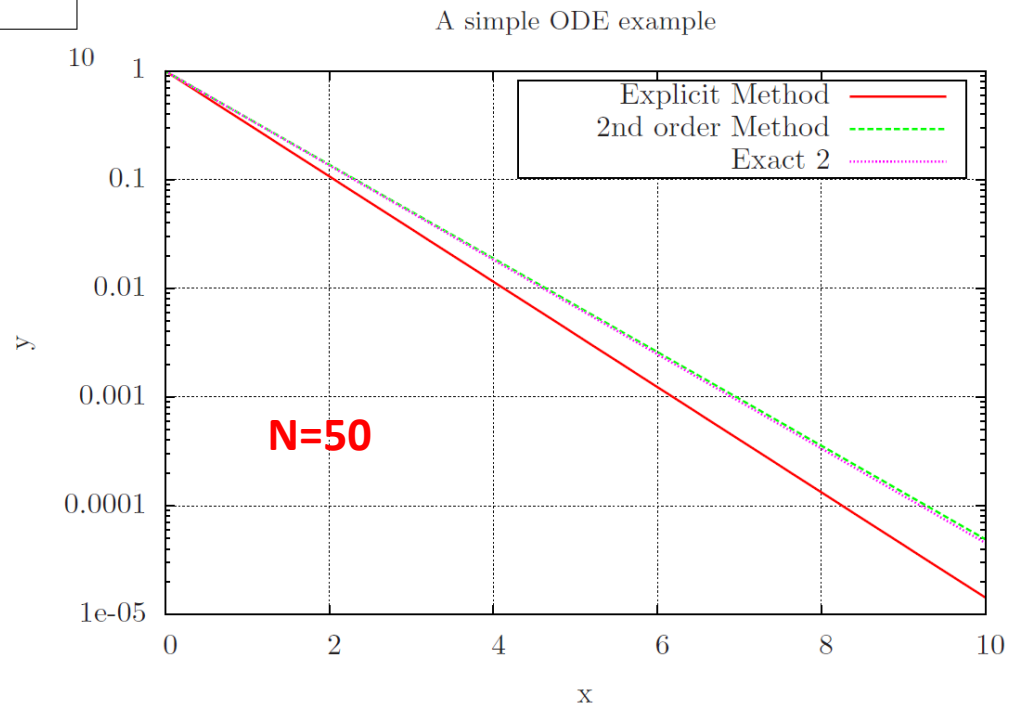
```
do i=2,nstep+1
y(i)=y(i-1)/(1.+xstep)
write(unit=16, fmt=10)(i-1.)*xstep, y(i), y(1)*dexp(-(i-1.)*xstep)
enddo
```

```
y(2)=y(1)+(-1.*y(1))*xstep
write(unit=16, fmt=10) 0.*xstep, y(1), y(1)
write(unit=16, fmt=10) 1.*xstep, y(2), y(2)
do i=2,nstep
y(i+1)=y(i-1)-y(i)*2.0*xstep
write(unit=16, fmt=10) i*xstep, y(i+1), y(1)*dexp(-i*xstep)
enddo
```

```
do i=2,nstep+1
    y(i)=y(i-1)+(-1.)*xstep*y(i-1)+xstep**2/2.0*y(i-1)
    write(unit=16, fmt=10)(i-1.)*xstep, y(i), y(1)*dexp(-(i-1.)*xstep)
enddo
```

初值问题：欧拉法2，3,4


A simple ODE example
Explicit Method
Inexplicit Method
Two Step Method
Exact 2
N=1000
y
x
1
0.1
0.01
0.001
0.0001
1e-05
0
2
4
6
8
10


A simple ODE example
Explicit Method
2nd order Method
Exact 2
N=50
y
x
1
0.1
0.01
0.001
0.0001
1e-05
0
2
4
6
8
10

Runge-Kutta法

Developed around 1900 by the German mathematicians C. Runge and M. W. Kutta.

高阶展开需要计算高阶导数，并不总是很容易。

可用邻近点的函数值近似去表示该点的高阶导数值

30 August 1856 – 3 January 1927

November 3, 1867 – December 25, 1944

$$u_{k+1} = u_k + h\sum_{i=1}^{n} c_i k_i$$

$$where$$

$$k_1 = f(t_k, u_k),$$

$$k_i = f(t_k + a_i h, u_k + h\sum_{j=1}^{i-1} b_{ij} k_j), \qquad i=2,3,\ldots n,$$

$$\sum_{i=1}^{n} c_i = 1,$$

$$\sum_{j=1}^{i-1} b_{ij} = a_i,$$

将近似公式和 泰勒展开 相比较，确定系数。 如果截断误差 $O(h^{p+1})$，则称为 N级p阶的RK方法

二阶Runge-Kutta法

$$u_{k+1} = u_k + h(c_1 k_1 + c_2 k_2)$$

$$where$$
$$k_1 = f(t_k, u_k),$$
$$k_2 = f(t_k + ah, u_k + hbk_1),$$

$$c_1 + c_2 = 1,$$
$$a = b$$

$$u_{k+1} = u_k + c_1 hf(t_k, u_k) + c_2 hf(t_k + ah, u_k + hbk_1)$$

$$f(t_k + ah, u_k + hbk_1) = f(t_k + ah, u_k + hbf(t_k, u_k))$$
$$\approx f(t_k, u_k) + f'_t(t_k, u_k)\, ah + f'_u(t_k, u_k)\, hbf(t_k, u_k)$$

$$u_{k+1} = u_k + (c_1 + c_2) hf(t_k, u_k) + ac_2 h^2 f'_t(t_k, u_k) + bc_2 h^2 f'_u(t_k, u_k) f(t_k, u_k)$$

另一方面， u_{k+1}在t_k的泰勒展开：

$$u_{k+1} = u_k + hf(t_k, u_k) + \frac{h^2}{2} f'(t_k, u_k)$$
$$= u_k + hf(t_k, u_k) + \frac{h^2}{2}(f'_t(t_k, u_k) + f'_u(t_k, u_k) f(t_k, u_k))$$

$$c_1 + c_2 = 1,$$
$$a\, c_2 = b c_2 = 1/2$$

二阶Runge-Kutta法

$$u_{k+1} = u_k + h(c_1 k_1 + c_2 k_2)$$

where

$$k_1 = f(t_k, u_k),$$
$$k_2 = f(t_k + ah, u_k + hbk_1),$$

$$c_1 + c_2 = 1,$$
$$a\,c_2 = b c_2 = 1/2$$

让我们选取c_1=0， $c_2 = 1$，
$\mathbf{a} = b = 1/2$

$$u_{k+1} = u_k + hk_2$$
$$k_1 = f(t_k, u_k)$$
$$k_2 = f(t_k + h/2, u_k + hk_1/2)$$

中点平均

让我们选取c_1=1/2， $c_2 = 1/2$，
$\mathbf{a} = b = 1$

$$u_{k+1} = u_k + h/2(k_1 + k_2)$$
$$k_1 = f(t_k, u_k)$$
$$k_2 = f(t_k + h, u_k + hk_1)$$

函数值平均

四级四阶Runge-Kutta法

$$u_{k+1} = u_k + h\sum_{i=1}^{n} c_i k_i$$

where

$$k_1 = f(t_k, u_k),$$

$$k_i = f\left(t_k + a_i h, u_k + h\sum_{j=1}^{i-1} b_{ij} k_j\right), \qquad i=2,3,\ldots n,$$

$$\sum_{i=1}^{n} c_i = 1,$$

$$\sum_{j=1}^{i-1} b_{ij} = a_i,$$

$$k_1 = f(t_k, u_k)$$
$$k_2 = f(t_k + h/2, u_k + hk_1/2)$$
$$k_3 = f(t_k + h/2, u_k + hk_2/2)$$
$$k_4 = f(t_k + h, u_k + hk_3)$$
$$u_{k+1} = u_k + h/6(k_1 + 2k_2 + 2k_3 + k_4)$$

$c_1 = 1/6,\ \ c_2 = 1/3,\ \ c_3 = 1/3,\ \ c_4 = 1/6,$
$a_2 = 1/2,\ \ a_3 = 1/2,\ \ a_4 = 1,$
$b_{21} = 1/2$
$b_{31} = 0,\ b_{32} = 1/2$
$b_{41} = 0,\ b_{42} = 0,\ b_{43} = 1$

Runge-Kutta法 例1

An ODE Example: dy/dt=-y, with y(0)=1. x:[0-10]

```
      Program main
      IMPLICIT NONE
      Real*8 nstep
      Real*8 xfinal, xini
      Real*8 xstep
      Real*8 y(10000)
      Real*8 k1,k2,k3,k4

      Integer i
      character*50 file_name_out
      file_name_out = 'plot-rk.gnu'

      open( unit=16, file=file_name_out,
access="sequential",
     *      form='formatted', status="unknown" )
 10   format( E15.7, E15.7, E15.7 )
```

```
      nstep=20.
      xfinal=10.
      xini=0.
      y(1)=1.0
      xstep=(xfinal-xini)/nstep
      write(unit=16, fmt=10) 0., y(1), y(1)
      do i=2,nstep+1
      k1=-y(i-1)
      k2=-y(i-1)-xstep/2.0*k1
      k3=-y(i-1)-xstep/2.0*k2
      k4=-y(i-1)-xstep*k3
      y(i)=y(i-1)+xstep*(k1+2.0*k2+2.0*k3+k4)/6.0
      write(unit=16, fmt=10)(i-1.)*xstep, y(i),
y(1)*dexp(-(i-1.)*xstep)
      enddo
      close( unit=16 )
      end
```

Runge-Kutta法 例1

An ODE Example: dy/dt=-y,
with y(0)=1. x:[0-10]

```
#include <fstream>
#include <ostream>
#include <iostream>
#include "TMath.h"
#include "TFile.h"
using namespace std;

float fa(float m, float n)
{
  return(-n);
}
```

```
main1b_RK()
{
  char *out= "plot-rk.gnu";
  FILE *fp2 = fopen(out,"w");
  float nstep=20., xfinal=10., xini=0., y1=1.0;
  float k, k1, k2, k3, k4;
  float xstep=(xfinal-xini)/nstep;
  float r=xini, a=y1;
  unsigned i;
  for(i=0;i<nstep+1;i++)
  {
    fprintf(fp2,"%15.7f %15.7f %15.7f \n", r, a,
y1*exp(-float(i)*xstep));
    k1=fa(r,a);
    k2=fa(r+xstep/2.,a+xstep*k1/2.);
    k3=fa(r+xstep/2.,a+xstep*k2/2.);
    k4=fa(r+xstep,a+xstep*k3);
    k=xstep*(k1+2*k2+2*k3+k4)/6.0;
    r=r+xstep; a=a+k;
  }
}
```

Runge-Kutta法 例2

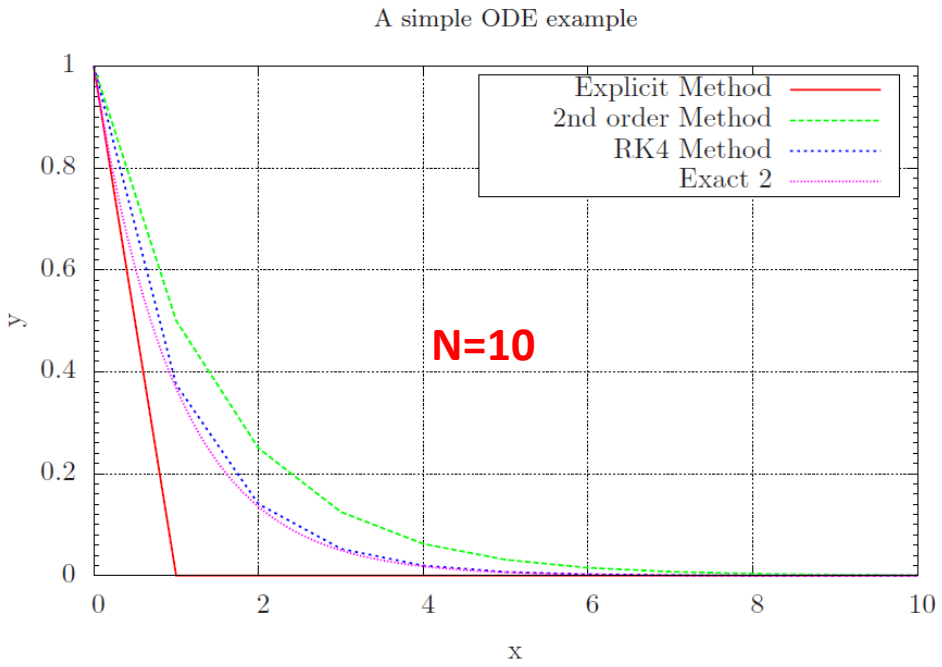
dy/dx=x*sqrt(y),
with y(2)=4
x: [2-3]

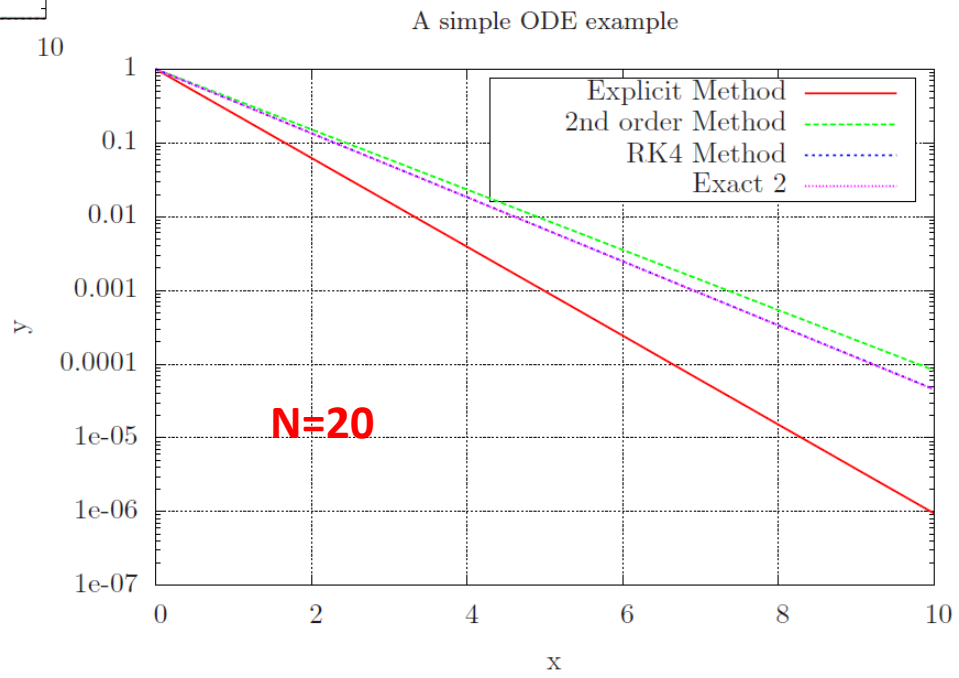
```
#include <fstream>
#include <ostream>
#include <iostream>
#include "TMath.h"
#include "TFile.h"
using namespace std;

float fa(float m, float n)
{
  return(m*sqrt(n));
}

main1b_RK()
{
  char *out= "plot-rk.gnu";
  FILE *fp2 = fopen(out,"w");
```

```
 float nstep=20., xfinal=3., xini=2., y1=4.0;
  float k, k1, k2, k3, k4;
  float xstep=(xfinal-xini)/nstep;
  float r=xini, a=y1;
  unsigned i;
  for(i=0;i<nstep+1;i++)
  {
   fprintf(fp2,"%15.7f %15.7f %15.7f \n", r, a,
y1*exp(-float(i)*xstep));
    k1=fa(r,a);
    k2=fa(r+xstep/2.,a+xstep*k1/2.);
    k3=fa(r+xstep/2.,a+xstep*k2/2.);
    k4=fa(r+xstep,a+xstep*k3);
    k=xstep*(k1+2*k2+2*k3+k4)/6.0;
    r=r+xstep; a=a+k;
  }
}
```

