信陽師範学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

第9章 MATLAB程序设计

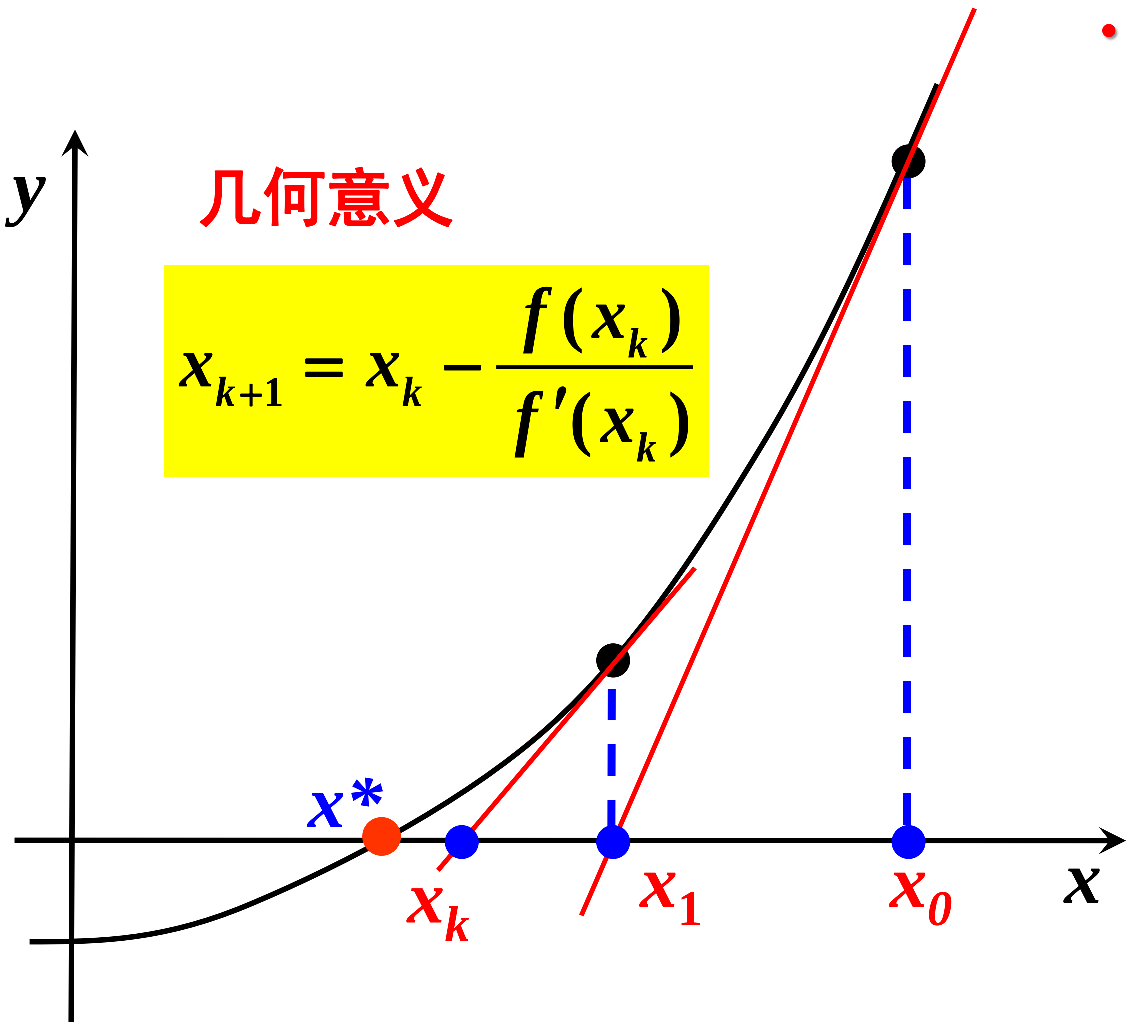
讲授人：牛言涛

日期：2020年4月4日

目录 CONTENTS

MATLAB程序设计

1. 方程求根：牛顿迭代法



- **牛顿迭代法的主要思想**
  - 牛顿迭代法又称**切线法**，是一种有特色的求根方法。 用牛顿迭代法求$f(x) = 0$的单根x^*的主要步骤为：

① Newton法的迭代公式

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0,1,2,\dots$$

② 以x^*附近的某一个值x_0为迭代初值，代入迭代公式，反复迭代，得到序列$x_0, x_1, x_2, \dots\dots$

③ 若序列收敛，则必收敛于精确根x^*，

即$\lim\limits_{n\to\infty} x_n = x^*$。

1. 方程求根：牛顿迭代法

Newton_iterative.m

```matlab
function sol = Newton_iterative(f,x0,eps,maxiter)
% Newton_iterative 使用牛顿迭代法求解非线性方程的根
% sol = Newton_iterative(f,x0,eps,maxiter)
% sol = Newton_iterative(f,x0,eps)
% Newton_iterative(f,x0)

%输入参数：
% 1.f是求解的非线性方程，要求输入的是符号方程
% 2.x0是初值
% 3. eps是精度，如1e-16，默认1e-12
% 4. maxiter是最大迭代次数，默认最大迭代次数100
% 输出参数：sol是一个结构体，包含了迭代过程的信息

%示例....

  %% 输入参数的控制
  if nargin == 3
      maxiter = 100; %默认100次
  elseif nargin == 2
      maxiter = 100;
      eps = 1e-12; %默认精度
  elseif nargin < 2 || nargin > 4
      error('输入参数个数不足或参数个数too many...')
  end

  % x(k+1) = x(k) - f(x(k))/f'(x(k)) 迭代公式

%% 变量的初始化
x_n = x0; % x(k+1)表示下一次值
df = diff(f,symvar(f),1); %方程的一阶导

fprintf('\n%5s %20s %20s\n','iter','ApproximateSolution',...
    'ErrorPrecision');
%% 利用牛顿迭代思想，进行数值逼近
for k = 1:maxiter
    x_b = x_n;  %迭代序列，x0,x1,x2,x3....，其中x_b表示x(k)
    fx = subs(f,symvar(f),x_b);  %求f(x(k))
    dfx = subs(df,symvar(f),x_b);
    x_n = double(x_b - fx/dfx);  %迭代公式
    errval = abs(double(subs(f,symvar(f),x_n)));  %每次迭代误差大小
    % 迭代过程的输出
    fprintf('%3d %20.15f %24.15f\n',k,x_n,errval);
    if errval <= eps  %满足精度，退出循环
        break
    end
end

%% 迭代收敛的问题
if k < maxiter
    disp(['方程在满足精度',num2str(eps),'的条件下，近似解为：',...
        num2str(x_n),'，误差：',num2str(errval)])
else
    disp('牛顿迭代求解数值逼近，到达最大迭代次数，可能不收敛....')
    return
end
```

