

**《数值分析》实验报告(2)**

**——线性代数方程组求解综合实验**

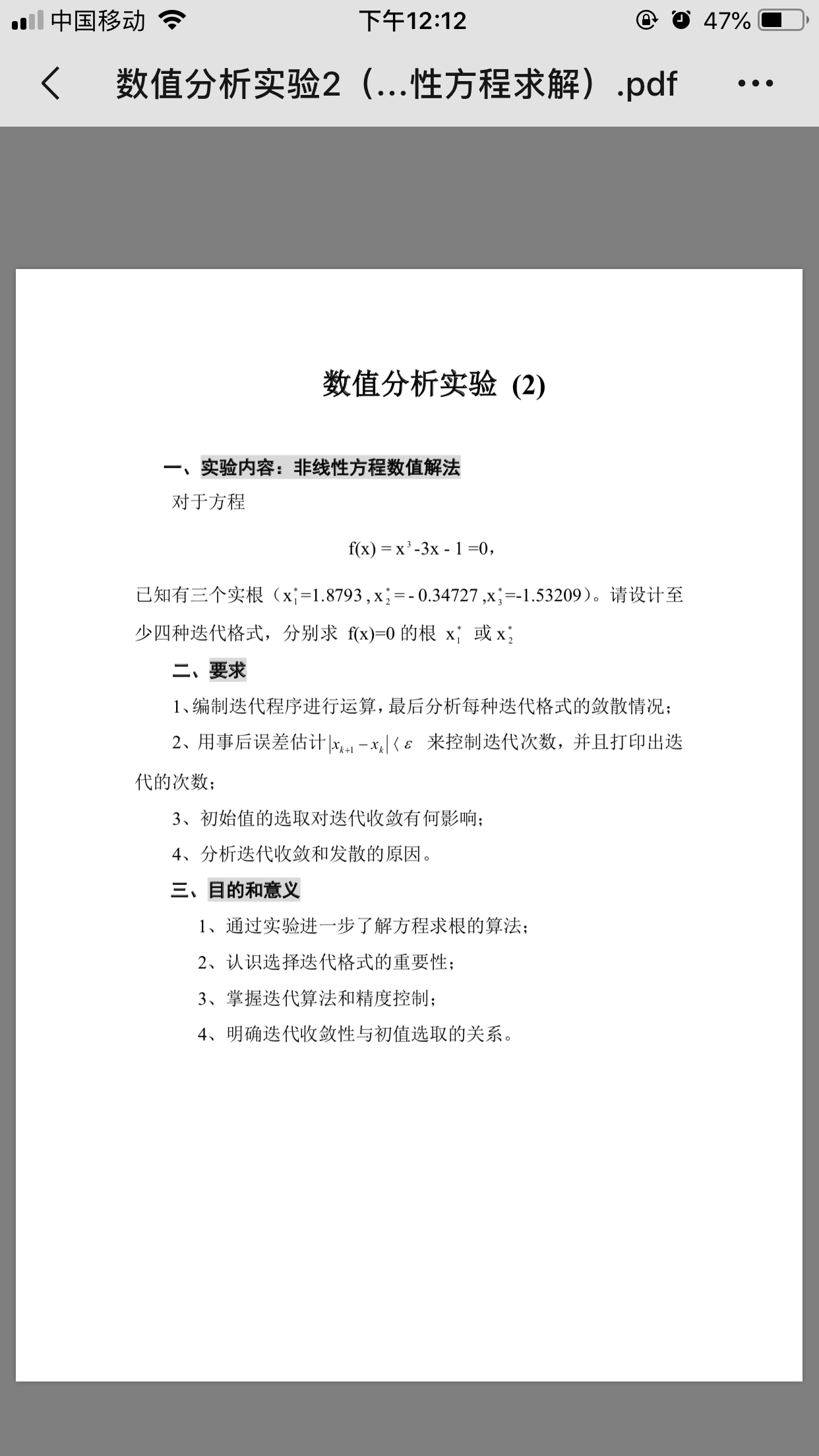
姓 名

学 号

专 业

班 级

**二Ｏ 20 年 5 月 21 日**



**代码均放置在附件中**

**更详细的数值结果也在附件中**

1. **牛顿迭代法**

**代码：**

**1.**

clc

clear

fun=inline('x^3-3\*x-1'); %原方程

dfun=inline('3\*x^2-3'); %原方程的微分

x=newton(fun,dfun,1.5) %初值1.5

**2.**

function x=newton(fun,dfun,x0,ep,N)