Runge-Kutta法 例1

An ODE Example: dy/dt=-y,
with y(0)=1. x:[0-10]





请比较RK2与RK4

常微分方程组

$$\frac{dy_i}{dx} = y'(x) = f_i(x, y_1, y_2, \dots, y_n), \qquad a \le x \le b$$
$$y_i(a) = y_{i0} \qquad \text{i=1,2,...,n}$$

矩阵形式：

$$Y = (y_1, y_2, \dots, y_n)^T$$
$$F = (f_1, f_2, \dots, f_n)^T$$
$$Y(a) = Y_0 = (y_{10}, y_{20}, \dots, y_{n0})^T$$

$$\frac{dY}{dx} = F(x, y), \qquad Y(a) = Y_0$$

可与单变量常微分方程类比，解法也一样

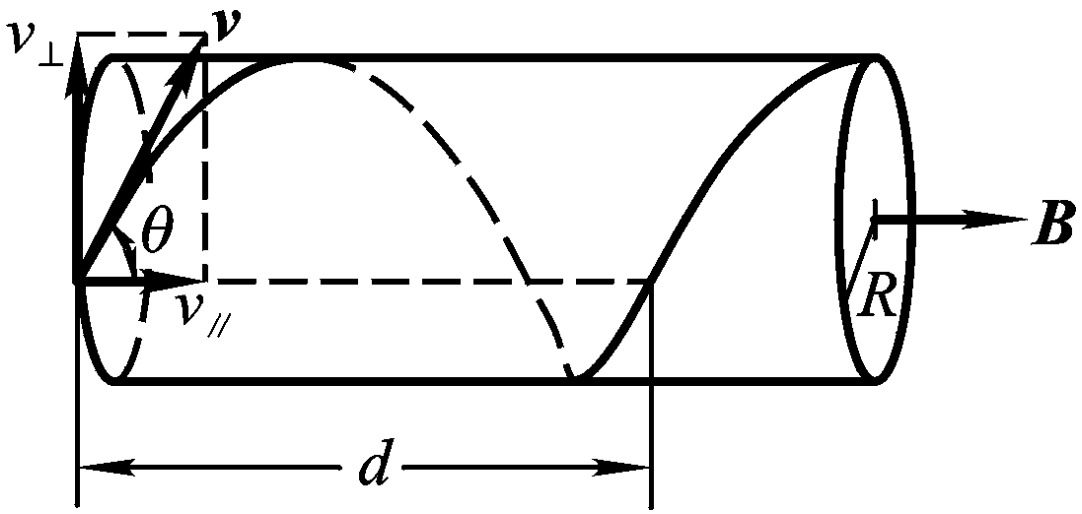
常微分方程组 例

带电粒子在均匀磁场中的运动 $m\frac{d\vec{v}}{dt}=q\vec{v}\times\vec{B}$

设 $y_1=x,\ y_2=y,\ y_3=z,\ y_4=v_x,\ y_5=v_y,\ y_6=v_z$

设磁场方向为z向： $\omega=qB/m$：

$$\frac{dy_1}{dt}=y_4,\quad \frac{dy_2}{dt}=y_5,\quad \frac{dy_3}{dt}=y_6,$$
$$\frac{dy_4}{dt}=\omega y_5,\quad \frac{dy_5}{dt}=-\omega y_4,\quad \frac{dy_6}{dt}=0$$



$$R=\frac{mv\sin\theta}{qB},$$

$$T=\frac{2\pi R}{v}=\frac{2\pi m}{Bq},$$

$$d=v\cos\theta\cdot T$$

常微分方程组 例

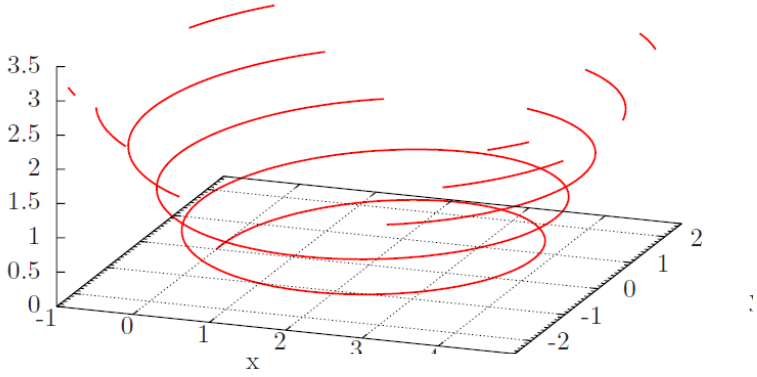
```
    Program main
    IMPLICIT NONE
    Real*8 nstep
    Real*8 xfinal, xini
    Real*8 xstep, wg
    Real*8 x(10000), y(10000), z(10000)
    Real*8 vx(10000), vy(10000), vz(10000)

    Integer i
    character*50 file_name_out
    file_name_out = 'plot.gnu'

    open( unit=16, file=file_name_out, access="sequential",
   *      form='formatted', status="unknown" )
10   format( E15.7, E15.7, E15.7 )

    nstep=10000.
    xfinal=40.
    xini=0.
    wg=1.
    x(1)=0.0
    y(1)=0.0
    z(1)=0.0
    vx(1)=0.0
    vy(1)=2.
    vz(1)=0.1
```

```
xstep=(xfinal-xini)/nstep

    write(unit=16, fmt=10) x(1), y(1), z(1)

    do i=2,nstep+1
    x(i)=x(i-1)+xstep*vx(i-1)
    y(i)=y(i-1)+xstep*vy(i-1)
    z(i)=z(i-1)+xstep*vz(i-1)
    vx(i)=vx(i-1)+wg*xstep*vy(i-1)
    vy(i)=vy(i-1)-wg*xstep*vx(i-1)
    vz(i)=vz(i-1)


    write(unit=16, fmt=10) x(i), y(i), z(i)

    enddo

    close( unit=16 )

    end
```

常微分方程组 例

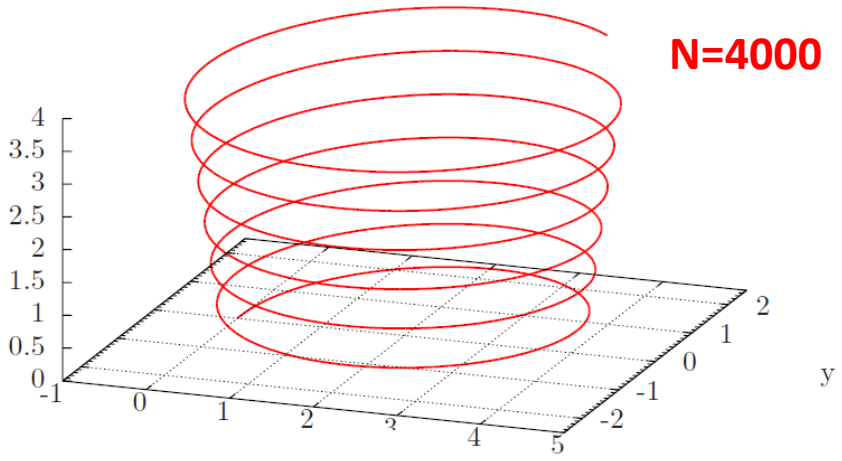
A simple ODE example

N=1000



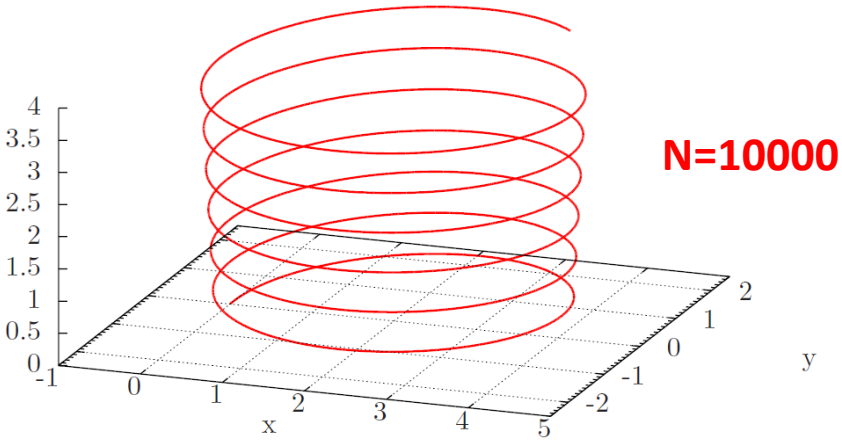
A simple ODE example

N=4000



A simple ODE example

N=10000



请用RK4求解

二阶微分方程

对于高阶微分方程

$y^{(n)} = f\left(x, y, y', \dots, y^{(n-1)}\right)$

$y(x_0) = y_0, \quad y'(x_0) = y'_0, \quad \dots, \quad y^{(n-1)}(x_0) = y^{(n-1)}{}_0$

可以引入新的未知函数

$y_1 = y, \quad y_2 = y', \quad \dots, \quad y_n = y^{(n-1)}$

降阶为n维的一阶微分方程组

$$
\begin{aligned}
y'_1 &= y_2 \\
y'_2 &= y_3 \\
&\cdots\cdots \\
y'_{n-1} &= y_n \\
y'_n &= f(x, y_1, y_2, \dots, y_n)
\end{aligned}
\qquad
\begin{aligned}
y_1(x_0) &= y_0 \\
y_2(x_0) &= y'_0 \\
&\cdots \\
y_{n-1}(x_0) &= y^{(n-2)}{}_0 \\
y_n(x_0) &= y^{(n-1)}{}_0
\end{aligned}
$$

二阶微分方程 例1

y''-2y'=-e^x+1,
with y(0)=1, y'(0)=1/2 x:[0-10]
solution: y=e^x-x/2

降阶法
(dy)'=2*dy-e^x+1
y'=dy

```
      Program main
      IMPLICIT NONE
      Real*8 nstep
      Real*8 xfinal, xini
      Real*8 xstep
      Real*8 y(20000), dy(20000)
      Integer i
      character*50 file_name_out
      file_name_out = 'plot.gnu'

      open( unit=16, file=file_name_out,
access="sequential",
     *      form='formatted', status="unknown" )
 10   format( E15.7, E15.7, E15.7 )
```

```
      nstep=20000.
      xfinal=10.
      xini=0.
      y(1)=1.0
      dy(1)=0.5
      xstep=(xfinal-xini)/nstep

      write(unit=16, fmt=10) 0., y(1), 1.0
      do i=2,nstep+1
      dy(i)=dy(i-1)+xstep*(2.*dy(i-1)-dexp((i-
2.)*xstep)+1.0)
      y(i)=y(i-1)+xstep*dy(i-1)
      write(unit=16, fmt=10)(i-1.)*xstep, y(i),
dexp((i-1.)*xstep)
     &                -((i-1.)*xstep)/2.0
      enddo

      close( unit=16 )
      end
```

二阶微分方程 例1

二阶差分

y''(i)=(y(i+1)-2*y(i)+y(i-1))/xstep**2

y'(i)=(y(i+1)-y(i-1))/2/xstep

```
    Program main
    IMPLICIT NONE
    Real*8 nstep
    Real*8 xfinal, xini
    Real*8 xstep
    Real*8 y(50000)
    Integer i
    character*50 file_name_out
    file_name_out = 'plot2.gnu'

    open( unit=16, file=file_name_out,
access="sequential",
   *      form='formatted', status="unknown" )
 10   format( E15.7, E15.7, E15.7 )

    nstep=20000.
    xfinal=10.
    xini=0.
    y(1)=1.0
    xstep=(xfinal-xini)/nstep
    y(2)=y(1)+0.5*xstep
```

```
    write(unit=16, fmt=10) 0.*xstep, y(1), 1.0
    write(unit=16, fmt=10) 1.*xstep, y(2),
dexp(xstep)-xstep/2.0

    do i=2,nstep
    y(i+1)=2.*y(i)-(xstep+1.)*y(i-1)+(1.-dexp((i-
1)*xstep))*xstep**2
    y(i+1)=y(i+1)/(1.-xstep)
    write(unit=16, fmt=10)  i*xstep, y(i+1),
dexp(i*xstep)
   &                -(i*xstep)/2.0
    enddo

    close( unit=16 )
    end
```

