《数值逼近》实验题目

第一部分：算法实现样例（1~7 题每人至少做四个，题目见选题方案）

插值法：

1、利用 Lagrange 插值多项式验证 Runge 现象。将区间$[-1,1]$十等分，求函数

$f(x)=\dfrac{1}{1+25x^2}$ 的 Lagrange 插值多项式在 $x=0.9$ 处的值，并与 $f(0.9)$ 比较。

C 语言程序清单：

```
//include "stdio.h"
//****************************************
//程 序 名: Runge.
//程序功能:验证 Runge 现象
//****************************************
void main()
{ int i,j;
  float x[11],y[11],xx=0.9,yy=0,temp;

  for(i=0;i<=10;i++)
   { x[i] = -1.0 + 0.2*i;
     y[i] = 1/(1+25*x[i]*x[i]);
   }

  for(i=0;i<=10;i++)
   { temp=y[i];
     for(j=0;j<=10;j++)
       if(j!=i) temp=temp*(xx-x[j])/(x[i]-x[j]);
     yy=yy+temp;
   }

  printf("\n插值多项式在 x=%f 点的值为%f,",xx,yy);
  printf("而 f(%f)=%f\n",xx,1/(1+25*xx*xx));
}
```

Matlab 程序清单：

```
x = -1:0.2:1;   y=1./(1+25*x.*x);   xx=0.9;   yy=0;
```

```
for  i =1:11
   temp =y(i);
   for j =1:11
      if  j~=i
         temp =temp * (xx -x(j)) /(x(i) -x(j));
      end
   end
   yy =yy + temp;
end
fprintf('\n 插值多项式在 x = %f 点的值为%f, ',xx,yy);
fprintf('而 f(%f) = %f \n',xx,1 /(1 +25 * xx * xx));
```

2、天安门广场升旗的时间是日出的时刻，而降旗的时间是日落时分。根据天安门广场管理委员会的公告，某年 10 月份升降旗的时间如下：

日期	1	15	22
升旗	6:09	6:23	6:31
降旗	17:58	17:36	17:26

请根据上述数据构造 Newton 插值多项式，并计算当年 10 月 8 日北京市的日照时长。

C 语言程序清单：

```
#include "stdio.h"
#include "math.h"
//***********************************
//程序名:NewtonInterpolation
程序功能:利用 Newton 插值多项式计算北京某年 10 月的日照时长
//***********************************
#define n 3       //节点个数

void main( )
{  int i,j,xx,x[3] = {1,15,22};
   float y[3] = {17 -6 +(58 -9) /60.0,17 -6 +(36 -23) /60.0,17 -6 +(26 -31) /60.0};
   float N[3][3], yy,temp;

   for(i =0;i <n;i + +)
      N[i][0] =y[i];          //0 阶差商
   for(j =1;j <n;j + +)
      for(i =j;i <n;i + +)
         N[i][j] =(N[i][j -1] -N[i -1][j -1]) /(x[i] -x[i -j]); //构造差商表
```

```
  xx = 8;   yy = 0.0;   temp = 1.0;
  for(i = 0;i < n;i + +)
    { yy = yy + N[i][i] * temp;    //Newton 型插值多项式的计算
      temp = temp * (xx - x[i]);
    }
  printf("\n 这年 10 月%d 日北京日照时长为%7.4f 小时.\n",xx,yy);
  printf("即%d 小时%3.0f 分.\n",(int)floor(yy),60 * fmod(yy,1.0));
}
```

Matlab 程序清单：

```
n = 3; x = [1,15,22];
y = [17 - 6 + (58 - 9) /60.0,17 - 6 + (36 - 23) /60.0,17 - 6 + (26 - 31) /60.0];
N = zeros(3);

for  i = 1:n
   N(i,1) = y(i);
end
for  j = 2:n
   for i = j:n
      N(i,j) = (N(i,j - 1) - N(i - 1,j - 1)) /(x(i) - x(i - j + 1));
   end
end
xx = 8;   yy = 0;   temp = 1;
for i = 1:n
   yy = yy + N(i,i) * temp;
   temp = temp * (xx - x(i));
end
fprintf('这年 10 月 8 日北京日照时长为%7.4f 小时.\n',yy);
fprintf('    即%d 小时%2.0f 分.\n',fix(yy),60 * mod(yy,1.0));
```