1. 方程求根：牛顿迭代法


```matlab
%% 输出参数的控制
if nargout == 1
    sol.info = '迭代收敛，逼近终止';
    sol.X = x_n;
    sol.norm_error = errval;
    sol.iterative = k;
    sol.eps = eps;
    sol.success = '成功';
end

%% 可视化问题
fplot(f,[x_n-x0,x_n+x0],'LineWidth',1.5)
hold on
grid on
fplot(@(t)0.*t,[x_n-x0,x_n+x0])
plot(x_n,errval,'r.','MarkerSize',20)
title('牛顿迭代法求解非线性方程的近似解')
xlabel('X')
ylabel('Y')
end
```

例1：用牛顿迭代法求 $f(x)=2e^{-x}\sin x+2\cos x-0.25=0$

命令行窗口

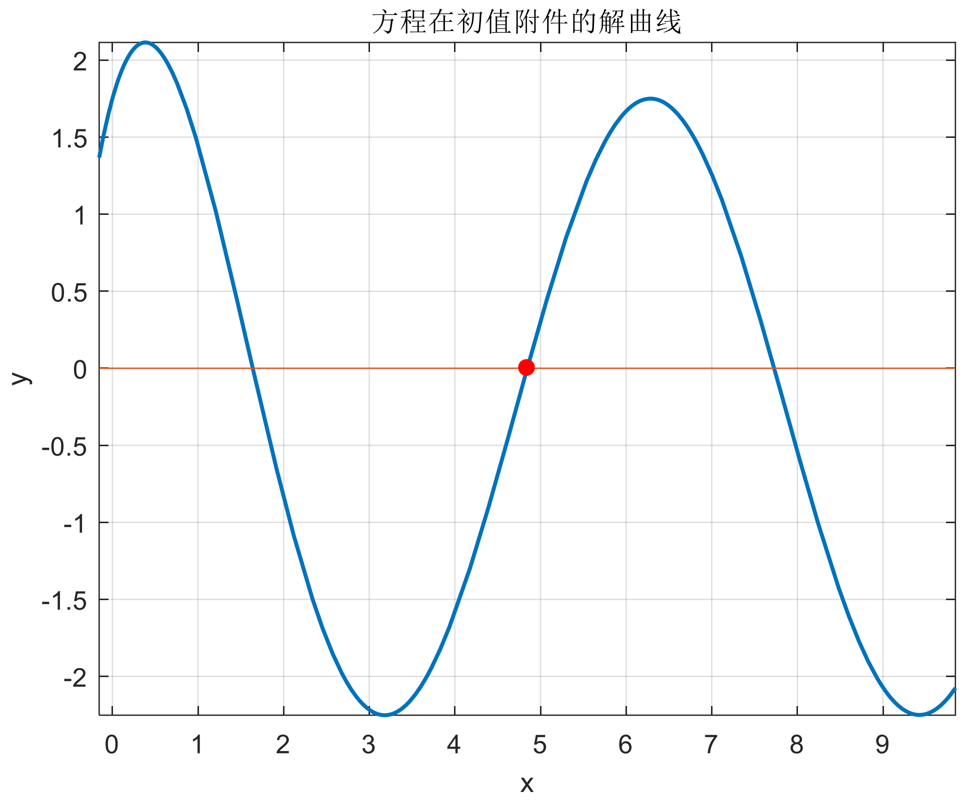
```
>> syms x
>> fh = 2*exp(-x)*sin(x)+2*cos(x)-0.25;
>> eps = 1e-10;
>> x0 = 1;
>> max = 100;
>> sol = Newton_iterative(fh,x0,eps,max)

 iter   ApproximateSolution       ErrorPrecision
  1     1.761198166196976        0.291040129889134
  2     1.638207816540044        0.003052361728492
  3     1.639474666447673        0.000000151045006
  4     1.639474729143502        0.000000000000000
方程在满足精度1e-10的条件下，近似解为：1.6395，误差：1.8814e-16
sol =
  包含以下字段的 struct:

          info: '迭代收敛，逼近终止'
             X: 1.6395
    norm_error: 1.8814e-16
     iterative: 4
           eps: 1.0000e-10
       success: '成功'
```

1. 方程求根：牛顿迭代法

例1：用牛顿迭代法求 $f(x)=2e^{-x}\sin x+2\cos x-0.25=0$

```
>> syms x
>> fh = 2*exp(-x)*sin(x)+2*cos(x)-0.25;
>> eps = 1e-10;
>> x0 = 5;
>> max = 100;
>> x0 = 5;
>> Newton_iterative(fh,x0,eps,max)
```

iter	ApproximateSolution	ErrorPrecision
1	4.842653246253955	0.005846228472196
2	4.845575275645596	0.000001134937422
3	4.845575843124848	0.000000000000043



方程在满足精度1e-10的条件下，近似解为：4.8456，误差：4.2759e-14

2. 方程组求解：雅克比迭代法

雅可比迭代法的主要思想：对线性方程组进行同解变形，使方程i的等号左端只有x_i，等号右端不出现x_i，其中$i = 1, 2, \dots, n$。

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & \cdots + & a_{1,n}x_n = & b_1 \\ a_{21}x_1 + & a_{22}x_2 + & \cdots + & a_{2,n}x_n = & b_2 \\ \cdots & \cdots & \cdots & \cdots & \\ a_{n,1}x_1 + & a_{n,2}x_2 + & \cdots + & a_{n,n}x_n = & b_n \end{cases}$$

同解变形

$$\begin{cases} x_1 = (-a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n + b_1)/a_{11} \\ x_2 = (-a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n + b_2)/a_{22} \\ \cdots \quad \cdots \quad \cdots \quad \cdots \\ x_n = (-a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1} + b_n)/a_{nn} \end{cases}$$

由上式构造雅可比迭代法的迭代公式：

$$\begin{cases} x_1^{(k+1)} = (-a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)} + b_1)/a_{11} \\ x_2^{(k+1)} = (-a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_n^{(k)} + b_2)/a_{22} \\ \cdots \quad \cdots \quad \cdots \quad \cdots \\ x_n^{(k+1)} = (-a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \cdots - a_{n,n-1}x_{n-1}^{(k)} + b_n)/a_{nn} \end{cases}$$

$$x_i^{(k+1)} = \frac{\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}\right)}{a_{ii}} \quad i = 1, 2, \cdots, n,\ k = 0, 1, \cdots$$