二阶微分方程 例1

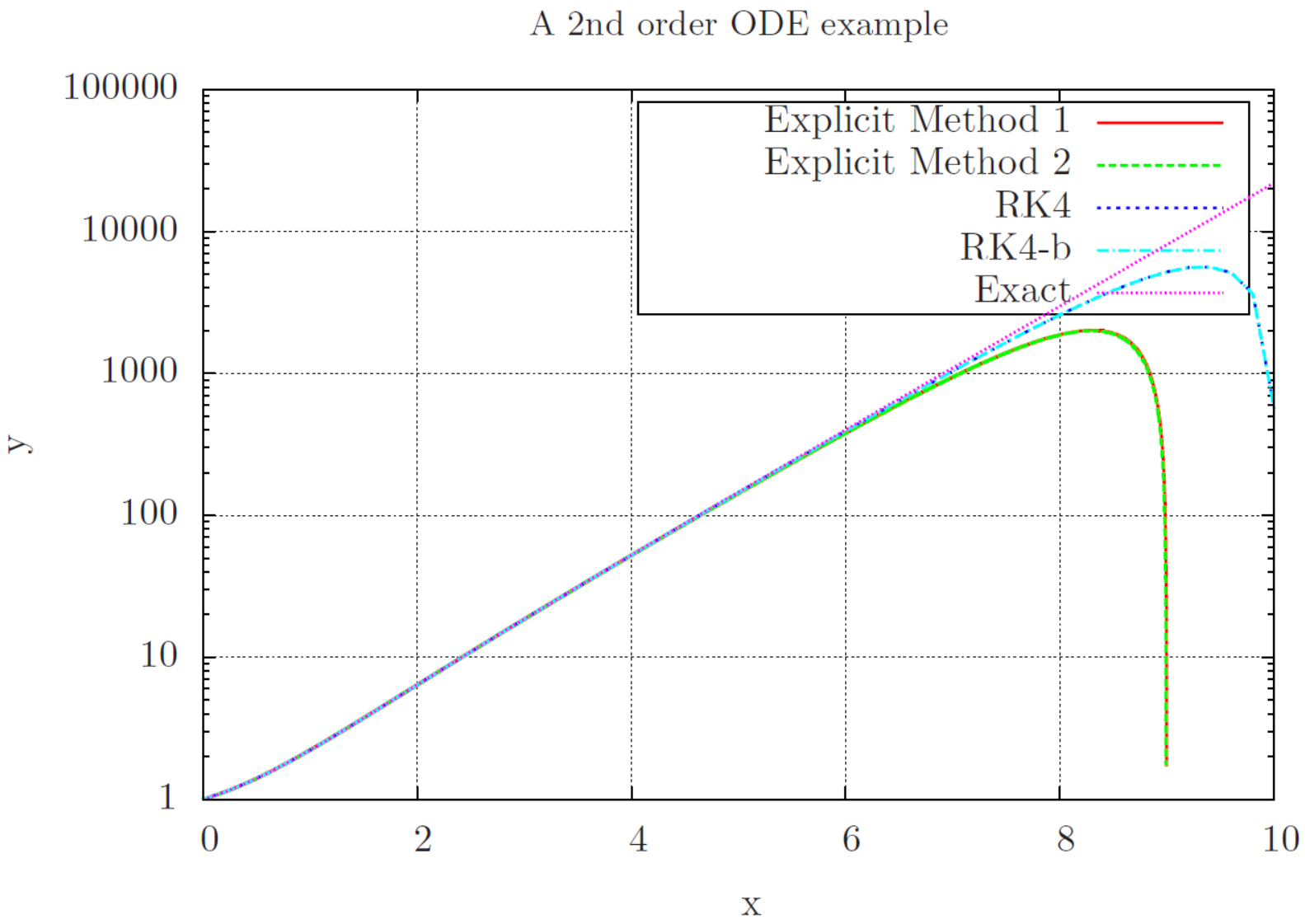
```
#include <fstream>
#include <ostream>
#include <iostream>
#include "TMath.h"
#include "TFile.h"
using namespace std;

float fa(float m, float n1, float n0)
{
  return(n1);
}
float fb(float m, float n1, float n0)
{
  return(2.*n1-exp(m)+1);
}

main5_RK2()
{
  char *out= "plot-rk.gnu";
  FILE *fp2 = fopen(out,"w");
```

RK4

```
 float nstep=50., xfinal=10., xini=0., y11=1.0, y21=0.5;
  float k, k1, k2, k3, k4;
  float l, l1, l2, l3, l4;
  float xstep=(xfinal-xini)/nstep;
  float r=xini, a=y11, b=y21;
  unsigned i;
  for(i=0;i<nstep+1;i++)
  {
    fprintf(fp2,"%15.7f %15.7f %15.7f \n", r, a, b);
    k1=fa(r,b,a);
    l1=fb(r,b,a);
    k2=fa(r+xstep/2., b+xstep*l1/2., a+xstep*k1/2.);
    l2=fb(r+xstep/2., b+xstep*l1/2., a+xstep*k1/2.);
    k3=fa(r+xstep/2., b+xstep*l2/2., a+xstep*k2/2.);
    l3=fb(r+xstep/2., b+xstep*l2/2., a+xstep*k2/2.);
    k4=fa(r+xstep, b+xstep*l3, a+xstep*k3);
    l4=fb(r+xstep, b+xstep*l3, a+xstep*k3);
    k=xstep*(k1+2*k2+2*k3+k4)/6.0;
    l=xstep*(l1+2*l2+2*l3+l4)/6.0;
    r=r+xstep; a=a+k; b=b+l;
  }
}
```

二阶微分方程 例1

RK： N=50
Euler： N=20000



二阶微分方程 例2

A 2nd order ODE Example:

y''-2y'+2y=exp(2x)sin(x)

with y(0)=-0.4, y'(0)=-0.6, x:[0-1]

RK4 Method

y1'=y2, y1(0)=-0.4

y2'=2y2-2y1+exp(2x)*sin(x), y2(0)=-0.6

```
#include <fstream>
#include <ostream>
#include <iostream>
#include "TMath.h"
#include "TFile.h"
using namespace std;

float fa(float m, float n1, float n0)
{
  return(n1);
}

float fb(float m, float n1, float n0)
{
  return(2.*n1-2.*n0+exp(2.*m)*sin(m));
}
```

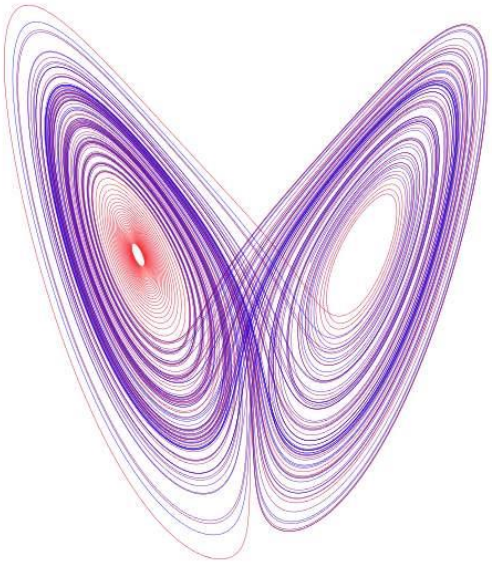
```
main1c_RK()
{
  char *out= "plot-rk.gnu";
  FILE *fp2 = fopen(out,"w");
  float nstep=20., xfinal=1., xini=0., y11=-0.4, y21=-0.6;
  float k, k1, k2, k3, k4;
  float l, l1, l2, l3, l4;
  float xstep=(xfinal-xini)/nstep;
  float r=xini, a=y11, b=y21;
  unsigned i;
  for(i=0;i<nstep+1;i++)
  {
   fprintf(fp2,"%15.7f %15.7f %15.7f \n", r, a, b);
    k1=fa(r,b,a);
    l1=fb(r,b,a);
    k2=fa(r+xstep/2., b+xstep*l1/2., a+xstep*k1/2.);
    l2=fb(r+xstep/2., b+xstep*l1/2., a+xstep*k1/2.);
    k3=fa(r+xstep/2., b+xstep*l2/2., a+xstep*k2/2.);
    l3=fb(r+xstep/2., b+xstep*l2/2., a+xstep*k2/2.);
    k4=fa(r+xstep, b+xstep*l3, a+xstep*k3);
    l4=fb(r+xstep, b+xstep*l3, a+xstep*k3);
    k=xstep*(k1+2*k2+2*k3+k4)/6.0;
    l=xstep*(l1+2*l2+2*l3+l4)/6.0;
    r=r+xstep; a=a+k; b=b+l;
  }
}
```

洛伦兹吸引子

Edward Norton Lorenz，
1917年5月23日—2008年4月16日

1963年美国气象学家爱德华·诺顿·洛伦茨提出**混沌理论**（Chaos），非线性系统具有的多样性和多尺度性。混沌理论解释了决定系统可能产生随机结果。理论的最大的贡献是用简单的模型获得明确的非周期结果。在气象、航空及航天等领域的研究里有重大的作用。

“一只南美洲亚马逊河流域热带雨林中的蝴蝶，偶尔扇动几下翅膀，可以在两周以后引起美国得克萨斯州的一场龙卷风。”



1961年，冬季的一天，洛伦兹在电脑上进行关于天气预报的计算。为了考察一个很长的序列，他走了一条捷径，没有令计算机从头运行，而是从中途开始。他把上次的输出直接打入作为计算的初值，然后他穿过大厅下楼，去喝咖啡。一小时后他回来时，发生了出乎意料的事。 第一次的计算机运算结果，打印只显示到小数点后三位的0．506，而非完整的小数点后六位：0．506127。这个远小于千分之一的差异，造成第二次的仿真结果和第一次完全不同。

洛伦兹吸引子

$$\begin{pmatrix} y'_1 \\ y'_2 \\ y'_3 \end{pmatrix} = \begin{pmatrix} -\beta & 0 & y_2 \\ 0 & -\sigma & \sigma \\ -y_2 & \rho & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

这个微分方程组不是线性的，方程中y_1与大气对流相关，另外两个分量分别与温度的竖直和水平变化相关。

其中包含三个常参数：σ为普兰特数；ρ是规范化的瑞利数；β和区域的几何形状相关。通常可取σ=10，ρ=28，β=5/3。这些值与地球大气相关。

系统中y_2的引入极大地改变了系统的性质。方程没有随机因数，但解却由上面的参数和初始条件决定，并很难预测解的特性。

当取某些参数值时，三维空间的轨迹是著名的奇异吸引子，有界、无周期、不收敛，也不自交。轨道混乱地围绕两个不动点，即吸引子。
对某些参数，解可能收敛于一个固定点，分叉到无穷或有周期性的摆动。

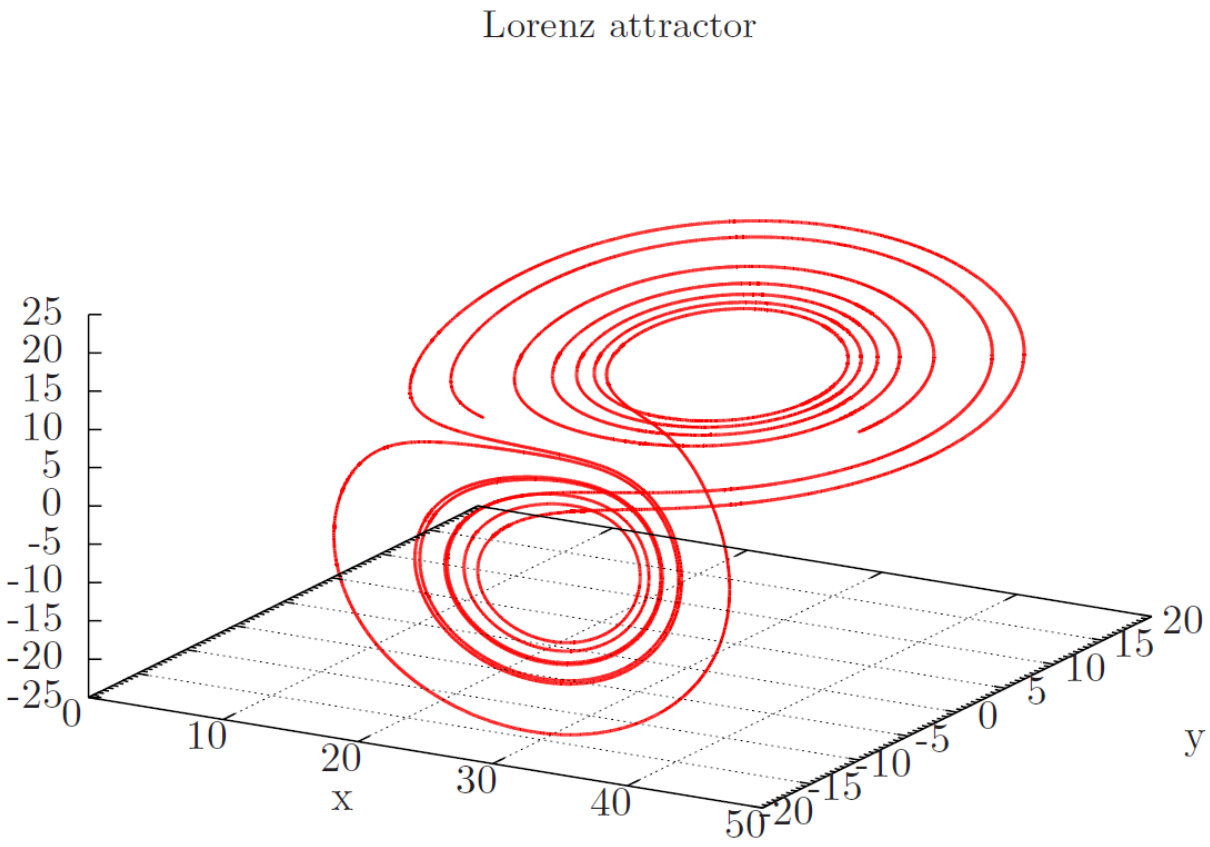
洛伦兹吸引子

```
c  y1'=-b*y1+0*y2+y2*y3
c  y2'=0*y1-s*y2+s*y3
c  y3'=-y2*y1+r*y2-y3
c  usually take s=10., r=28., b=8./3.
```

```
      Program main
      IMPLICIT NONE
      Real*8 nstep
      Real*8 xfinal, xini
      Real*8 xstep, wg
      Real*8 x(100000), y(100000), z(100000)
      Real*8 beta, sigma, rho
      Integer i
      character*50 file_name_out
      file_name_out = 'plot.gnu'
      open( unit=16, file=file_name_out,
access="sequential",
     *      form='formatted', status="unknown" )
 10   format( E15.7, E15.7, E15.7 )
      nstep=100000.
      xfinal=10.
      xini=0.
      beta=8./3.
      rho=28.
      sigma=10.
      x(1)=12.0
      y(1)=4.0
      z(1)=0.0
```

```
      xstep=(xfinal-xini)/nstep
      write(unit=16, fmt=10) x(1), y(1), z(1)
       do i=2,nstep+1
      x(i)=x(i-1)+xstep*(-beta*x(i-1)+y(i-1)*z(i-
1))
      y(i)=y(i-1)+xstep*(-sigma*y(i-
1)+sigma*z(i-1))
      z(i)=z(i-1)+xstep*(-y(i-1)*x(i-1)+rho*y(i-1)-
z(i-1))
      write(unit=16, fmt=10) x(i), y(i), z(i)
      enddo

      close( unit=16 )
      end
```