数值积分：

3、将区间 $[0,1]$ 四等分，分别用复化梯形公式和复化 Simpson 公式计算定积分 $I=\int_0^1 \frac{4}{1+x^2}dx$ 的近似值。

C 语言程序清单：

```
#include "stdio.h"
#define  f(x)  4.0/(1+(x)*(x))
//************************************
//程序名:  CotesianIntegral
//程序功能:用复化梯形公式和复化 Simpson 公式计算积分
//************************************

void main()
{  float h=(1.0-0)/4, temp, xk, yk, xkh, ykh, xk1, yk1;
   int i;
   temp=f(0);   xk=0;
   for(i=1;i<4;i++)
     {  xk=xk+h;
        temp=temp + 2*f(xk);
     }
   temp=temp+f(1);
   temp=temp*h/2;       //使用化简后的公式
   printf("\n 复化梯形公式计算的结果: %f",temp);

   temp=0;    h=(1.0-0)/2;      //对复化 Simpson 公式分成 2 个小区间
   xk=0;      yk=f(0);           //xk——x[k]
   for(i=0;i<2;i++)
     {  xkh=xk+h/2; ykh=f(xkh);      //xkh——x[k+1/2]
        xk1=xk+h;    yk1=f(xk1);      //xk1——x[k+1]
        temp=temp + h*(yk+4*ykh+yk1)/6;  //加上每个小区间上的面积
        xk=xk1;      yk=yk1;         //右端点是下一个小区间的左端点
     }
   printf("\n 复化 Simpson 公式计算的结果: %f \n",temp);
}
```

Matlab 程序清单：

```
f=inline('4./(1+x.^2)');
h=(1-0)/4;
temp=f(0);   xk=0;
for  i=1:3
     xk=xk+h;
     temp=temp + 2*f(xk);
end
temp=temp+f(1);
```

```
temp = temp * h/2;      % 使用化简后的公式
fprintf('\n 复化梯形公式计算的结果:%f',temp);

temp = 0;
h = (1 - 0)/2;      % 对复化 Simpson 公式分成 2 个小区间
xk = 0;    yk = f(0);     % xk——x[k]
for  i = 0:1
     xkh = xk + h/2;    ykh = f(xkh);     % xkh——x[k+1/2]
     xk1 = xk + h;      yk1 = f(xk1);      % xk1——x[k+1]
     temp = temp + h*(yk + 4*ykh + yk1)/6; % 加上每个小区间上的面积
     xk = xk1;   yk = yk1;     % 右端点是下一个小区间的左端点
end
fprintf('\n 复化 Simpson 公式计算的结果:%f \n',temp);
```

常微分方程数值解法：

4、取步长 $h = 0.1$，分别用 Euler 方法、改进 Euler 方法和经典四阶 Runge-Kutta 方法求解初值问题

$$\begin{cases} y' = y - \dfrac{2x}{y}, \quad 0 < x < 1 \\ y(0) = 1 \end{cases}$$