取迭代初值为向量$x^{(0)}$，反复迭代，得到序列$x^{(0)}$，$x^{(1)}$，$x^{(2)}\cdots$。如果序列收敛于向量x^*，那么向量x^*就是原线性方程组的精确解向量。

2. 方程组求解：雅克比迭代法

信阳师范学院 数学与统计学院 SCHOOL OF MATHEMATICS AND STATISTICS

考虑线性方程组$\boldsymbol{Ax}=\boldsymbol{b}$，其中$\boldsymbol{A}=(\boldsymbol{a_{ij}})_{n\times m}$非奇异，且对角线元素全不为0，将A分解成$\boldsymbol{A}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$，其中

$$D=diag\left(a_{11},a_{22},\cdots,a_{nn}\right),\quad L=\begin{bmatrix} 0 & & & \\ -a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix},\quad U=\begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ & 0 & \ddots & \vdots \\ & & \ddots & -a_{n-1,n} \\ & & & 0 \end{bmatrix}$$

可得雅可比(Jacobi)迭代方法：$x^{(k+1)}=D^{-1}\left(L+U\right)x^{(k)}+D^{-1}b,\ k=0,1,2,\cdots$

迭代矩阵记为 $J=D^{-1}\left(L+U\right)$

对任意选取初始向量$\boldsymbol{x}^{(0)}$，迭代法收敛的充要条件是矩阵$\boldsymbol{J}$的谱半径$\boldsymbol{\rho(J)<1}$.

（$\rho(J)=\max\limits_{1\le i\le n}\left|\lambda_i\right|$为$J$的谱半径，其中$\lambda_i$为$J$的特征值。）

2. 方程组求解：雅克比迭代法

信阳师范学院 数学与统计学院 SCHOOL OF MATHEMATICS AND STATISTICS

例2：求解 $\begin{cases} x_1 + 2x_2 - 2x_3 = 1 \\ x_1 + x_2 + x_3 = 3 \\ 2x_1 + 2x_2 + x_3 = 5 \end{cases}$ ，取初始向量 $x^{(0)} = (0,0,0)^T$

解：写出雅可比迭代形式并求解 $x^{(k+1)} = \begin{pmatrix} 0 & -2 & 2 \\ -1 & 0 & -1 \\ -2 & -2 & 0 \end{pmatrix} x^{(k)} + \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix}, \quad k = 0,1,\ldots$

其中 $J = \begin{pmatrix} 0 & -2 & 2 \\ -1 & 0 & -1 \\ -2 & -2 & 0 \end{pmatrix}$ ，特征多项式 $|\lambda I - J| = \begin{vmatrix} \lambda & 2 & -2 \\ 1 & \lambda & 1 \\ 2 & 2 & \lambda \end{vmatrix} = \lambda^3 = 0 \Rightarrow \rho(B) = \max\limits_{1\le i\le n} |\lambda_i| = 0 < 1$

$x^{(0)} = (0,0,0)^T, x^{(1)} = (1,3,5)^T, \left\| x^{(1)} - x^{(0)} \right\|_\infty = 5 \qquad x^{(2)} = (5,-3,-3)^T, \left\| x^{(2)} - x^{(1)} \right\|_\infty = 8$

$x^{(3)} = (1,1,1)^T, \left\| x^{(3)} - x^{(2)} \right\|_\infty = 4 \qquad x^{(4)} = (1,1,1)^T, \left\| x^{(4)} - x^{(3)} \right\|_\infty = 0$ ，所以 $x^* = x^{(4)} = (1,1,1)^T$

2. 方程组求解：雅克比迭代法

信阳师范学院
数学与统计学院

Jacobi_DD.m

```matlab
function output = Jacobi_DD(A,b,x0,eps,maxiter)
% 雅克比迭代法求解方程组的近似解
% A为系数矩阵，b为右端向量，x0为迭代初值，eps为精度要求，max为最大迭代次数
% 方程组示例：A = [8,-3,2;4,11,-1;6,3,12]; b = [20;33;36]; x0 = [0;0;0];
% 求解示例：eps = 0.000005; maxiter = 50; Jacobi_DD(A,b,x0,eps,maxiter)

  %% 输入矩阵判别
  [m,n] = size(A);
  if m ~= n
      error('矩阵不是方阵，不能使用雅可比迭代法！')
  elseif m ~= length(b)
      error('矩阵的维度与右端向量的维度不一致！')
  end

  %% 判断雅可比迭代矩阵是否收敛
  if det(A) ~= 0   %方阵非奇异
      D = diag(A);
      if ~all(D == 0) %判断对角线元素是否都是0
          J = inv(diag(D))*(A - diag(D)); %迭代矩阵
          lamdar = max(abs(eig(J)));
          if lamdar >= 1
              error('雅可比矩阵不收敛！')
          end
      else
          error('矩阵对角线都是0元素，不适宜雅可比迭代法。')
      end
  else
      error('奇异矩阵，不能用雅可比迭代法')
  end
```

Jacobi_DD.m

```matlab
  %% 输入参数个数判别，设置缺省值
  if nargin == 4
      maxiter = 100;
  elseif nargin == 3
      maxiter = 100;
      eps = 1e-12;
  elseif nargin == 2
      maxiter = 100;
      eps = 1e-12;
      x0 = rand(length(b),1);
  elseif nargin < 2
      error('输入参数个数不足！')
  end

  %% 变量的初始化
  n = length(A);
  iter = 0; %迭代次数
  x_n = x0;%下一次迭代向量值
  x_b = x0;%上一次迭代向量值

  %% 循环迭代求解
  for num = 1:maxiter
      iter = iter + 1; %迭代次数加一
      x_b = x_n; %近似解的迭代
      for i = 1:n
          S = sum(A(i,:)*x_b) - A(i,i)*x_b(i);
          x_n(i) = (b(i) - S)/A(i,i);  %雅克比迭代公式
```