洛伦兹吸引子

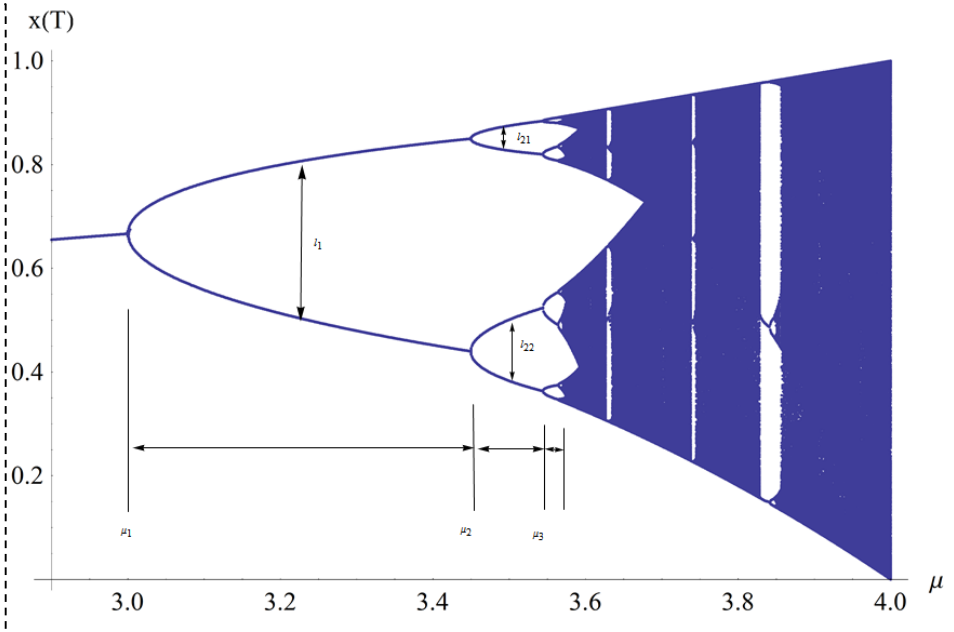


调整不同参数

混沌,分形

**Chaos: When the present determines the future,
but the approximate present does not approximately determine the future.**

Chaos theory is a branch of mathematics focused on the behavior of dynamical systems that are highly sensitive to initial conditions. 'Chaos' is an interdisciplinary theory stating that within the apparent randomness of chaotic complex systems, there are underlying patterns, constant feedback loops, repetition, self-similarity, fractals, self-organization, and reliance on programming at the initial point known as sensitive dependence on initial conditions.



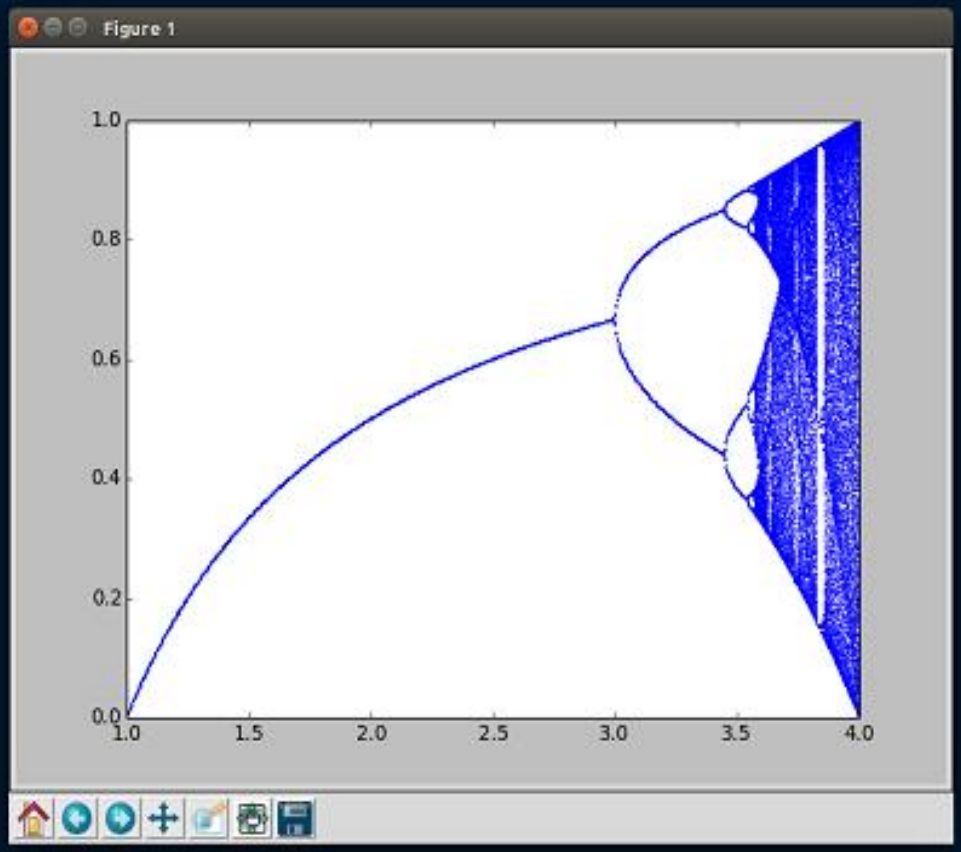
Logistic映射是研究动力系统、混沌、分形等复杂系统行为的一个经典模型。Logistic映射又叫Logistic迭代，其实就是一个时间离散的动力系统，即按照如下方程进行反复迭代：

$$x(t+1) = \mu x(t)(1 - x(t))$$

其中，t为迭代时间步，对于任意的t，$x(t) \in [0,1]$，μ为一可调参数，为了保证映射得到的$x(t)$始终位于[0,1]内，则$\mu \in [0,4]$。当变化不同的参数μ的时候，该方程会展现出不同的动力学极限行为（即当t趋于无穷大，x(t)的变化情况），包括：稳定点（即最终x(t)始终为同一个数值）、周期（x(t)会在2个或者多个数值之间跳跃，以及混沌：x(t)的终态不会重复，而会等概率地取遍某区间）。

https://github.com/StewartDouglas/

```
# u is the dependent variable here
u = 0.0
# Small perturbation in u
u = u + eps
# r is plotted on the x-axis. Different
# values of r produce very different behaviour.
# In population dynamics r represents the
# combined rate for reproduction and starvation
r = 0.0
dr = 0.005
# Stored the x-axis values in r_list,
# and the y-axis values in u_list
r_list = []
u_list = []
while r < 4.0:
    # Throw away the 1st 200 iterations so that we
    # converge to the equilibria
    for i in range(1,200):
        u = r*u*(1.0 - u)
    for i in range(1,500):
        u = r*u*(1.0 - u)
        r_list.append(r)
        u_list.append(u)
    r = r+dr
    u = u+eps
# Plot the output
plt.pyplot.scatter(r_list,u_list, s=1, facecolor='blue', lw = 0)
plt.pyplot.axis([1.0,4.0,0,1.0])
plt.pyplot.show()
```



http://wiki.swarma.net/index.php/Logistic%E6%98%A0%E5%B0%84

当迭代方程的极限行为出现两个周期点的时候，我们不妨设这两个点分别为x_1, x_2，则从x_1开始一步迭代应该能得到x_2，而从x_2开始迭代又得到x_1，周而复始，因此，我们可以列出方程：

$$\begin{cases} x_2 = f(x_1), \\ x_1 = f(x_2). \end{cases}$$

其中，我们定义$f(x) = \mu x(1-x)$。如果将$x_2 = f(x_1)$代入第二个方程，那么上述方程组相当于求解：

$$x^* = f(f(x^*)) = f^{(2)}(x^*) = \mu^2 x^* - \mu^2 (x^*)^2 - \mu^3 (x^*)^2 + 2\mu^3 (x^*)^3 - \mu^3 (x^*)^4$$

显然，这是一个一元四次方程，我们可以求解该方程得到四个解：

$$x_1^* = \frac{1}{2}(1 + \frac{1}{\mu} + \frac{\sqrt{-3-2\mu+\mu^2}}{\mu}), x_2^* = \frac{1}{2}(1 + \frac{1}{\mu} - \frac{\sqrt{-3-2\mu+\mu^2}}{\mu}), x_3^* = 0, x_4^* = \frac{\mu - 1}{\mu}$$

其中第3个是和第4个是不动点。我们知道，如果x*是迭代方程的不动点，那么，x*一定也是迭代方程的2周期点、4周期点、任意的周期点。所以，后两个解并非我们要求的二周期点。而第1和第2个解才是真正的二分周期点。我们可以验证：
$x_1^* = f(x_2^*), x_2^* = f(x_1^*)$。

为了让这2个解有意义，μ必须使得根号中的式子大于0，也就是$-3 - 2\mu + \mu^2 > 0$，于是可以推得：

$\mu > 3$或者$\mu < -1$

刚性微分方程

在某些物理问题里，可能涉及的研究对象的时间尺度相差很大，例如，在等离子体物理中 电子和离子的标度不同， 这类问题的运动方程通常是刚性微分方程组，如：

$$\begin{cases} x' = 998x + 1998y \\ y' = -999x - 1999y \\ x(0) = 1, y(0) = 0 \end{cases}$$

其解析解为：

$$\begin{cases} x(t) = 2e^{-t} - e^{-1000t} \\ y(t) = -e^{-t} + e^{-1000t} \end{cases}$$

e^{-t}为慢变分量，e^{-1000t}为快变分量。
计算步长由快变分量决定；计算步数由两者共同确定。
需要用很小的步长，计算很长的区间。

虽然从理论上来说，快变分量很快趋于0，影响范围很小，
但是在数值方法中，快变分量的存在对于求解带来了一定困难。

刚性微分方程

一般地，将刚性微分方程组定义为常系数线性微分方程组：

$$\frac{dY}{dt} = AY + B$$

如果系数矩阵A的特征值得实部$\mathrm{Re}(\lambda_i) < 0, \quad i = 1,2,\ldots,n$

且

$$s = \frac{\max|\mathrm{Re}(\lambda_i)|}{\min|\mathrm{Re}(\lambda_i)|} \gg 1$$

则称该方程组为刚性微分方程组，称s为刚性比。

一般s>10就可以认为微分方程组是刚性的。

求解的数值方法一般为高阶单步和线性多步法。隐式线性多步法中的**Gear法**稳定性较好：

$$y_{n+k} = \sum_{j=0}^{k-1} a_j y_{n+j} + h\beta_k f_{n+k}$$

其中 $\beta_k \neq 0$ 。选择适当的参数值，k步Gear方法可以是k阶收敛的。

下面分别给出**二阶和三阶Gear公式：**

$$y_{n+2} = (4y_{n+1} - y_n + 2hf_{n+2})/3,$$
$$y_{n+3} = (18y_{n+2} - 9y_{n+1} + 2y_n + 6hf_{n+3})/11$$

刚性微分方程

$$y_{n+2} = (4y_{n+1} - y_n + 2hf_{n+2})/3,$$

$$4y_{n+1} = 4y_n + 4hf_n + 2h^2 f'_n$$
$$2hf_{n+2} = 2h(f_n + 2hf'_n) = 2hf_n + 4h^2 f'_n$$

$$4y_{n+1} - y_n + 2hf_{n+2} = 3y_n + 6hf_n + 6h^2 f'_n$$

$$3y_{n+2} = 3y_n + 6hf_n + 6h^2 f'_n$$

刚性微分方程

```
// A Stiff ODE Example
// RK4 Method
// y'=-999x-1999y
// x'=998x+1998y
// x(0)=1, y(0)=0
// Exact Solution: y(t)=-exp(-t)+exp(-1000t)
// Exact Solution: x(t)=2exp(-t)-exp(-1000t)
```

```
// Strong dependence on nstep!!!
#include <fstream>
#include <ostream>
#include <iostream>
#include "TMath.h"
#include "TFile.h"
using namespace std;

float fa(float m, float n1, float n0)
{
  return(998.*n0+1998.*n1);
}
float fb(float m, float n1, float n0)
{
  return(-999.*n0-1999.*n1);
}
main8_RK()
{
  char *out= "plot-rk.gnu";
  FILE *fp2 = fopen(out,"w");
  float nstep=200., xfinal=0.02, xini=0., y11=1.0,
y21=0.0;
  float k, k1, k2, k3, k4;
  float l, l1, l2, l3, l4;
```

```
float xstep=(xfinal-xini)/nstep;
 float r=xini, a=y11, b=y21;
 unsigned i;
 for(i=0;i<nstep+1;i++)
 {
  fprintf(fp2,"%15.7f %15.7f %15.7f \n", r, a, b);
  k1=fa(r,b,a);
  l1=fb(r,b,a);
  k2=fa(r+xstep/2., b+xstep*l1/2., a+xstep*k1/2.);
  l2=fb(r+xstep/2., b+xstep*l1/2., a+xstep*k1/2.);
  k3=fa(r+xstep/2., b+xstep*l2/2., a+xstep*k2/2.);
  l3=fb(r+xstep/2., b+xstep*l2/2., a+xstep*k2/2.);
  k4=fa(r+xstep, b+xstep*l3, a+xstep*k3);
  l4=fb(r+xstep, b+xstep*l3, a+xstep*k3);
  k=xstep*(k1+2*k2+2*k3+k4)/6.0;
  l=xstep*(l1+2*l2+2*l3+l4)/6.0;
  r=r+xstep; a=a+k; b=b+l;
 }
}
```