if nargin<5

N = 1000;

end

if nargin<4

ep = 1e-5;

end

k=0;

while k<N

k

x = x0-feval(fun,x0)/feval(dfun,x0)

if abs(x-x0)<ep

break;

end

k=k+1;

x0=x;

end

if k==N

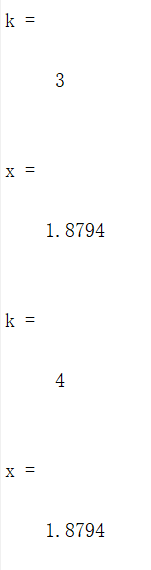
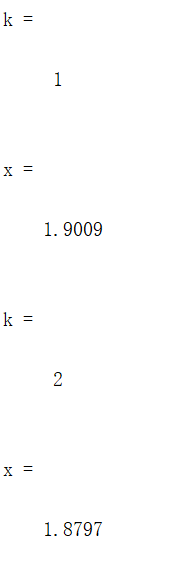
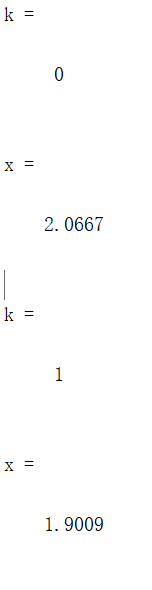
warning('已到达迭代次数上限！');

end

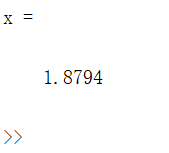
精度

初值取1.5

**实验结果：**



即经过4次迭代后最终解得：



还可以通过修改求根区间求得可能存在的更多的解。

1. **二分法**

**代码：**

**1.**

clc

clear

fun=inline('x^3-3\*x-1'); %原方程

x=cut(fun,-5,5) %求根区间

**2.**

function [c,err,yc] = cut(f,a,b,eps)

if nargin < 4

eps = 1e-5;

end

ya = feval(f,a);

yb = feval(f,b);

if yb == 0

c = b;

return

end

if ya\*yb>0

disp('(a,b)不是有根区间');

return

end

max1 = 1 + round((log(b-a)-log(eps))/log(2));

for k = 1:max1

c = (a+b)/2;

yc = feval(f,c);

fprintf('迭代第%d次\n',k)

if yc == 0

a = c;

b = c;

break;

elseif yb\*yc > 0

b = c;

yb = yc;

else

a = c;

ya = c;

end

if(b-a) < eps

break

end

end

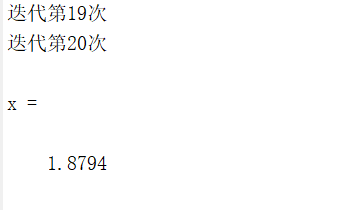
k = (a+b)/2;

c = (a+b)/2;

err = abs(b-a);

yc = feval(f,c);

**实验结果：**



还可以通过修改求根区间求得可能存在的更多的解。

1. **弦截法**

**代码：**

**1.**

clc

clear

fun=inline('x^3-3\*x-1'); %原方程

x=secant(fun,-5,5,1e-5,1000) %求根区间

**2.**

function [p1,err,k,y] = secant(f,p0,p1,eps,max1)

p0,p1,feval(f,p0),feval(f,p1),k=0,

for k = 1:max1

p2 = p1 - feval(f,p1)\*(p1-p0)/(feval(f,p1)-feval(f,p0));

err = abs(p2 - p1);

p0 = p1;

p1 = p2;

p1,err,k,y=feval(f,p1)