2. 方程组求解：雅克比迭代法

```
            end
            r = norm(b - A*x_n); %计算误差模
            if r <= eps  %满足精度要求，退出
                break;
            end
        end

    %% 判断迭代次数是否达到上限
    if num > maxiter
        error('雅克比迭代法迭代次数已经达到上限，迭代失败！');
    end
    if nargout == 1
        output.info = '优化终止';
        output.convergence = strcat('谱半径',num2str(lamdar),...
            '，收敛到近似解。');
        output.iter = iter;
        output.sol_x = x_n;
        output.eps = r;
        output.success = '成功';
    elseif nargout < 1
        disp(x_n)
    end

end
```

```
>> A = [8,-3,2;4,11,-1;6,3,12];
>> b = [20;33;36];
>> Jacobi_DD(A,b)
    3.0000
    2.0000
    1.0000
>> output = Jacobi_DD(A,b)
output =
  包含以下字段的 struct:

           info: '优化终止'
    convergence: '谱半径0.35925，收敛到近似解。'
           iter: 31
          sol_x: [3×1 double]
            eps: 4.1256e-13
        success: '成功'
```

3. 方程组直接法求解：LU分解法

基本思想：将系数矩阵A转变成等价的L和U的乘积$A = LU$，L和U分别是下三角和上三角矩阵，而且要求L的对角元素都是1；这样可把$Ax = b$写成$Ax = (LU)x = L(Ux) = b$。令$Ux = y$，则$Ax = b$可首先求解向量y使$Ly = b$，然后求解 $Ux = y$，从而达到求解$Ax = b$的目的。

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{44} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}$$

$$= \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ l_{21}u_{11} & l_{21}u_{12}+u_{22} & l_{21}u_{13}+u_{23} & l_{21}u_{14}+u_{24} \\ l_{31}u_{11} & l_{31}u_{12}+l_{32}u_{22} & l_{31}u_{13}+l_{32}u_{23}+u_{33} & l_{31}u_{14}+l_{32}u_{24}+u_{34} \\ l_{41}u_{11} & l_{41}u_{12}+l_{42}u_{22} & l_{41}u_{13}+l_{42}u_{23}+l_{43}u_{33} & l_{41}u_{14}+l_{42}u_{24}+l_{43}u_{34}+u_{44} \end{bmatrix}$$

$$\bullet \to \bullet \to \bullet \to \bullet \to \bullet \to \bullet \to \bullet$$

按颜色顺序依次计算

$$u_{1j} = a_{1j}, \quad j = 1,2,\cdots,n$$

$$l_{i1} = \frac{a_{i1}}{u_{11}}, \quad i = 2,3,\cdots,n$$

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}$$

$$j = i, i+1, \cdots, n$$

$$l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}) / u_{jj}$$

$$i = j+1, j+2, \cdots, n$$

3. 方程组直接法求解：LU分解法

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

- LU分解法求解线性方程组的过程

1、对系数矩阵A进行LU分解，得到L和U。

方程组 $Ax=b \Leftrightarrow (LU)x=b \Leftrightarrow L(Ux)=b$

2、令向量$y=Ux$，代入上式得：$Ly=b$

回代，求解$Ly=b$，得到y。

$Ly=b$可表示为

$$\begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \cdots & \cdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

通项公式为 $y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k,\ i=1,2,\cdots,n$

3、回代，求解 $Ux=y$，得到解向量 x。

$Ux=y$可表示为

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1,n} \\ & u_{22} & \cdots & u_{2,n} \\ & & \ddots & \vdots \\ & & & u_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

通项公式为

$$x_i = (y_i - \sum_{k=i+1}^{n} u_{ik} x_k) / u_{ii},\ i=n,n-1,\cdots,1$$

3. 方程组直接法求解：LU分解法

LU_decompose.m

```matlab
function output = LU_decompose(A,b)
%% LU分解法，A为系数矩阵，b为右端向量，output输出为结构体数组。
%示例：A = [-2,-2,3,5;1,2,1,-2;2,5,3,-2;1,3,2,3];
%示例：b = [-1;4;7;0];
%调用格式：output = LU_decompose(A,b)

%% 输入矩阵判别
[m,n] = size(A);
[mb,nb] = size(b);
if m ~= n
    error('矩阵不是方阵，不能使用LU分解法！')
elseif m ~= length(b)
    error('矩阵的维度与右端向量的维度不一致！')
elseif mb < nb %确保右端向量b为列向量
    b = b';
end

%% 变量的初始化
L = eye(n);  %生成一个nxn大小的单位矩阵
U = zeros(n,n);
x = zeros(n,1);
y = zeros(n,1);

%% L和U的分解过程
U(1,:) = A(1,:);  %U的第一行与A相同
L(:,1) = A(:,1)/U(1,1); %求L的第一列
for r =2:n  %每次循环计算第r行和第r列
    for j =r:n  %列在变化，求第r行
        U(r,j) = A(r,j) - sum(L(r,1:r-1) * U(1:r-1,j));  %U的计算模型
    end
    for i = r+1:n %行在变化，求第r列
        L(i,r) = (A(i,r) - sum(L(i,1:r-1) * ...
            U(1:r-1,r)))/U(r,r); %L的计算模型
    end
end

%% 以下为方程的回代求解计算过程
y(1) = b(1); %令Ly=b，求y
for i = 2:n
    y(i) = b(i) - sum(L(i,1:i-1) * y(1:i-1));
end
x(n) = y(n)/U(n,n); %令Ux=y，求x
for i = n-1:-1:1
    x(i) = (y(i) - sum(U(i,i+1:n) * x(i+1:n)))/U(i,i);
end
%% 结果输出
if norm(b - A*x) < 1e-16
    output.L = L;
    output.U = U;
    output.Y = y;
    output.X = x;
    output.eps = norm(b - A*x);
else
    warning('线性方程组的求解可能存在精度问题，请自行验证...');
end
end
```

3. 方程组直接法求解：LU分解法

例3：用LU分解法求解 $\begin{pmatrix} -2 & -2 & 3 & 5 \\ 1 & 2 & 1 & -2 \\ 2 & 5 & 3 & -2 \\ 1 & 3 & 2 & 3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -1 \\ 4 \\ 7 \\ 0 \end{pmatrix}$

```
%命令窗口调用函数
>> A = [-2,-2,3,5;1,2,1,-2;2,5,3,-2;1,3,2,3];
>> b = [-1,4,7,0];
>> output = LU_decompose(A,b)
output =
  包含以下字段的 struct:
      L: [4×4 double]
      U: [4×4 double]
      Y: [4×1 double]
      X: [4×1 double]
    eps: 0
```