并将计算结果与精确解 $y = \sqrt{1+2x}$ 相比较。

C 语言程序清单：

```
#include "stdio.h"
#include "math.h"

//*****************************************/
// 程序名: ODE
// 程序功能:用 Euler 方法、改进 Euler 方法和 4 阶 Runge - Kutta 方法
//          求解常微分方程初值问题
/*****************************************/

#define f(x,y)  ((y) - 2*(x)/(y))
#define y(x)    sqrt(1 + 2*(x))
```

```
void main()
{   float a=0, b=1.0f, h=0.1f, y0=1.0f,x, ye, yp, ym, k1, k2, k3, k4, yr, yx;

    printf("\n      精确解          Euler 方法         ");
    printf("改进 Euler 方法        4 阶 Runge-Kutta 方法");
    printf("\n x       y          ye[k]   |ye[k]-y|  ");
    printf("ym[k]  |ym[k]-y|   yr[k]    |yr[k]-y|\n");
    printf("%3.1f  %8.6f  %8.6f  %8.6f     ",a,y0,y0,0);
    printf("%8.6f  %8.6f  %8.6f  %8.6f\n",y0,0,y0,0.0);

    x=a;
    ye=y0;              //Euler 方法的初值
    ym=y0;              //改进的 Euler 方法的初值
    yr=y0;              //4 阶 Runge-Kutta 方法的初值

    while (x<b)
    {   ye=ye+h*f(x,ye);        //Euler 公式在下一分点的值

        yp=ym+h*f(x,ym);        //改进 Euler 公式在下一分点的预报值

        ym=ym+h/2*(f(x,ym)+f(x+h,yp));  //改进 Euler 公式在下一分点的值

        k1=f(x,yr);
        k2=f(x+h/2,yr+h/2*k1);
        k3=f(x+h/2,yr+h/2*k2);
        k4=f(x+h,yr+h*k3);
        yr=yr+h/6*(k1+2*k2+2*k3+k4);  //4 阶 Runge-Kutta 方法在下一分点的值

        x=x+h;                  //下一分点 x 坐标
        yx=y(x);                //下一分点的精确解

        printf("%3.1f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f %8.6f\n"
            ,x,yx,ye,fabs(ye-yx),ym,fabs(ym-yx),yr,fabs(yr-yx));
    }
}
```

Matlab 程序清单：

```
f=inline('y-2*x/y','x','y');
y=inline('sqrt(1+2*x)');

a=0; b=1; h=0.1; y0=1;

fprintf('\n         精确解                 Euler 方法          ');
```

```
fprintf('改进 Euler 方法                    4 阶 Runge - Kutta 方法');
fprintf('\n x         y           ye[k]      |ye[k] - y |     ym[k]');
fprintf('  |ym[k] - y |                   yr[k]       |yr[k] - y |\n');
fprintf('%3.1f  %8.6f  %8.6f  %8.6f        ',a,y0,y0,0);
fprintf('%8.6f  %8.6f    %8.6f  %8.6f\n',y0,0,y0,0.0);

x = a;
ye = y0;            % Euler 方法的初值
ym = y0;            % 改进的 Euler 方法的初值
yr = y0;    % 4 阶 Runge - Kutta 方法的初值

while (x < b)
   ye = ye + h * f(x,ye);       % Euler 公式在下一分点的值

   yp = ym + h * f(x,ym);       % 改进 Euler 公式在下一分点的预报值
   ym = ym + h/2 * (f(x,ym) + f(x + h,yp));     % 改进 Euler 公式在下一分点的值

   k1 = f(x,yr);
   k2 = f(x + h/2,yr + h/2 * k1);
   k3 = f(x + h/2,yr + h/2 * k2);
   k4 = f(x + h,yr + h * k3);
   yr = yr + h/6 * (k1 + 2 * k2 + 2 * k3 + k4);     % 4 阶 Runge - Kutta 方法在下一分点的值

   x = x + h;          % 下一分点 x 坐标
   yx = y(x);          % 下一分点的精确解

   fprintf('%3.1f  %8.6f    %8.6f    %8.6f        ',x,yx,ye,abs(ye - yx));
   fprintf('%8.6f  %8.6f    %8.6f    %8.6f\n',ym,abs(ym - yx),yr,abs(yr - yx));
end
```