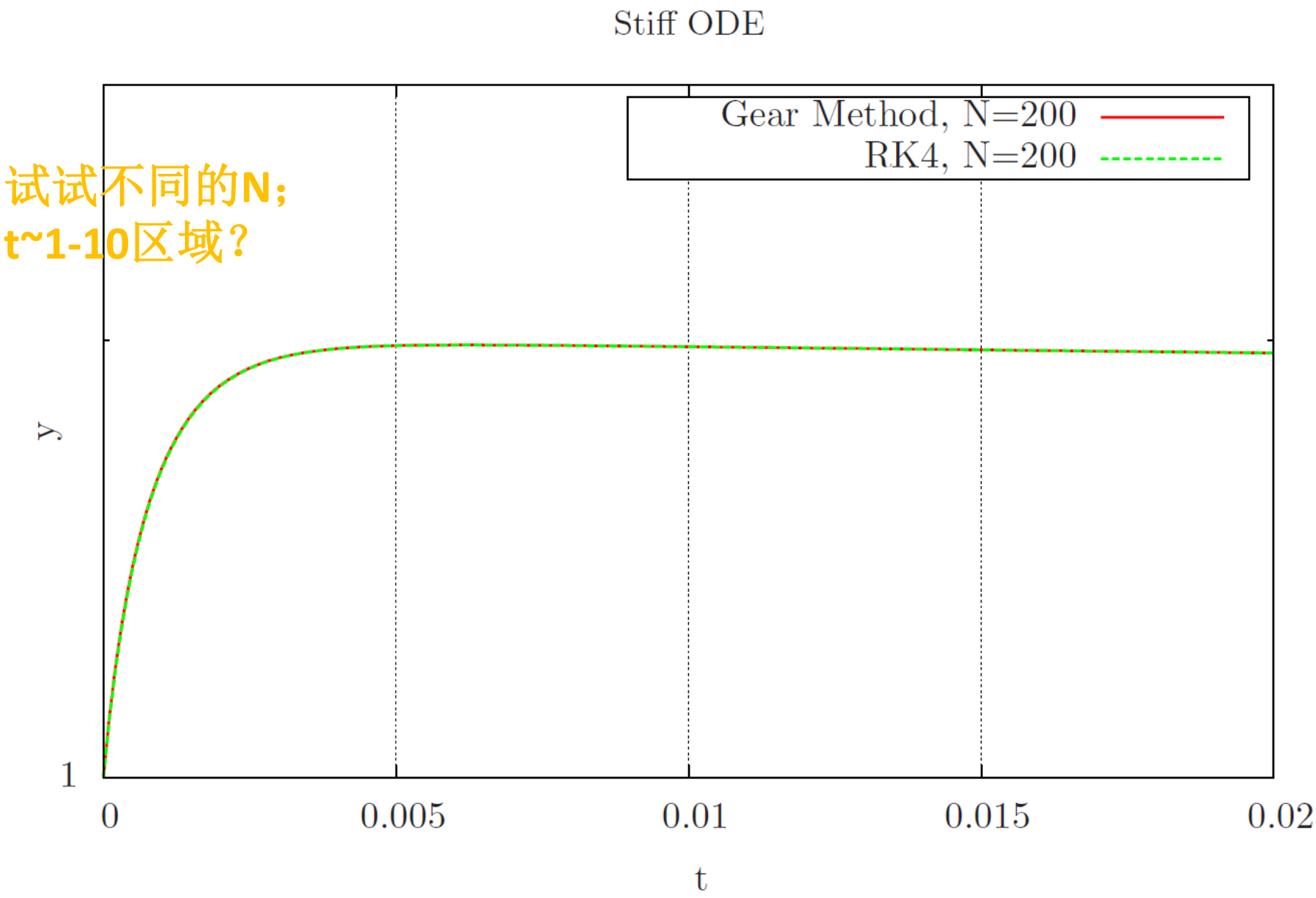
刚性微分方程

```
// A Stiff ODE Example
// Gear Method
// y'=-999x-1999y
// x'=998x+1998y
// x(0)=1, y(0)=0
// Exact Solution: y(t)=-exp(-t)+exp(-1000t)
// Exact Solution: x(t)=2exp(-t)-exp(-1000t)
```

```
      Program main
      IMPLICIT NONE
      Real*8 nstep
      Real*8 xfinal, xini
      Real*8 xstep
      Real*8 x(10000), y(10000)
      Real*8 a,b,c,d,e,f;
      Integer i
      character*50 file_name_out
      file_name_out = 'plot.gnu'
      open( unit=16, file=file_name_out,
access="sequential",
     *      form='formatted', status="unknown" )
 10   format( E15.7, E15.7, E15.7 )
      nstep=5000.
      xfinal=0.02
      xini=0.
      x(1)=1.0
      y(1)=0.0
      xstep=(xfinal-xini)/nstep

      x(2)=x(1)+xstep*(998.*x(1)+1998.*y(1))
      y(2)=y(1)+xstep*(-999.*x(1)-1999.*y(1))
```

```
      write(unit=16, fmt=10) xini+(1.-1.)*xstep, x(1),
y(1)
      write(unit=16, fmt=10) xini+(2.-1.)*xstep, x(2),
y(2)
      do i=3,nstep+1
       a=3.+2.*1999.*xstep
       b=2.*999.*xstep
       c=4.*y(i-1)-y(i-2)
       d=3.-2.*998.*xstep
       e=-2.*1998.*xstep
       f=4.*x(i-1)-x(i-2)
       x(i)=(c*e-a*f)/(b*e-a*d)
       y(i)=(c*d-b*f)/(a*d-b*e)
      write(unit=16, fmt=10) xini+(i-1.)*xstep, x(i),
y(i)
      enddo
      close( unit=16 )
      end
```

刚性微分方程



试试不同的N；
t~1-10区域？

边值问题：介绍

两点边值问题的一般形式为：

$$y''(x) = f(x, y, y')$$
$$\begin{cases} a_0 y(a) + b_0 y'(a) = r_0 \\ a_1 y(b) + b_1 y'(b) = r_1 \end{cases}$$

第一类边界条件 $b_0 = b_1 = 1$
第二类边界条件 $a_0 = a_1 = 1$
其他为第三类混合边界条件

差分法，打靶法

$$y''(x_i) = (y_{i+1} - 2y_i + y_{i-1})/h^2$$
$$y'(x_i) = (y_{i+1} - y_{i-1})/(2h)$$

边值问题：差分法

简单二阶微分方程

$$\begin{cases} u(x)y'' + v(x)y' + w(x)y = f(x) \\ y(a) = \alpha \\ y(b) = \beta \end{cases}$$

代入

$$\begin{aligned} y''(x_i) &= (y_{i+1} - 2y_i + y_{i-1})/h^2 \\ y'(x_i) &= (y_{i+1} - y_{i-1})/(2h) \end{aligned}$$

$$\Rightarrow$$

$$\begin{cases} a_i y_{i-1} + b_i y_i + c_i y_{i+1} = d_i, & i = 2,3\ldots,n-1 \\ y_1 = \alpha, \\ y_n = \beta \end{cases}$$

其中

$$\begin{aligned} a_i &= u(x_i) - hv(x_i)/2, \\ b_i &= h^2 w(x_i) - 2u(x_i), \\ c_i &= u(x_i) + hv(x_i)/2, \\ d_i &= h^2 f(x_i) \end{aligned}$$

$$\Downarrow$$

$$\begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \cdots & \cdots & \cdots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \cdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

其中

$$b_1 = 1,\ c_1 = 0,\ d_1 = \alpha,$$
$$a_{n=0},\ b_n = 1,\ d_n = \beta$$

三对角矩阵方程组： 追赶法

边值问题：1维泊松方程

静电场方程的微分形式

$$\begin{cases} \nabla \cdot \vec{E} = \dfrac{\rho}{\varepsilon_0}, \\ \nabla \times \vec{E} = 0. \end{cases}$$

将$\vec{E} = -\nabla U$代入$\nabla \cdot \vec{E} = \dfrac{\rho}{\varepsilon_0}$,

即可得 $\nabla \cdot \vec{E} = -\nabla \cdot (\nabla U) = -\nabla^2 U = \dfrac{\rho}{\varepsilon_0}$

$$\nabla^2 U = -\frac{\rho}{\varepsilon_0}.$$

1-D Problem with Dirichlet Boundary Conditions

$$\frac{d^2u(x)}{dx^2} = v(x),$$

for $x_l \le x \le x_h$, subject to the Dirichlet boundary conditions $u(x_l) = u_l$ and $u(x_h) = u_h$.

边值问题：1维泊松方程

$$\frac{d^2u(x)}{dx^2} = v(x),$$

for $x_l \le x \le x_h$, subject to the Dirichlet boundary conditions $u(x_l) = u_l$ and $u(x_h) = u_h$.

$$x_i = x_l + \frac{i\,(x_h - x_l)}{N+1},$$

边界x_l和x_h对应0和N+1

$$u_{i-1} - 2\,u_i + u_{i+1} = v_i\,(\delta x)^2,$$

for $i = 1, N$, where $v_i \equiv v(x_i)$. Furthermore, $u_0 = u_l$ and $u_{N+1} = u_h$.

Let $\mathbf{u} = (u_1, u_2, \cdots, u_N)$ be the vector of the u-values, and let

$$\mathbf{w} = [v_1\,(\delta x)^2 - u_l,\ v_2\,(\delta x)^2,\ v_3\,(\delta x)^2,\ \cdots,\ v_{N-1}\,(\delta x)^2,\ v_N\,(\delta x)^2 - u_h]$$

边值问题：1维泊松方程

$$\mathbf{M}\,\mathbf{u} = \mathbf{w}.$$

The matrix $\mathbf{M}$ takes the form

$$\mathbf{M} = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix} \qquad \text{for } N = 6.$$

$$\mathbf{u} = \mathbf{M}^{-1}\,\mathbf{w},$$

三对角矩阵方程组： 追赶法

1维泊松方程：混合边值问题

$$\alpha_l\, u(x) + \beta_l \frac{du(x)}{dx} = \gamma_l,$$

at $x = x_l$, and

$$\alpha_h\, u(x) + \beta_h \frac{du(x)}{dx} = \gamma_h,$$

at $x = x_h$. Here, α_l, β_l, *etc.*, are known constants.

$$\alpha_l\, u_0 + \beta_l \frac{u_1 - u_0}{\delta x} = \gamma_l,$$

$$\alpha_h\, u_{N+1} + \beta_h \frac{u_{N+1} - u_N}{\delta x} = \gamma_h,$$

→

$$u_0 - 2u_1 + u_2 = v_1(\delta x)^2$$

$$u_{N-1} - 2u_N + u_{N+1} = v_N(\delta x)^2$$

→

$$b_1 = -2 - \frac{\beta_l}{\alpha_l\, \delta x - \beta_l}, \qquad b_N = -2 + \frac{\beta_h}{\alpha_h\, \delta x + \beta_h},$$

$$w_1 = v_1\, (\delta x)^2 - \frac{\gamma_l\, \delta x}{\alpha_l\, \delta x - \beta_l}, \qquad w_N = v_N\, (\delta x)^2 - \frac{\gamma_h\, \delta x}{\alpha_h\, \delta x + \beta_h}.$$

1维泊松方程实例

$$\frac{d^2u(x)}{dx^2} = v(x), \quad \text{in the region } 0 \le x \le 1, \text{ with } v(x) = 1 - 2x^2.$$

$$\alpha_l\, u(x) + \beta_l \frac{du(x)}{dx} = \gamma_l, \quad \text{at } x = x_l, \text{ and}$$
$$\alpha_h\, u(x) + \beta_h \frac{du(x)}{dx} = \gamma_h, \quad \text{at } x = x_h.$$

$$u(x) = g + h\,x + \frac{x^2}{2} - \frac{x^4}{6},$$

where

$$g = \frac{\gamma_l\,(\alpha_h + \beta_h) - \beta_l\,[\gamma_h - (\alpha_h + \beta_h)/3]}{\alpha_l\,\alpha_h + \alpha_l\,\beta_h - \beta_l\,\alpha_h},$$
$$h = \frac{\alpha_l\,[\gamma_h - (\alpha_h + \beta_h)/3] - \gamma_l\,\alpha_h}{\alpha_l\,\alpha_h + \alpha_l\,\beta_h - \beta_l\,\alpha_h}.$$

1维泊松方程实例

```
// d^2 u / dx^2 = v  for xl <= x <= xh
// alpha_l u + beta_l du/dx = gamma_l at x=xl
// alpha_h u + beta_h du/dx = gamma_h at x=xh
// Arrays u and v assumed to be of extent N+2.
// Now, ith elements of arrays correspond to
// x_i = xl + i * dx  i=0,N+1
// Here, dx = (xh - xl) / (N+1) is grid spacing.
```

$$a_i u_{i-1} + b_i u_i + c_i u_{i+1} = w_i,$$

for $i = 1, N$. Let us search for a solution of the form

$$u_{i+1} = x_i u_i + y_i.$$

$$a_i u_{i-1} + b_i u_i + c_i (x_i u_i + y_i) = w_i,$$

which can be rearranged to give

$$u_i = -\frac{a_i u_{i-1}}{b_i + c_i x_i} + \frac{w_i - c_i y_i}{b_i + c_i x_i}.$$

$$x_{i-1} = -\frac{a_i}{b_i + c_i x_i},$$

$$y_{i-1} = \frac{w_i - c_i y_i}{b_i + c_i x_i}.$$

$$x_{N-1} = -\frac{a_N}{b_N}, \qquad y_{N-1} = \frac{w_N}{b_N},$$

since $c_N = 0$. Furthermore,

$$x_i = -\frac{a_{i+1}}{b_{i+1} + c_{i+1} x_{i+1}}, \qquad y_i = \frac{w_{i+1} - c_{i+1} y_{i+1}}{b_{i+1} + c_{i+1} x_{i+1}}$$

for $i = N - 2, 1$. Finally,

$$x_0 = 0, \qquad y_0 = \frac{w_1 - c_1 y_1}{b_1 + c_1 x_1}, \qquad \text{since } a_1 = 0.$$

$u_1 = y_0$, since $x_0 = 0$, and

$u_{i+1} = x_i u_i + y_i$ for $i = 1, N - 1$.