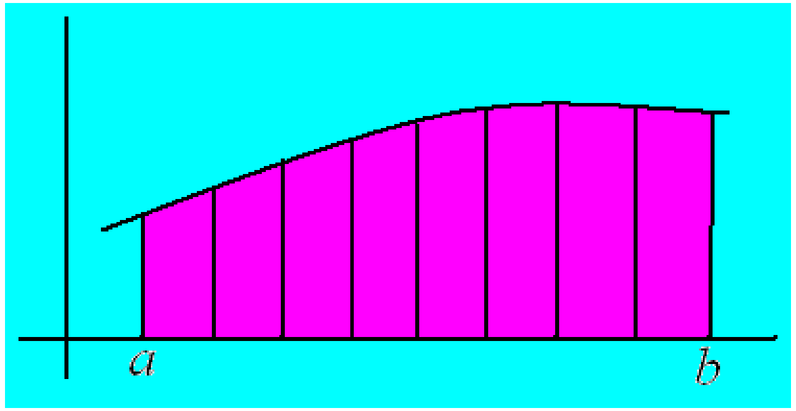
```
>> output.L
    1.0000         0         0         0
   -0.5000    1.0000         0         0
   -1.0000    3.0000    1.0000         0
   -0.5000    2.0000    1.0000    1.0000
>> output.U
   -2.0000   -2.0000    3.0000    5.0000
         0    1.0000    2.5000    0.5000
         0         0   -1.5000    1.5000
         0         0         0    3.0000
>> output.sol'
ans =
     2    -1     2    -1
```

4. 数值积分：牛顿—科特斯积分法

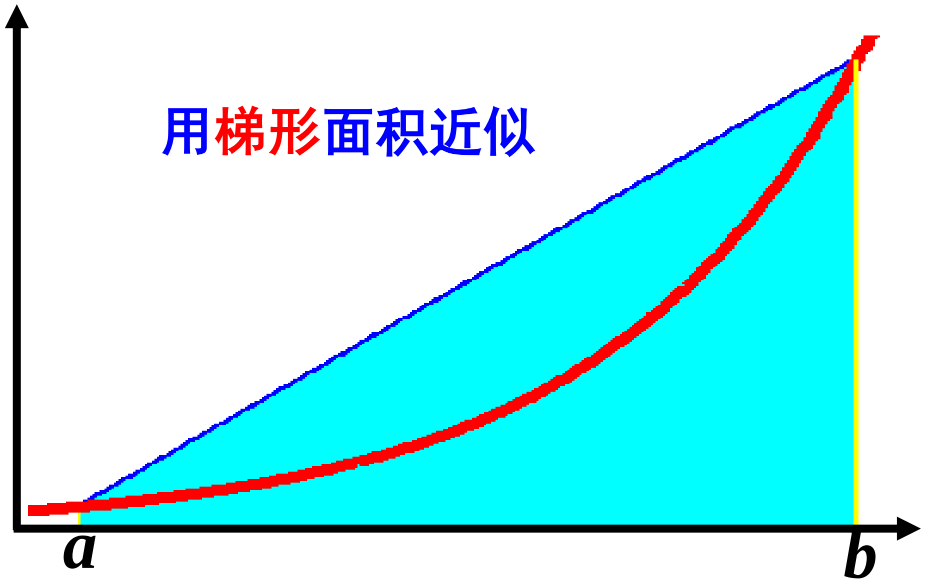
- 牛顿-柯特斯公式基本思想：
  - 牛顿-柯特斯（Newton-Cotes）公式，简记为N-C公式，是一种等距节点的代数插值型求积公式（插值点不宜过多，否则出现Runge振荡现象）。
  - 公式：积分区间为$[a,b]$，$n+1$个求积节点$x_0, x_1, \dots, x_n$把积分区间n等分，步长$h=(b-a)/n$，即$x_0=a$，$x_n=b$，$x_i=a+i\times h$（$i=0,1,\dots,n$），则N-C公式的一般形式为：

$$\int_a^b f(x)dx \approx (b-a)\sum_{i=0}^{n}\left(C_i^{(n)} f(x_i)\right)$$

其中 $C_i^{(n)} = \dfrac{(-1)^{n-i}}{ni!(n-i)!}\displaystyle\int_0^n \prod_{\substack{j=0\\ j\neq i}}^{n}(s-j)ds$ 称为柯特斯系数.



4. 数值积分：牛顿—科特斯积分法

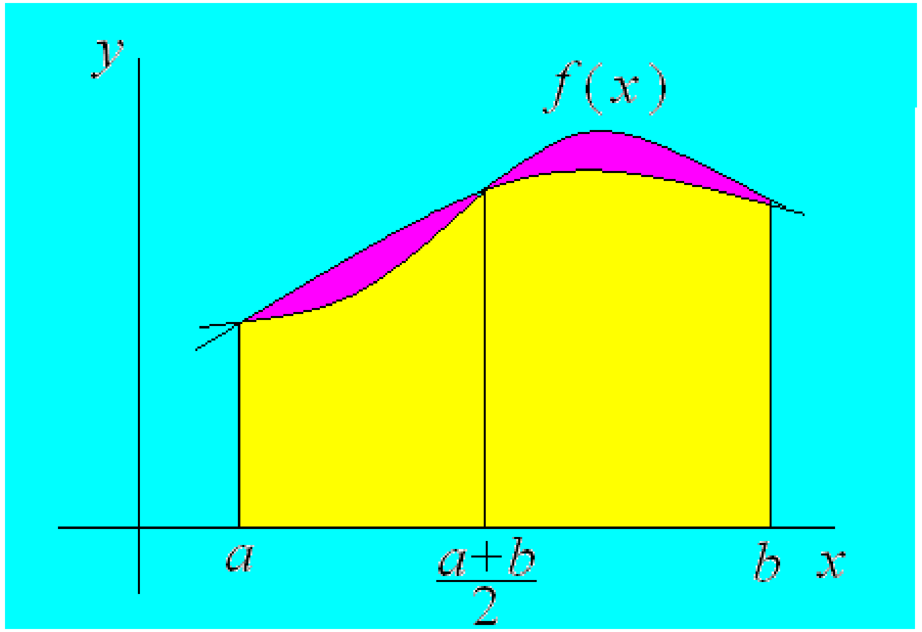


$$\int_a^b f(x)dx \approx (b-a)\left(\frac{f(x_0)}{2}+\frac{f(x_1)}{2}\right)$$

$$=(b-a)\frac{f(a)+f(b)}{2}$$

$$\int_a^b f(x)dx \approx (b-a)\left(\frac{f(x_0)}{6}+\frac{2f(x_1)}{3}+\frac{f(x_2)}{6}\right)$$

$$=\frac{(b-a)}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)$$



辛普生公式，4点为辛普生3/8法则

4. 数值积分：牛顿—科特斯积分法

$$\int_a^b f(x)dx \approx (b-a)\sum_{i=0}^{n}\left(C_i^{(n)} f(x_i)\right)$$

$$C_i^{(n)} = \frac{(-1)^{n-i}}{n i!(n-i)!}\int_0^n \prod_{\substack{j=0\\ j\neq i}}^{n}(s-j)ds$$