方程求根：

5、用二分法求非线性方程 $f(x)=0$ 在区间 $[a,b]$ 内的根。取 $f(x)=(x+1)^2-\sin x-3$，

$a=0, b=1$，精度要求为 1×10^{-2}，最多迭代 100 次。

C 语言程序清单：

```
#include "stdio.h"
#include "math.h"
float f(float x)
{    return (x+1)*(x+1)-sin(x)-3;  }

float Bisection(float a,float b,float epsilon,int N)
//********************************
// 程序名:Bisection
// 程序功能:实现使用二分法求解非线性方程
// 输入:    初始区间[a,b],误差精度 tepsilon 和最大迭代次数 N
// 输出:    方程的近似解
//********************************
{ int n=1;
  float p,fa,fp;
  printf("\n k    a(k)      b(k)");
  printf("\n %2d    %f    %f",0,a,b);
  fa=f(a);
if(fa*f(b)>0) {printf("无法求解.");exit(0);}
  while(n<N)
  { p=(a+b)/2;        fp=f(p);
    if(fp==0 || fabs(b-a)<epsilon)
        return p;
    printf("\n %2d    %f    %f    %f",n,a,b,p);
    n++;
    if(fa*fp>0)
    {   a=p;     fa=fp;}
    else
         b=p;
  }
  printf("\n \n"%d 次迭代后未达到精度要求.\n",N);
  exit(0);
}
void main()
{
  int N;
  float a,b,abtol;
  printf("\na=");             scanf("%f",&a);
  printf("\nb=");             scanf("%f",&b);
  printf("\nabtol=");        scanf("%f",&abtol);
  printf("\nN=");             scanf("%d",&N);
  printf("\n 结果:%f \n",Bisection(a,b,abtol,N));
}
```

Matlab 程序清单：

```
% ****** fun.m
function [y]=fun(x)
y=(x+1)^2-sin(x)-3;
% ****** Bisection.m
function [y]=Bisection(a,b,abtol,N)
n=1; fa=fun(a);
  if ya*yb>0
   disp('注意:ya*yb>0,请重新调整区间端点 a 和 b.'),return
  end
  fprintf('\n   k      a(k)      b(k)         p');
fprintf('\n %2d    %f       %f',0,a,b);
while n<N
   p=(a+b)/2;     fp=fun(p);
   if ((fp==0)|(abs(b-a))<abtol)
       y=p;    return;
   end
   fprintf('\n %2d     %f     %f     %f',n,a,b,p);
   n=n+1;
   if fa*fp>0
       a=p;    fa=fp;
   else
       b=p;
   end
end
fprintf('\n\n%d 次迭代后未达到精度要求.\n',N);return
% ******   test.m
a=input('a=');            b=input('b=');
abtol=input('abtol=');    N=input('N=');
fprintf('\n 结果:%f',bisection(a,b,abtol,N));
```

n=1; fa=fun(a); fb=fun(b);
if fa*fb>0
disp('注意：fa*fb>0,请新调整区间端点 a 和 b.'), return

Bisection(a,b,abtol,N)

6、用迭代法求非线性方程 $(x+1)^2-\sin x-3=0$ 的根。取迭代函数

$\varphi(x)=0.5*(3+\sin x-1-x^2)$，精度要求为 1×10^{-2}，最多迭代 100 次。

C 语言程序清单：

```
#include "stdio.h"
#include "math.h"
//*******************************************
// 程序数名:Iteration
// 程序功能:实现使用简单迭代法求解非线性方程
//*******************************************
#define phi(x) 0.5*(3+sin((x))-1-(x)*(x))      //迭代函数
void main()
{ int n=1,N;
  float x,x0,del;
  printf("x0 =");       scanf("%f",&x0);
  printf("\ndel =:");    scanf("%f",&del);
  printf("\nN =");       scanf("%d",&N);
```

```
  printf("\n k     x(k) ");
  printf("\n %2d %f ",0,x0);
  while(n<N)
  {  x=phi(x0);
     if(fabs(x-x0)<del)
      {  printf("\n\n 近似解 = %f\n",x);
         return ;
      }
     printf("\n %2d     %f  ",n,x0);
     n=n+1;    x0=x;
  }
  printf("\n\n%d 次迭代后未达到精度要求.\n",N);
}
```

Matlab 程序清单：

```
phi=inline('0.5*(3+sin(x)-1-x^2)');     % 迭代函数
x0=input('x0='); del=input('del=');
N=input('N=');     n=1;
fprintf('\n k      x(k)    ');
fprintf('\n %2d        %f  ',0,x0);
while n<N
    x=phi(x0);
    if abs(x-x0)<del
       fprintf('\n\n 近似解 = %f\n',x);
       return ;
    end
    fprintf('\n %2d      %f   ',n,x);
    n=n+1;    x0=x;
end
fprintf('\n\n%d 次迭代后未达到精度要求.\n',N);
```