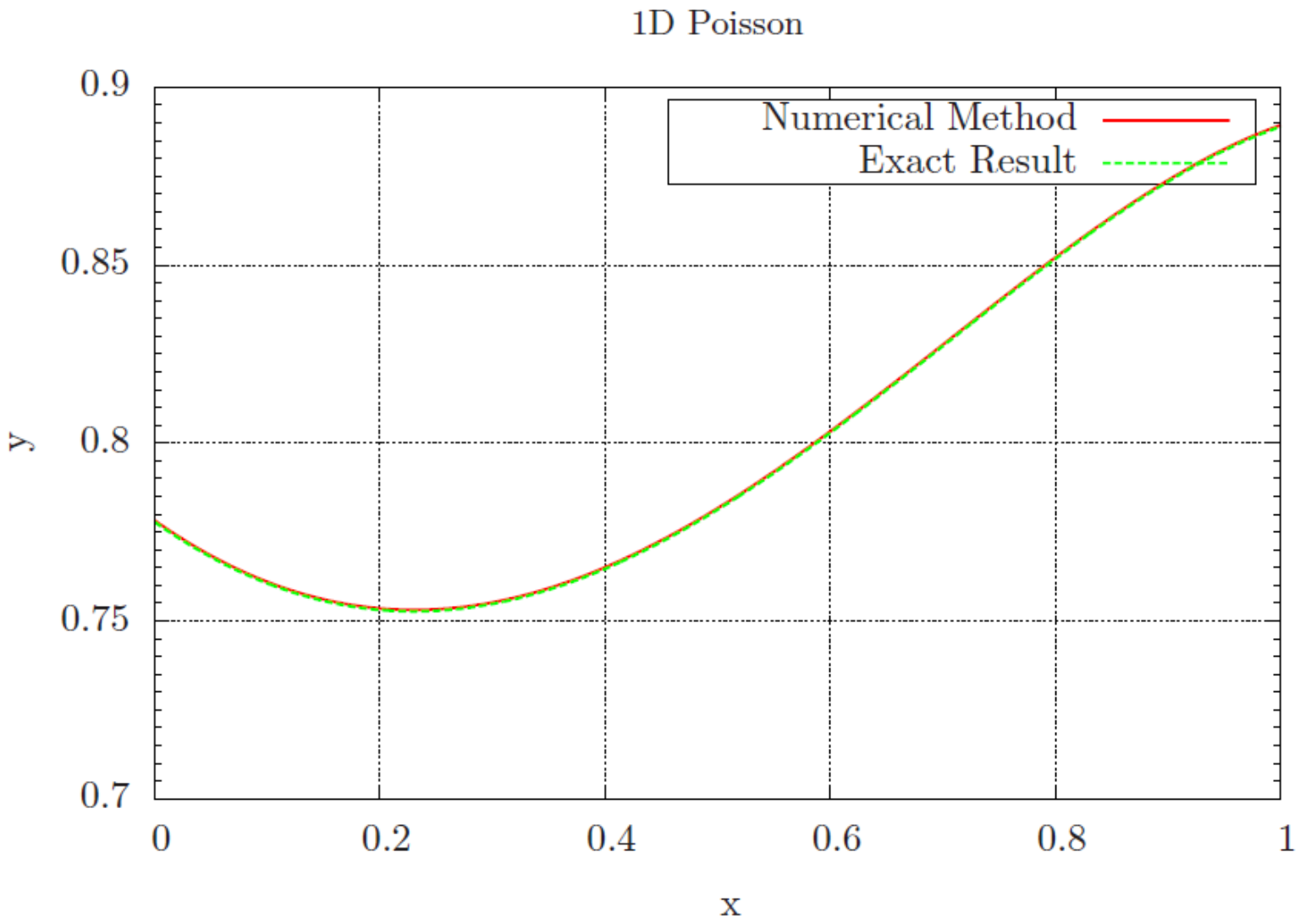
1维泊松方程实例

```
#include <blitz/array.h>
using namespace blitz;

void Tridiagonal (Array<double,1> a, Array<double,1> b, Array<double,1> c,
Array<double,1> w, Array<double,1>& u)
{
  // Find N. Declare local arrays.
  int N = a.extent(0) - 2;
  Array<double,1> x(N), y(N);
  // Scan up diagonal from i = N to 1
  x(N-1) = - a(N) / b(N);
  y(N-1) = w(N) / b(N);
  for (int i = N-2; i > 0; i--)
  {
   x(i) = - a(i+1) / (b(i+1) + c(i+1) * x(i+1));
   y(i) = (w(i+1) - c(i+1) * y(i+1)) / (b(i+1) + c(i+1) * x(i+1));
  }
  x(0) = 0.;
  y(0) = (w(1) - c(1) * y(1)) / (b(1) + c(1) * x(1));
  // Scan down diagonal from i = 0 to N-1
  u(1) = y(0);
  for (int i = 1; i < N; i++)
   u(i+1) = x(i) * u(i) + y(i);
  }
```

1维泊松方程实例

```
int main()
{
  char *out= "plot.gnu";
  FILE *fp2 = fopen(out,"w");
    double alpha_l, beta_l, gamma_l;
    double alpha_h, beta_h, gamma_h;
    double dx=0.001;
    double xin=0.0;
    int N = 1000;
    alpha_l=1.0; beta_l=-1.0; gamma_l=1.0;
    alpha_h=1.0; beta_h=1.0; gamma_h=1.0;

    Array<double,1> u(N+2), v(N+2);
    Array<double,1> a(N+2), b(N+2), c(N+2), w(N+2);

  for (int i = 1; i <= N; i++) v(i) = 1.-
2.*(xin+i*dx)*(xin+i*dx);
  // Initialize tridiagonal matrix
  for (int i = 2; i <= N; i++) a(i) = 1.;
  for (int i = 1; i <= N; i++) b(i) = -2.;
  b(1) -= beta_l / (alpha_l * dx - beta_l);
  b(N) += beta_h / (alpha_h * dx + beta_h);
  for (int i = 1; i <= N-1; i++) c(i) = 1.;
  for (int i = 1; i <= N; i++)
      w(i) = v(i) * dx * dx;
w(1) -= gamma_l * dx / (alpha_l * dx - beta_l);
w(N) -= gamma_h * dx / (alpha_h * dx + beta_h);
Tridiagonal (a, b, c, w, u);
  // Calculate i=0 and i=N+1 values
  u(0) = (gamma_l * dx - beta_l * u(1)) /
        (alpha_l * dx - beta_l);
  u(N+1) = (gamma_h * dx + beta_h * u(N)) /
        (alpha_h * dx + beta_h);

  for (int i=0; i<=N+1;i++) {
     double xg=(gamma_l*(alpha_h+beta_h)-
beta_l*(gamma_h-
(alpha_h+beta_h)/3.0))/(alpha_l*alpha_h+alpha_l*b
eta_h-beta_l*alpha_h);
     double xh=(alpha_l*(gamma_h-
(alpha_h+beta_h)/3.0)-
gamma_l*alpha_h)/(alpha_l*alpha_h+alpha_l*beta_
h-beta_l*alpha_h);
     double xx=xin+i*dx;

    fprintf(fp2,"%15.7f %15.7f %15.7f \n", xx, u(i),
xg+xh*xx+xx*xx/2.0-xx*xx*xx*xx/6.0);
} }
```

1维泊松方程实例


1D Poisson
Numerical Method
Exact Result
0.9
0.85
0.8
0.75
0.7
y
0
0.2
0.4
0.6
0.8
1
x

$$\frac{d^2u}{dx^2} = 1-2x^2, \quad x_l = 0, \quad x_h = 1$$

$$u(x_l) = 0, \quad u(x_h) = 0$$

let $N=3$: $\quad \delta x = \frac{x_h - x_l}{N+1} = 0.25$

$$x_i = x_l + i\,\delta x = 0.25\,i$$

解析解: $u(x) = -\frac{x}{3} + \frac{x^2}{2} - \frac{x^4}{6}$

$$\left.\frac{d^2u}{dx^2}\right|_i = \frac{u_{i-1} - 2u_i + u_{i+1}}{(\delta x)^2}, \quad i = 1, \cdots N$$

$$\Rightarrow u_{i-1} - 2u_i + u_{i+1} = (1-2x_i^2)(\delta x)^2$$

$$\begin{cases} u_0 - 2u_1 + u_2 = (1-2x_1^2)(\delta x)^2 \\ u_1 - 2u_2 + u_3 = (1-2x_2^2)(\delta x)^2 \\ u_2 - 2u_3 + u_4 = (1-2x_3^2)(\delta x)^2 \end{cases}$$

$u_0 = 0, \quad u_4 = 0$

$$\Rightarrow \begin{pmatrix} b_1 & c_1 & 0 \\ a_2 & b_2 & c_2 \\ 0 & a_3 & b_3 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

$b_1 = b_2 = b_3 = -2$

$a_2 = a_3 = 1$

$c_1 = c_2 = 1$

$d_1 = (1-2x_1^2)(\delta x)^2 = 7/128$

$d_2 = (1-2x_2^2)(\delta x)^2 = 1/32$

$d_3 = (1-2x_3^2)(\delta x)^2 = -1/128$

$a_i u_{i-1} + b_i u_i + c_i u_{i+1} = d_i$,
$i = 1, \cdots N$

设 $u_{i+1} = x_i u_i + y_i$,

$$\Rightarrow u_i = \frac{-a_i u_{i-1}}{b_i + c_i x_i} + \frac{d_i - c_i y_i}{b_i + c_i x_i}$$

$$\Rightarrow \begin{cases} x_{i-1} = \dfrac{-a_i}{b_i + c_i x_i} \\ y_{i-1} = \dfrac{d_i - c_i y_i}{b_i + c_i x_i} \end{cases} \quad i = 1, \cdots, N$$

For $i = N = 3$

$$x_{N-1} = x_2 = \frac{-a_3}{b_3 + c_3 x_3} = \frac{-a_3}{b_3} = \frac{1}{2} \quad (\text{因 } c_N = 0)$$

$$y_{N-1} = y_2 = \frac{d_3 - c_3 y_3}{b_3 + c_3 y_3} = \frac{d_3}{b_3} = \frac{1}{256}$$

$$\Rightarrow x_1 = \frac{2}{3}, \quad y_1 = \frac{-7}{384}$$

$$x_0 = \frac{-a_1}{b_1 + a x_1} = 0 \quad (\text{因 } a_1 = 0)$$

$$y_0 = \frac{d_1 - c y_1}{b_1 + c_1 x_1} = \frac{-7}{128}$$

Now we have x_0, x_1, x_2, and y_0, y_1, y_2

$$\Rightarrow u_{i+1} = x_i u_i + y_i$$

$$\begin{cases} u_1 = x_0 u_0 + y_0 = y_0 = \frac{-7}{128} \approx -0.0547 \\ u_2 = x_1 u_1 + y_1 \approx -0.0547 \\ u_3 = x_2 u_2 + y_2 \approx -0.0234 \end{cases}$$

From analytic solution:

$u_1 \approx -0.0527$ $\quad u_2 \approx -0.052$

$u_3 \approx -0.0215$.

请给出N=4，5的结果

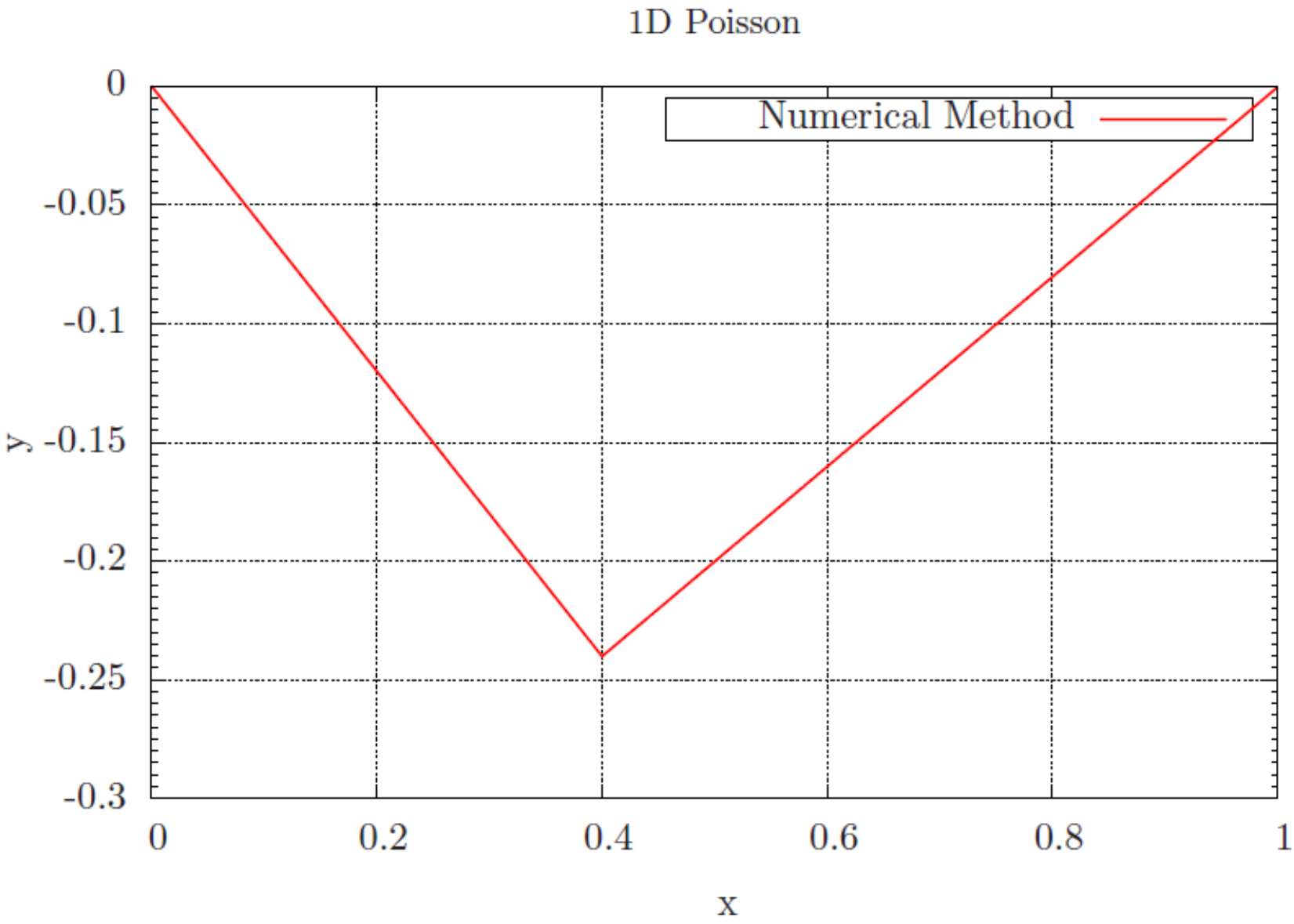
1维泊松方程实例B

```
alpha_l=1.0; beta_l=0.0; gamma_l=0.0;
alpha_h=1.0; beta_h=0.0; gamma_h=0.0;
```

点电荷

```
  for (int i = 1; i <= N; i++) {
      double xx=xin+i*dx;
      if(fabs(xx-0.4)<dx){
        v(i) = 1.0/dx;}
      else {
        v(i)=0.0;
      }
  }
```