由柯特斯系数表可以写出对应的各阶牛顿-柯特斯公式。n=4时，共有5个求积节点。

把它们代入N-C公式的一般形式，此公式称为柯特斯（Cotes）公式（又称为5点公式）。

n	$c_k^{(n)}$								
1	$\frac{1}{2}$	$\frac{1}{2}$							
2	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$						
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$					
4	$\frac{7}{90}$	$\frac{16}{45}$	$\frac{2}{15}$	$\frac{16}{45}$	$\frac{7}{90}$				
5	$\frac{19}{288}$	$\frac{25}{96}$	$\frac{25}{144}$	$\frac{25}{144}$	$\frac{25}{96}$	$\frac{19}{288}$			
6	$\frac{41}{840}$	$\frac{9}{35}$	$\frac{9}{280}$	$\frac{34}{105}$	$\frac{9}{280}$	$\frac{9}{35}$	$\frac{41}{840}$		
7	$\frac{751}{17280}$	$\frac{3577}{17280}$	$\frac{1323}{17280}$	$\frac{2989}{17280}$	$\frac{1323}{17280}$	$\frac{3577}{17280}$	$\frac{750}{17280}$		
8	$\frac{989}{28350}$	$\frac{5888}{28350}$	$\frac{-928}{28350}$	$\frac{10496}{283\ 50}$	$\frac{-45440}{28350}$	$\frac{10496}{28350}$	$\frac{-928}{28350}$	$\frac{5888}{28350}$	$\frac{989}{28350}$

$$\int_a^b f(x)dx \approx (b-a)\left(\frac{7f(x_0)}{90}+\frac{16f(x_1)}{45}+\frac{2f(x_2)}{15}+\frac{16f(x_3)}{45}+\frac{7f(x_4)}{90}\right)$$

4. 数值积分：牛顿—科特斯积分法

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

New_Cotes.m

```
function output = New_Cotes(f,a,b,n)
% f为要求输入的积分函数，a为积分下限，b为积分上限，n为求积区间等分数；
% 输入示例：
% syms x; f =exp(x); 或 f = x.*x.*x-2.*x.*x+7.*x-5; a=0;b=10;
% 调用格式[Ck,Ak,int_res] = New_Cotes(f,a,b,n)
% Ck——科特斯系数；Ak——求积系数；y——牛顿-科特斯数值积分结果
%% 此处省略输入参数的判断......

%% 计算科特斯系数
for i = 0:n
    Ck(i+1) = (-1)^(n-i)/factorial(i)/factorial(n-i)/n*integral(@(t)intfun(t,n,i),0,n);
end

%% Cotes公式计算
h = (b-a)/n;  %计算等分区间步长
output.nodeVal = subs(f,'x',a + (0:n)*h);  %节点处的函数值
output.Ak = (b-a)*Ck;  %求积系数
%牛顿-科特斯数值积分结果
output.int_result = double(sum(output.Ak.*output.nodeVal));

%% 嵌套函数：求科特斯系数
function fc = intfun(t,n,k)
fc = 1;
for i = [0:k-1,k+1:n]
    fc = fc.*(t-i);
end
```

例4： $\int_0^{3\pi} e^{-0.5x}\sin(x+\frac{\pi}{6})dx$

```
>> syms x;
>> fh = exp(-0.5*x)*sin(x+pi/6);
>> a = 0;
>> b = 3*pi;
>> output = New_Cotes(fh,a,b,15)
output =
  包含以下字段的 struct:
      nodeVal: [1×16 sym]
           Ak: [1×16 double]
   int_result: 0.900840570684864
>> S = vpa(int(fh,a,b),15) %精确值
S =
   0.900840787818886
```

5. 微分方程组：4阶龙格库塔公式

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

- 龙格-库塔（Runge-Kutta）法，简称R-K方法，是一种高阶显式一步法，而且不需要计算导数。
- 间接使用泰勒展开式的方法：用$f(x,y)$在一些点上函数值的线性组合代替$y(x)$的各阶导数，构造y_{n+1}的表达式；
- 比较这个表达式与$y(x_{n+1})$在$x=x_n$处泰勒展开式，确定y_{n+1}的表达式中的系数，使y_{n+1}的表达式与$y(x_{n+1})$泰勒展开式前面的若干项相等，从而具有对应泰勒展开式的精度阶次，这就是龙格-库塔法的主要思想。
- p级龙格—库塔公式的一般形式为：$y_{n+1}=y_n+h\sum_{i=1}^{p}(c_i k_i)\quad n=1,2,3,\cdots$ 其中 $k_1=f(x_n,y_n)$

$$k_i=f(x_n+a_i h,y_n+h\sum_{j=1}^{i-1}b_{ij}k_j)\qquad i=2,3,\cdots,p$$

上式中c_i、a_i、b_{ij}是与具体常微分方程初值问题以及n、h无关的常数。k_i是$f(x,y)$在某些点处函数值，c_i是在线性组合求y_{n+1}时k_i的“权”，a_i和b_{ij}用来确定k_i在$f(x,y)$上的位置。

5．微分方程组：4阶龙格库塔公式

- 标准（经典）龙格-库塔公式为：

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \\ k_1 = f\left(x_n, y_n\right) \\ k_2 = f\left(x_n + \dfrac{h}{2}, y_n + \dfrac{h}{2}k_1\right) \\ k_3 = f\left(x_n + \dfrac{h}{2}, y_n + \dfrac{h}{2}k_2\right) \\ k_4 = f\left(x_n + h, y_n + hk_3\right) \end{cases}$$

- 考察常微分方程组的一般情形：

$$\begin{cases} y' = f\left(t, y, z\right), \quad y\left(t_0\right) = y_0 \\ z' = g\left(t, y, z\right), \quad z\left(t_0\right) = z_0 \end{cases}$$

- 这时四阶龙格-库塔公式具有形式

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right) \\ z_{n+1} = z_n + \dfrac{h}{6}\left(L_1 + 2L_2 + 2L_3 + L_4\right) \end{cases}$$

5. 微分方程组：4阶龙格库塔公式

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = f(x_n, y_n) \\ k_2 = f\left(x_n + \dfrac{h}{2}, y_n + \dfrac{h}{2}k_1\right) \\ k_3 = f\left(x_n + \dfrac{h}{2}, y_n + \dfrac{h}{2}k_2\right) \\ k_4 = f(x_n + h, y_n + hk_3) \end{cases}$$

$$\Rightarrow$$

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ z_{n+1} = z_n + \dfrac{h}{6}(L_1 + 2L_2 + 2L_3 + L_4) \end{cases}$$