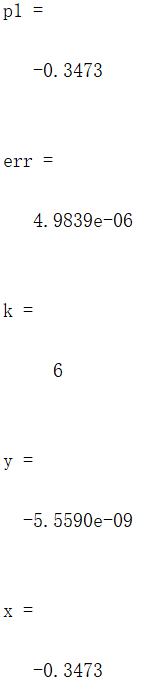
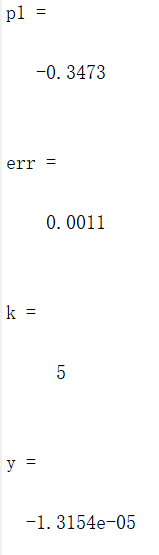
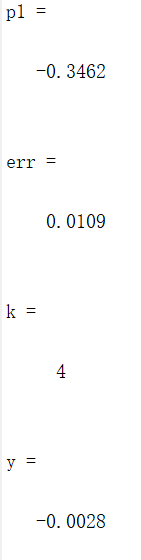
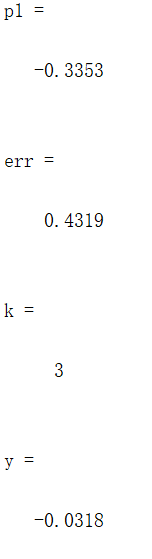
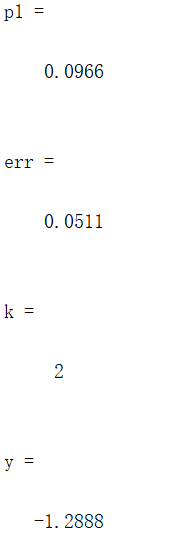
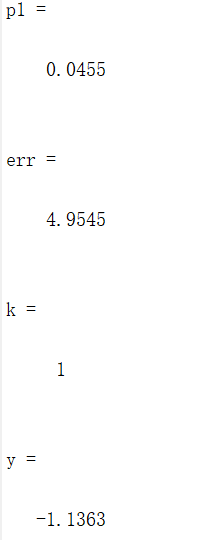
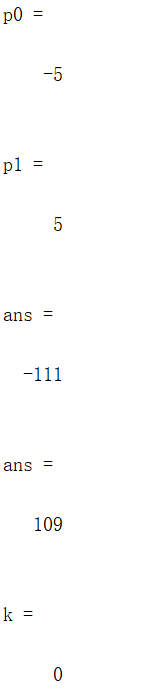
if(err < eps) | (y == 0)

break

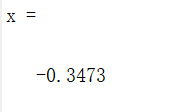
end

end

**实验结果：**



即解得：



还可以通过修改求根区间求得可能存在的更多的解。

1. **牛顿下山法**

**代码：**

**1.**

clc

clear

fun='x^3-3\*x-1'; %原方程

dfun='3\*x^2-3'; %原方程的微分

[x,y]=newton\_down(-3,fun,dfun,1e-5) %求根区间

**2.**

function [x,k]=newton\_down(x\_0,fun,dfun,eps)

%fun是原函数，fun是导函数，error是收敛误差，x0是迭代初始点

x=x\_0;

y=f(fun,x);

k=1; %标记迭代了多少次

while abs(y)>eps

d=1;

x2=x-d\*y/f(dfun,x);

while abs(f(fun,x2))>abs(f(fun,x))

d=d/2;

x2=x-d\*y/f(dfun,x);

end

x=x2;

y=f(fun,x);

k=k+1;

end

**3.**

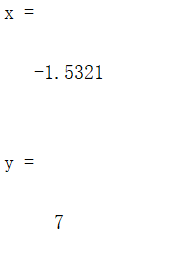
function y=f(fun,num)

x=num;

y=eval(fun);

%先定义一个函数，可以表达所有的数学函数，fun为字符串，表示函数

**实验结果：**



**实验心得：**

初始值（求根区间）的选取显然对于迭代次数和最终求得的解都有很大影响，选择合适的初始值或求根区间可以更快地求出解。

可以看出弦截法的迭代效果优于二分法，即就本例而言，弦截法的收敛速度比二分法快。（别的初值设置的不太一样没回头去修改，需要的话再调整一下初值就好）。