1维泊松方程实例B


1D Poisson
Numerical Method
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
y
0
0.2
0.4
0.6
0.8
1
x

边值问题：打靶法

将微分方程的边值问题化成初值问题。

考虑第一边界条件，$y(a)=\alpha$，$y(b)=\beta$

现在需要**猜测**在$x=\alpha$点的一阶导数$y'(a)=m_1$，从而问题变为

$$y''(x)=f(x,y,y'),\qquad y(a)=\alpha,\qquad y'(a)=m_1$$

用初值问题求解方法，得解$y_1(x)$，

如果$|y_1(b)-\beta|<\varepsilon$，则认为$y_1(x)$即所求；

否则取$m_2=m_1\beta/y_1(b)$，再解初值问题

$$y''(x)=f(x,y,y'),\qquad y(a)=\alpha,\qquad y'(a)=m_2$$

用初值问题求解方法，得解$y_2(x)$，

判断$|y_2(b)-\beta|<\varepsilon$成立与否

从m_3开始可以采用割线法

$$\frac{m_3-m_2}{\beta-y_2(b)}=\frac{m_2-m_1}{y_2(b)-y_1(b)}\quad\Rightarrow$$

$$m_3=m_2+[\beta-y_2(b)](m_2-m_1)/[y_2(b)-y_1(b)]$$

边值问题：打靶法

$$m_{n+1} = m_n + [\beta - y_n(b)](m_n - m_{n-1})/[y_n(b) - y_{n-1}(b)], \quad n \geq 2$$

例：为了使烟花从地面发射5s后在距离地面40m的空中爆炸，初始的发射速度应该是多大？（考虑空气阻力与速度成正比，阻力系数为0.01）

$$\frac{d^2y}{dt^2} = -9.8 - 0.01\frac{dy}{dt},$$
$$y(0) = 0$$
$$y(5) = 40$$

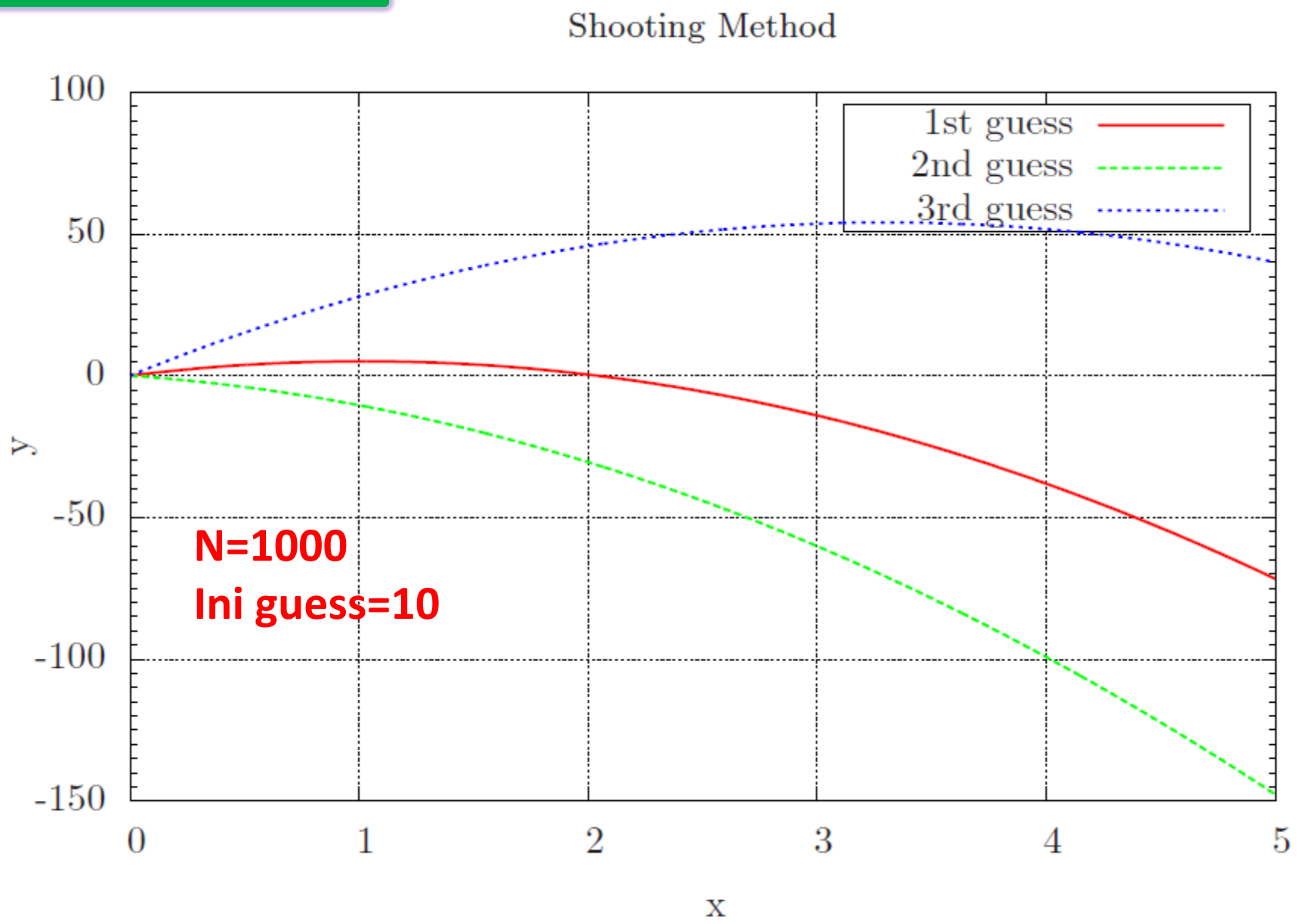
```
c  Shooting Method:
c  y''=-9.8-0.01*y'
c  y(0)=0, y(5)=40

c   (z)'=-9.8-0.01*z
c    y'=z
```

Guess=g*t=5*8=40?

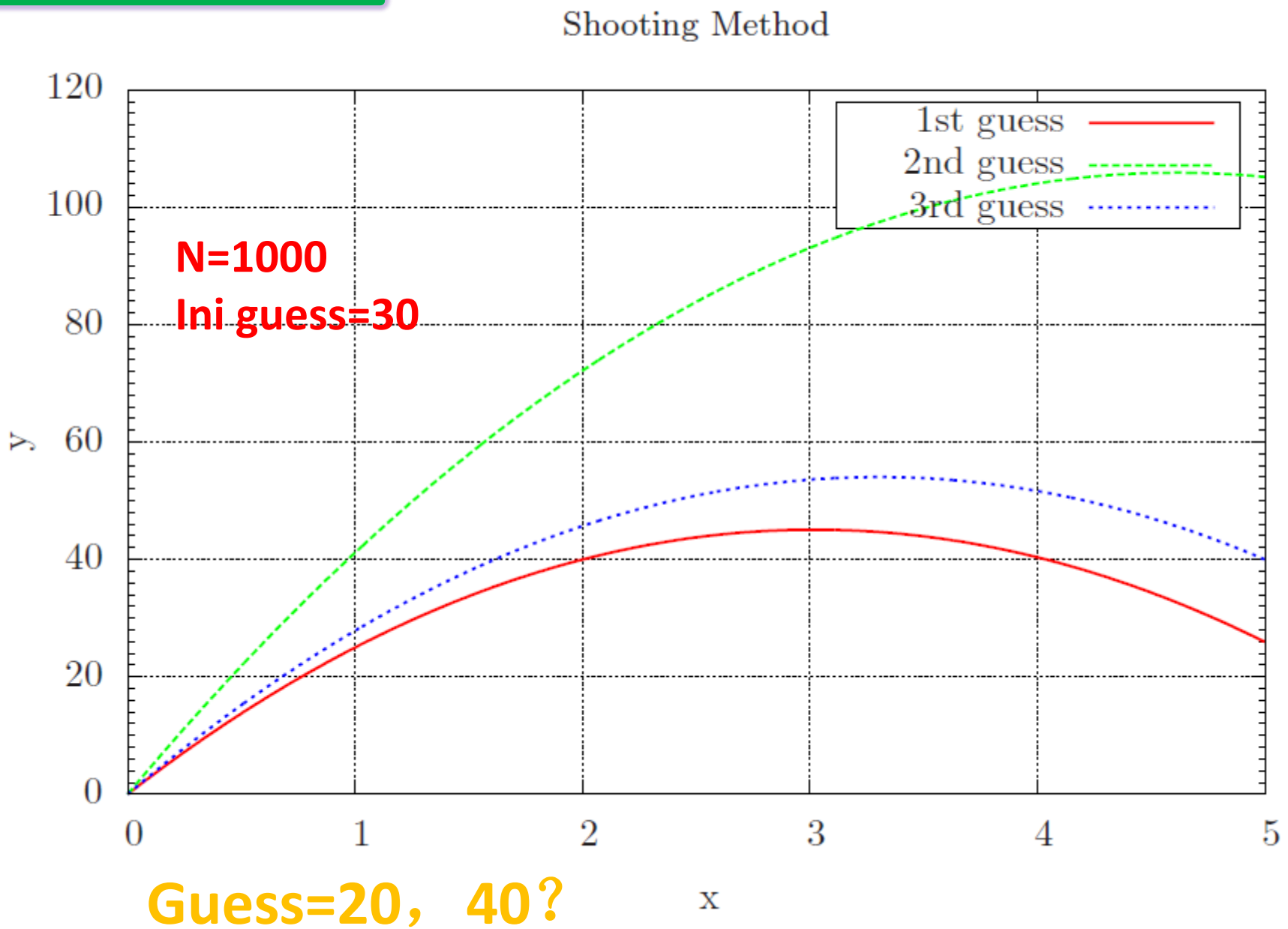
边值问题：打靶法

```
      Program main
      IMPLICIT NONE
      Real*8 nstep
      Real*8 xfinal, xini
      Real*8 xstep
      Real*8 y(10000), dy(10000)
      Real*8 guess, diff, epsi, yb
      Real*8 guess2, guess3, db1,db2
      Integer i
     character*50 file_name_out
     character*50 file_name_out2
     character*50 file_name_out3
     file_name_out = 'plot.gnu'
     open( unit=16, file=file_name_out, access="sequential",
    *      form='formatted', status="unknown" )
     file_name_out2 = 'plot2.gnu'
     open( unit=17, file=file_name_out2, access="sequential",
    *      form='formatted', status="unknown" )
     file_name_out3 = 'plot3.gnu'
     open( unit=18, file=file_name_out3, access="sequential",
    *      form='formatted', status="unknown" )
10   format( E15.7, E15.7)
      nstep=1000.
      xfinal=5.
      xini=0.
      xstep=(xfinal-xini)/nstep
      y(1)=0.0
     write(unit=16, fmt=10) 0., y(1)
     write(unit=17, fmt=10) 0., y(1)
     write(unit=18, fmt=10) 0., y(1)
      yb=40.
      epsi=0.001
      guess=30.
      dy(1)=guess
     do i=2,nstep+1
     dy(i)=dy(i-1)+xstep*(-9.8-0.01*dy(i-1))
     y(i)=y(i-1)+xstep*dy(i-1)
     write(unit=16, fmt=10) (i-1.)*xstep, y(i)
     enddo
     close( unit=16 )
     db1=y(nstep+1)
     diff=abs(y(nstep+1)-yb)
     if(diff>epsi) then
       guess2=guess*yb/y(nstep+1)
       dy(1)=guess2
      do i=2,nstep+1
      dy(i)=dy(i-1)+xstep*(-9.8-0.01*dy(i-1))
      y(i)=y(i-1)+xstep*dy(i-1)
      write(unit=17, fmt=10) (i-1.)*xstep, y(i)
      enddo
     endif
     close( unit=17 )
     db2=y(nstep+1)
     diff=abs(y(nstep+1)-yb)
     if(diff>epsi) then
       guess3=guess2+(yb-db2)*(guess2-guess)/(db2-db1)
       dy(1)=guess3
      do i=2,nstep+1
      dy(i)=dy(i-1)+xstep*(-9.8-0.01*dy(i-1))
      y(i)=y(i-1)+xstep*dy(i-1)
      write(unit=18, fmt=10) (i-1.)*xstep, y(i)
      enddo
     endif
     close( unit=18 )
     end
```

边值问题：打靶法

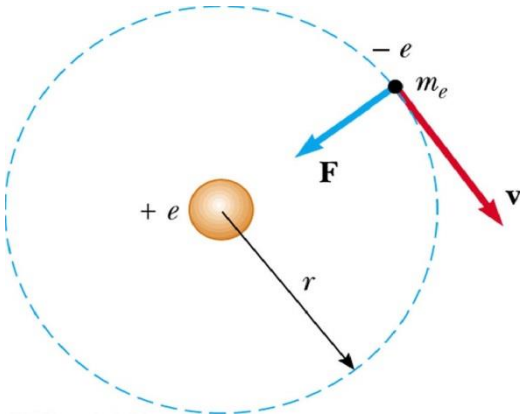

Shooting Method
1st guess
2nd guess
3rd guess
N=1000
Ini guess=10
y
x
100
50
0
-50
-100
-150
0
1
2
3
4
5