其中

$$K_1 = f(t_n, y_n, z_n), \quad L_1 = g(t_n, y_n, z_n),$$

$$K_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_1, z_n + \frac{h}{2}L_1\right), \quad L_2 = g\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_1, z_n + \frac{h}{2}L_1\right)$$

$$K_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_2, z_n + \frac{h}{2}L_2\right), \quad L_3 = g\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_2, z_n + \frac{h}{2}L_2\right),$$

$$K_4 = f(t_n + h, y_n + hK_3, z_n + hL_3), \quad L_4 = g(t_n + h, y_n + hK_3, z_n + hL_3).$$

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

5. 微分方程组：4阶龙格库塔公式

marunge4s.m

```matlab
function varargout = marunge4s(dyfun,xspan,y0,h)
% 四阶龙格库塔求解带初值问题的微分方程（组）或高阶微分方程
% [x,y] = marunge4s(dyfun,xspan,y0,h)或sol = marunge4s(dyfun,xspan,y0,h)
% dyfun:求解微分方程的M文件或匿名函数
% xspan：求解区间，y0：带有初值问题，h：求解步长，缺省为0.01

    %% 输入参数的判断
    if nargin < 3
        error('此方法求解带有初值问题的微分方程...')
    elseif nargin > 4
        error('输入参数个数太多...')
    else  %输入参数个数为3个，默认缺省h = 0.01
        h = 0.01;
    end

    %% 变量的初始化
    t = xspan(1):h:xspan(2); %根据步长划分区间
    n = length(t); %统计变量t的数量，用于循环求解对应y的值
    % 根据初值数量和区间个数初始化y，行数表示微分方程个数
    y = zeros(length(y0),n);
    y(:,1) = y0(:); %第一列为初值

    %% 循环求解，使用4阶龙格库塔公式
    for i = 2:n
        k1 = dyfun(t(i-1),y(:,i-1));
        k2 = dyfun(t(i-1)+h/2,y(:,i-1)+h/2*k1);
        %feval是计算函数，也可使用此函数计算函数值
        k3 = feval(dyfun, t(i-1) + h/2, y(:,i-1) + h/2*k2);
        k4 = feval(dyfun, t(i-1) + h, y(:,i-1) + h*k3);
        y(:,i) = y(:,i-1) + h/6*(k1 + 2*k2 + 3*k3 + k4);
    end

    %% 可变输出参数的判断
    if nargout == 2
        varargout{1} = t;
        varargout{2} = y;
    elseif nargout > 2
        error('输出参数个数过多...')
    else %无输出参数或一个输出参数
        varargout{1}.t = t;
        varargout{1}.y = y;
    end

    %% 数值解可视化
    leg = {};
    for i = 1:length(y0)
        plot(t,y(i,:))
        hold on
        leg{i} = strcat('y_',num2str(i),'(t)');
    end
    hold off; grid on; xlabel('t'); ylabel('y')
    legend(leg,'Location','best'); legend('boxoff')
    title('微分方程数值解曲线')
end
```

5. 微分方程组：4阶龙格库塔公式

例5：求解带有初值问题的二阶微分方程，求解区间为[0,60]

$$\begin{cases} y'' = 5\left(1-y^2\right)y' - y \\ y(0)=1,\ y'(0)=2 \end{cases} \Rightarrow \begin{cases} y_1' = y_2 \\ y_2' = 5\left(1-y_1^2\right)y_2 - y_1 \\ y_1(0)=1,\ y_2(0)=2 \end{cases}$$

%定义微分方程组函数文件

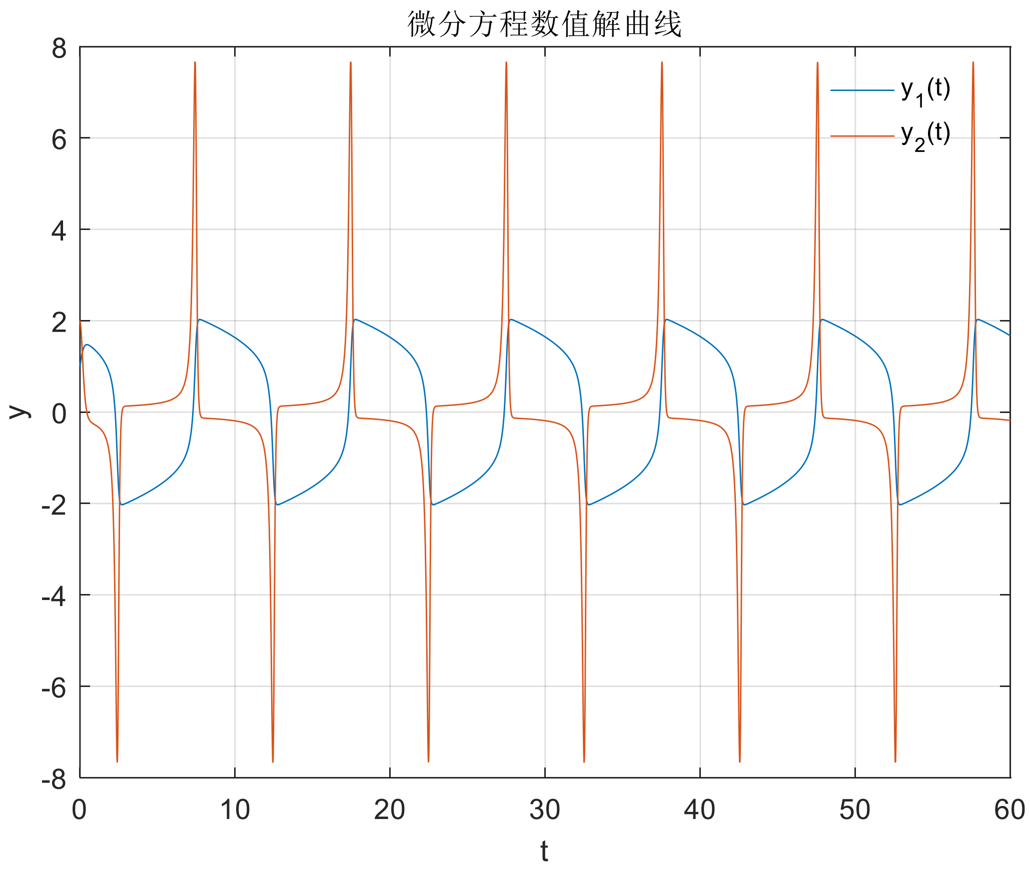
```
function dy = dyfun(t,y)
  dy = [y(2);5*(1-y(1)^2)*y(2)-y(1)];
end
```