7、用 Newton 迭代法求非线性方程 $xe^x-1=0$ 的根。取初值 $x_0=1$，精度要求为 1×10^{-5}，最多迭代 100 次。

C 语言程序清单：

```
#include "stdio.h"
#include "math.h"
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
/*   程序名:NewtonIteration
/*   程序功能:实现使用 Newton 迭代法求解非线性方程
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
#define f(x) (x) * exp((x)) -1              //f(x)
#define df(x)exp((x)) * (1 +(x))                //f'(x)
void main()
  { int n =1,N;
    float x0,x1,F0,dF0,F1,dF1,del;
    printf("x0 =");              scanf("%f",&x0);
    printf("\ndel =:");          scanf("%f",&del);
    printf("\nN =");             scanf("%d",&N);
    printf("\n     k x(k)");
    printf("\n %2d    %f   ",0,x0);
    F0 = f(x0);                  dF0 = df(x0);
    while(n <N)
    {
    if(dF0 = =0)
      {  printf("导数为0,迭代无法继续进行.");
         return ;
      }
    x1 = x0 - F0 /dF0;
    F1 = f(x1);       dF1 = df(x1);
    if((fabs(x1 -x0) <del) || fabs(F1) <del)
      {  printf("\n \n 结果:%f \n",x1);
         return ;
      }
    printf("\n %2d    %f   ",n,x1);
    n =n +1;      x0 =x1;
    F0 =F1;    dF0 =dF1;
    }
  printf("\n \n%d 次迭代后未达到精度要求. \n",N);
}
```

Matlab 程序清单：

```
f = inline('(x) * exp((x)) - 1')          %f(x)
df = inline('exp(x) * (1 + x)')          %f'(x)
n = 1
x0 = input('x0 = ');
del = input('del = ');
N = input('N = ');
fprintf('\n k    x(k)');
fprintf('\n %2d    %f  ',0,x0);
F0 = f(x0);          dF0 = df(x0);
while n < N
    if dF0 = = 0
    fprintf('导数为 0,迭代无法继续进行 .');
    return ;
    end
    x1 = x0 - F0 /dF0;
    F1 = f(x1);       dF1 = df(x1);
    if ((abs(x1 - x0) < del) | abs(F1) < del)
    fprintf('\n \n 结果: %f \n',x1);
    return ;
    end
    fprintf('\n %2d  %f  ',n,x1);
    n = n + 1;     x0 = x1;
    F0 = F1;    dF0 = dF1;
end
fprintf('\n \n%d 次迭代后未达到精度要求 .\n',N);
```

第二部分：算法设计与实现（8~9 题每人至少做一个，题目见选题方案）

8、已知函数 $f(x)=e^{-x}$ 在 10 个节点上的函数值和导数值，如下表：

x	0.10	0.15	0.30	0.45	0.55
$f(x)$	0.904837	0.860708	0.740818	0.637628	0.576950
$f'(x)$	- 0.904837	-0.860708	-0.740818	-0.637628	-0.576950
x	0.60	0.70	0.85	0.90	1.00
$f(x)$	0.548812	0.496585	0.427415	0.406570	0.367879
$f'(x)$	-0.548812	-0.496585	-0.427415	-0.406570	-0.367879

用熟悉的计算机语言编写 Hermite 插值法通用程序，计算 $f(0.356)$ 的近似值，并与 $f(0.356)$ 比较。

9、公元 1225 年，比萨的数学家 Leonardo（即 Fibonacci（斐波那契）），1170-1250）研究了方程

$$x^3+2x^2+10x-20=0$$

得到一个根 $x^*\approx 1.368\,808\,107$，没有人知道他用什么方法得到这个值。对于这个方程，分别用下列方法：

（1）迭代法 $x_{k+1}=\dfrac{20}{x_k^2+2x_k+10}$；　（2）迭代法 $x_{k+1}=\dfrac{20-2x_k^2-x_k^3}{10}$；

（3）对（1）的 Aitken 加速方法；　（4）对（2）的 Aitken 加速方法；

（5）Newton 法

求方程的根（可取 $x_0=1$），计算到 Leonardo 所得到的准确度。计算机语言不限。