边值问题：打靶法



Guess=20，40？

薛定谔方程：氢原子能级



氢原子能级：原子各个定态对应的能量是不连续的，这些能量值叫做能级。

①能级公式：$E_n=E_1/n^2$

②半径公式：$r_n=r_1 \cdot n^2$

在氢光谱中，

n=2,3,4,5,……向n=1跃迁发光形成赖曼线系；

n=3,4,5,6……向n=2跃迁发光形成巴耳末线系；

n=4,5,6,7……向n=3跃迁发光形成帕邢线系；

n=5,6,7,8……向n=4跃迁发光形成布喇开线系，

其中只有巴耳末线系的前4条谱线落在可见光区域内。

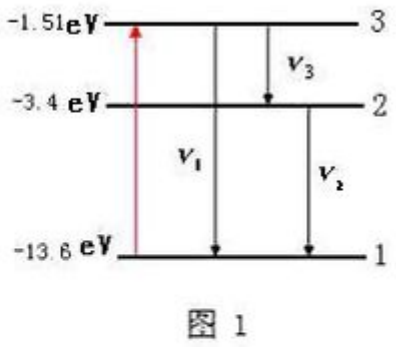


图 1

氢原子能级图

薛定谔方程：氢原子能级

在量子力学中描述微观粒子状态的波函数ψ满足标准条件和归一化条件，即ψ是时间和空间的单值、有限、连续而且归一化了的函数。其振幅函数：

$$\psi(x,y,z)=\psi_0\exp\left(\frac{i}{\hbar}p.r\right)$$

描述粒子在空间的稳定分布，只是空间坐标的函数，与时间无关。其中$\hbar=h/2\,\pi.$

在势场中，设粒子势能为U(x,y,z), 总能量为E，则一般定态薛定谔方程为

$$\Delta\psi+\frac{2m}{\hbar^2}(E-U)\psi=0$$

其中Δ为拉普拉斯算子$\nabla.\nabla$。
只有当总能量E为某些特定值时，方程才可能有解。这些E的特定值就成为本征值，相应的波函数则成为本征波函数。

氢原子势能函数$U=-\frac{e^2}{4\pi\varepsilon_0 r}$

薛定谔方程: 氢原子能级

$$\Delta\psi + \frac{2m}{\hbar^2}(E + \frac{e^2}{4\pi\varepsilon_0 r})\psi = 0$$

在球坐标系下,变为

$$\frac{1}{r^2}\frac{d}{dr}(r^2\frac{\partial\psi}{\partial r})+\frac{1}{r^2 sin\theta}\frac{\partial}{\partial\theta}(sin\theta\frac{\partial\psi}{\partial\theta})+\frac{1}{r^2 sin\theta}\frac{\partial^2\psi}{\partial\varphi^2}+\frac{2m}{\hbar^2}(E+\frac{e^2}{4\pi\varepsilon_0 r})\psi = 0$$

采用分离变量的办法, 令$\psi(r,\theta,\varphi) = R(r)\Theta(\theta)\Phi(\varphi)$

$$\begin{cases} \dfrac{d^2\Phi}{d\varphi^2} + m_l^2\Phi = 0 \\ \dfrac{1}{sin\theta}\dfrac{d}{d\theta}\left(sin\theta\dfrac{d\Theta}{d\theta}\right) + \left[l(l+1) - \dfrac{m_l^2}{sin^2\theta}\right]\Theta = 0 \\ \dfrac{1}{r^2}\dfrac{d}{dr}\left(r^2\dfrac{dR}{dr}\right) + \dfrac{2m}{\hbar^2}\left[E + \dfrac{e^2}{4\pi\varepsilon_0 r} - \dfrac{\hbar}{2m}\dfrac{l(l+1)}{r^2}\right]R = 0 \end{cases}$$

其中, m_l是磁量子数, l是角量子数.为了满足标准条件和归一化条件, 它们只能取某些特定的值.设主量子数为n,则

$$l = 0,1,\dots,n-1$$

$$m_l = 0,\pm1,\dots,\pm l$$

薛定谔方程: 氢原子能级

对于氢原子的基态, 亦即1s态, 有l=0, m_l=0, 电子分布成球对称型.
令$rR = A$

$$\frac{d^2A}{dr^2} = -\frac{2m}{\hbar^2}\left[E + \frac{e^2}{4\pi\varepsilon_0 r}\right]A = 0$$

或写成

$$\frac{d^2A}{dr^2} = -c_1\left[E + \frac{c_2}{r}\right]A$$

其中

$$c_1 = \frac{2m}{\hbar^2} = 26.2513\ (/\text{eV}/nm^2)$$

$$c_2 = \frac{e^2}{4\pi\varepsilon_0} = 1.43998\ (\text{eV.nm})$$

具体的几个本征函数如下：

$$\mathrm{R}_{10} = 2\left(\frac{1}{a_0^{3/2}}\right)e^{-r/a_0}, \quad \mathrm{R}_{10} = \frac{1}{(2a_0)^{\frac{3}{2}}}(2 - \frac{r}{a_0})e^{-r/(2a_0)},$$

其中$a_0 = 4\pi\varepsilon_0\hbar^2/(\mathrm{m}e^2)$~0.0529nm 为玻尔半径。

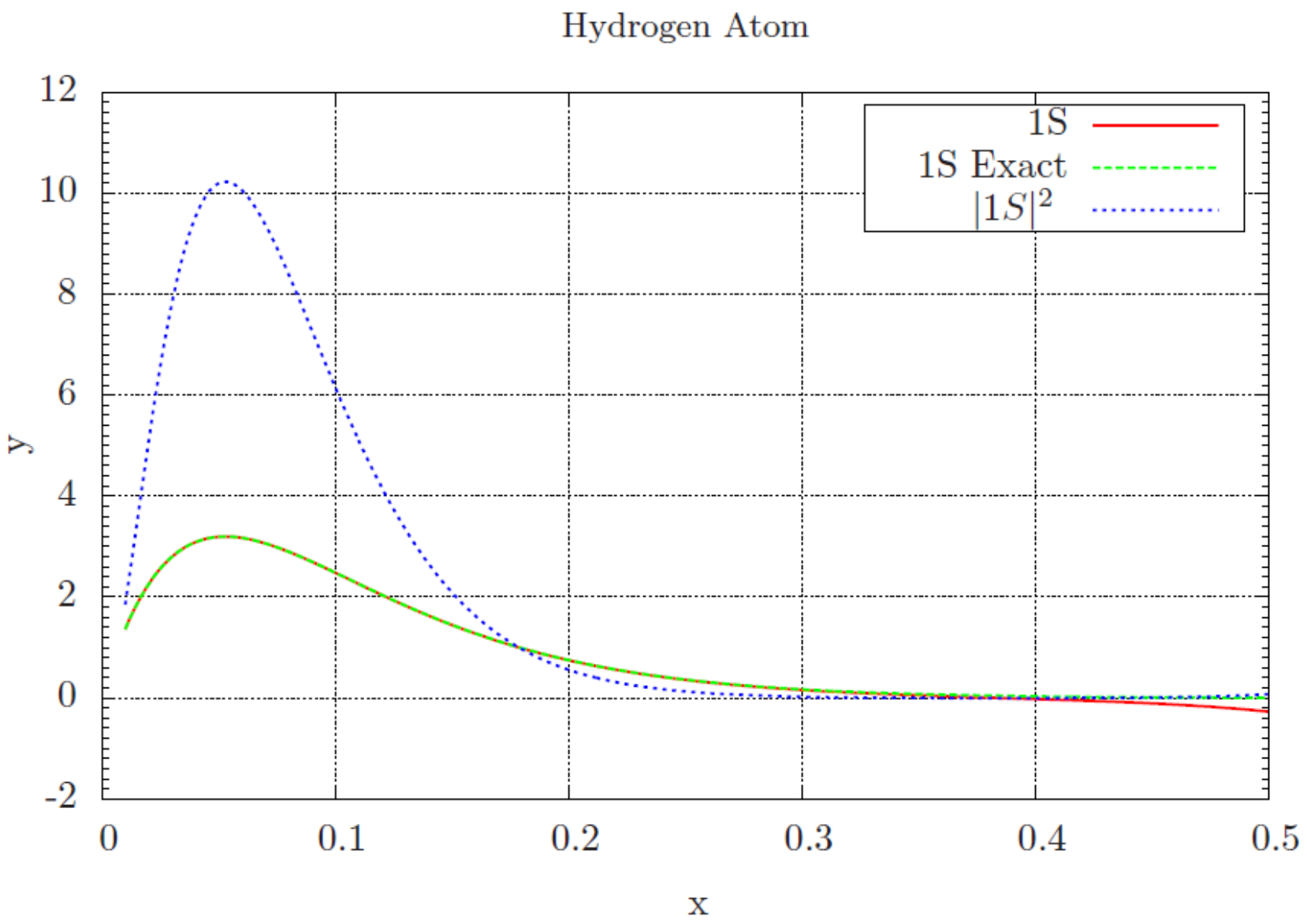
径向概率分布$\boldsymbol{P_{nl}dr = r^2dr\int|R_{nl}|^2|Y_{lm}|^2d\Omega = |R_{nl}|^2r^2dr}$

氢原子能级

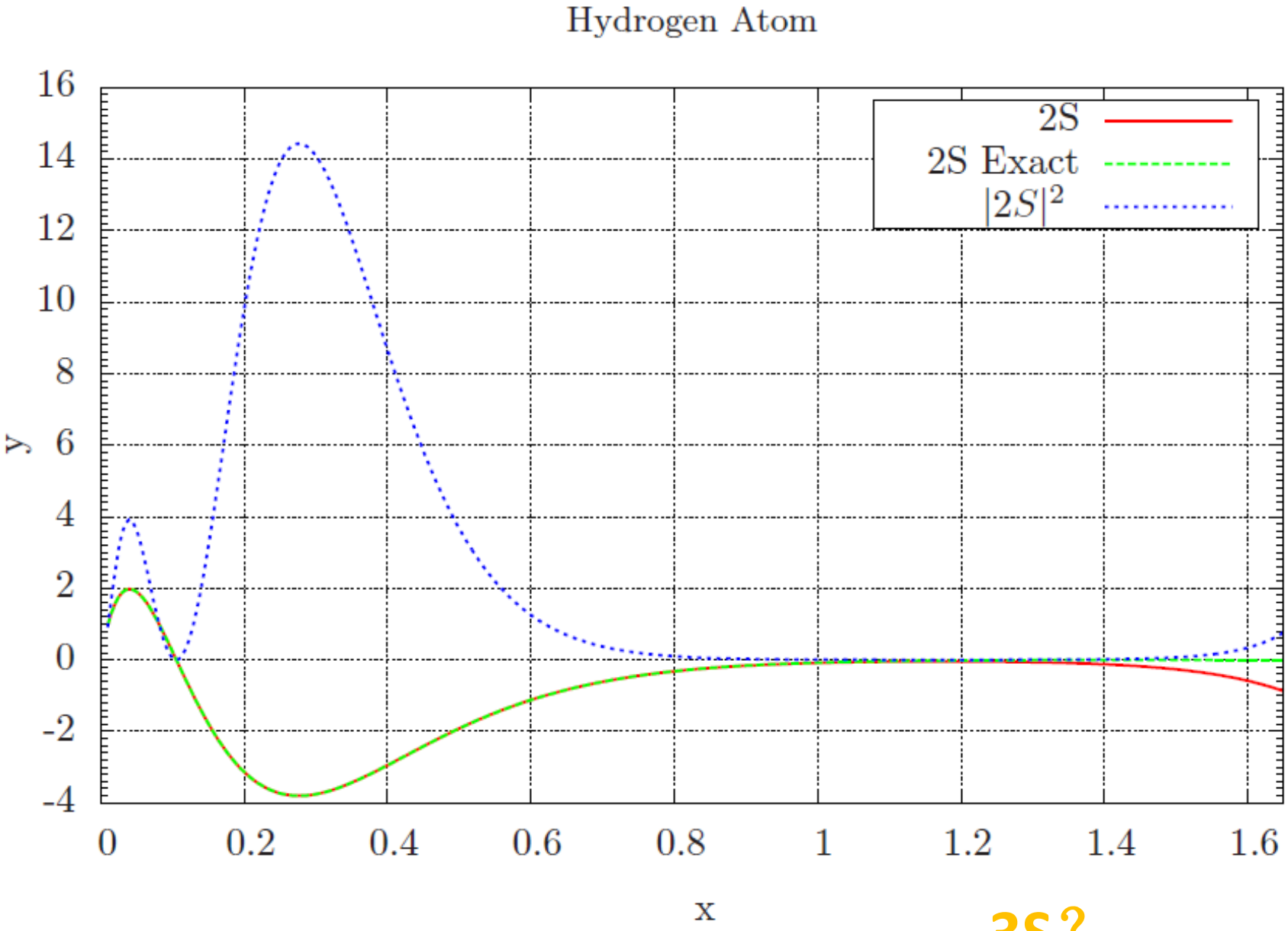
```
#include <fstream>
#include <ostream>
#include <iostream>
#include "TMath.h"
#include "TFile.h"
using namespace std;
float fa(float m, float n1, float n0)
{  return(n1); }
float fb(float m, float n1, float n0)
{ float c1=26.2513;
  float c2=1.43998;
  float E=-13.60829;
  return(-c1*(E+c2/m)*n0);}
main9_RK_1S()
{ char *out= "plot-rk-n1S.gnu";
  FILE *fp2 = fopen(out,"w");
  float nstep=500., xfinal=0.5, xini=0.01,
  y11=1.3606588,  y21=110.304;
  float k, k1, k2, k3, k4;
  float l, l1, l2, l3, l4;
  float a0=0.052918; // nm
  float solution;
 float xstep=(xfinal-xini)/nstep;
```

```
float r=xini, a=y11, b=y21;
  unsigned i;
solution=2.*sqrt(1./a0/a0/a0)*exp(-r/a0)*r;
   fprintf(fp2,"%15.7f %15.7f %15.7f %15.7f %15.7f
\n", r, a, a*a, b, solution);
  for(i=0;i<nstep+1;i++)
  { k1=fa(r,b,a);
    l1=fb(r,b,a);
    k2=fa(r+xstep/2., b+xstep*l1/2., a+xstep*k1/2.);
    l2=fb(r+xstep/2., b+xstep*l1/2., a+xstep*k1/2.);
    k3=fa(r+xstep/2., b+xstep*l2/2., a+xstep*k2/2.);
    l3=fb(r+xstep/2., b+xstep*l2/2., a+xstep*k2/2.);
    k4=fa(r+xstep, b+xstep*l3, a+xstep*k3);
    l4=fb(r+xstep, b+xstep*l3, a+xstep*k3);
    k=xstep*(k1+2*k2+2*k3+k4)/6.0;
    l=xstep*(l1+2*l2+2*l3+l4)/6.0;
     r=r+xstep; a=a+k; b=b+l;
    solution=2.*sqrt(1./a0/a0/a0)*exp(-r/a0)*r;
    fprintf(fp2,"%15.7f %15.7f %15.7f %15.7f
%15.7f\n", r, a, a*a, b, solution);
  }}
```

氢原子能级



氢原子能级


Hydrogen Atom
2S
2S Exact
$|2S|^2$
y
x
16
14
12
10
8
6
4
2
0
-2
-4
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6

3S?

作业：

1. **P22：** 电磁学常微分方程组，**RK4**求解
   ω=1.； 初始位置(0, 0, 0)，初始速度(0, 2, 0.1)

2. 数值求解二阶偏微分方程
   $y''-2y'+2y=e^{2x}\sin(x)$， x:[0-1]
   with y(0)=-0.4, y'(0)=-0.6

3. **P31** 洛伦兹吸引子，重复结果，并 额外给出至少2组不同的β，ρ，σ的结果

4. **P58** 给出数值结果