%两个输出参数的调用，观察变量空间

```
>> [x,y] = marunge4s(@dyfun,[0,60],[1;2],0.02);
```

%一个输出参数的调用，观察变量空间，缺省步长

```
>> sol = marunge4s(@dyfun,[0,60],[1;2]);
```

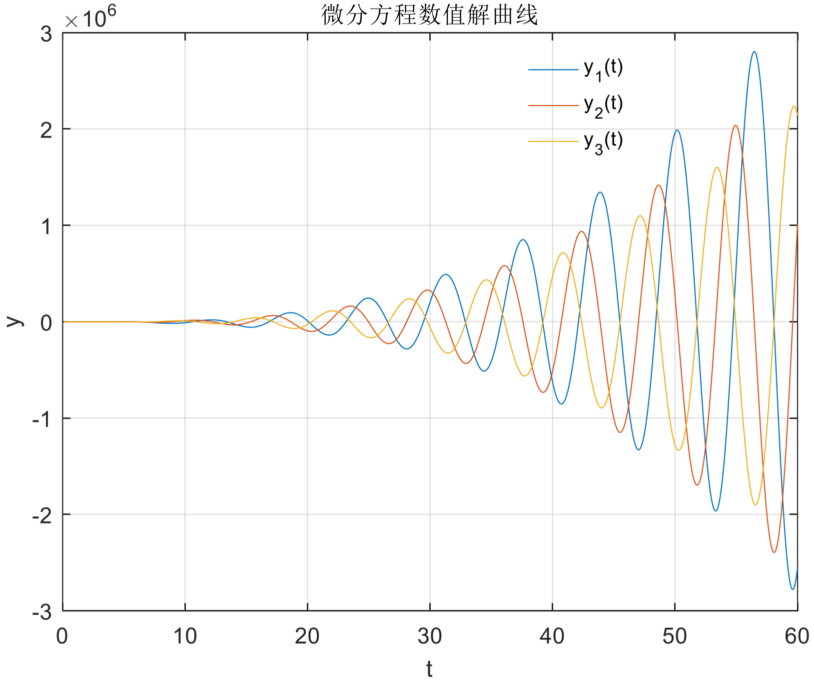


5. 微分方程组：4阶龙格库塔公式

例6：求解带有初值问题的三阶微分方程，求解区间为[0,60]

$$\begin{cases} y''' = \dfrac{y''}{t} - \dfrac{3y'}{t^2} + \dfrac{2y}{t^3} + 9t^3 \sin t \\ y'(0.1) = 1,\ y''(0.1) = 1,\ y'''(0.1) = 1 \end{cases} \Rightarrow \begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = \dfrac{y_3}{t} - \dfrac{3y_2}{t^2} + \dfrac{2y_1}{t^3} + 9t^3 \sin t \\ y_1(0.1) = 1,\ y_2(0.1) = 1,\ y_3(0.1) = 1 \end{cases}$$

```
%定义微分方程组的函数文件
function dy = dyfun2(t,y)
  dy = [y(2);y(3);y(3)/t-3*y(2)/t^2+2*y(1)/t^3+9*t^3*sin(t)];
end

%命令窗口调用
>> sol = marunge4s(@dyfun2,[0.1,60],[1;1;1]);
```



信阳师范学院 数学与统计学院

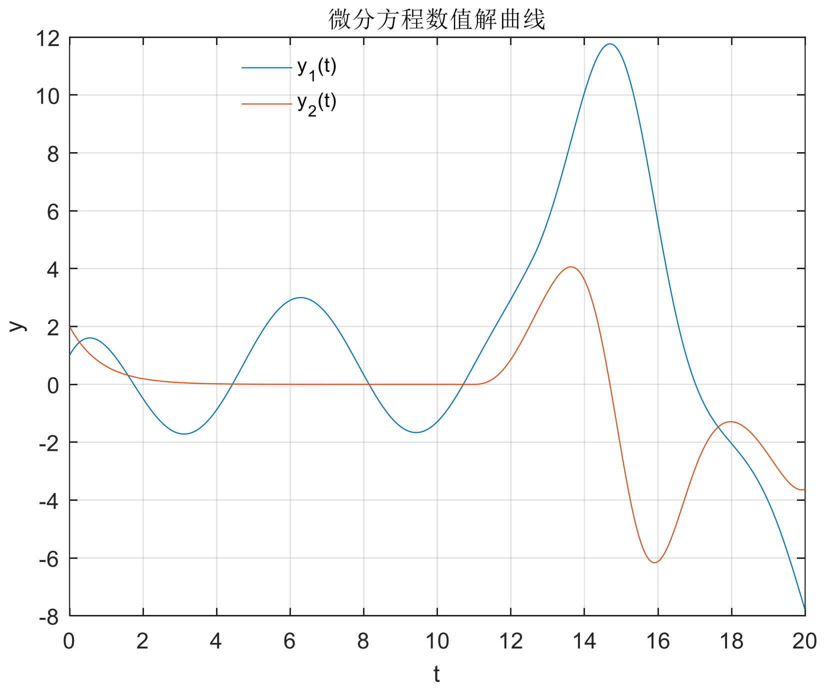
例7：求解带有初值问题的微分方程组

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_2 - f(t) \\ y_1 g(t) - y_2 \end{bmatrix}, \quad \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$f(t) = \begin{cases} 2\sin t & t < 4\pi \\ 0 & t \geq 4\pi \end{cases}, \quad g(t) = \begin{cases} 0 & t < 7\pi/2 \\ \cos t & t \geq 7\pi/2 \end{cases}$$

```
% 定义函数文件
function dy = DyDxNestedFun(t,y)
   ft = 2*sin(t).*(t<4*pi) + 0.*(t>=4*pi);   %定义分段函数f(t)
   gt = cos(t).*(t>=3.5*pi) + 0.*(t<3.5*pi);   %定义分段函数g(t)
   dy = [y(2) - ft; y(1)*gt-y(2)];   %微分方程组
end
```

```
>> sol = marunge4s(@DyDxNestedFun,[0,20],[1;2])
sol =
  包含以下字段的 struct:
    x: [1×2001 double]
    y: [2×2001 double]
```




EDUCATION
BUSINESS
01
IDEA
02
LEARNING
03
CREATIVE
INSPIRATION
PHOTO BOOK

感谢聆听