序号：36　　　学号：3190103782　　　姓名：孔伟杰

Q1：T4.3 用多种方法计算三次多项式的三个实数根

定义所求根函数

```
close all; clc; clear;

%函数表达式定义
p = inline('816.*(x.^3)-3835.*(x.^2)+6000.*x-3125','x');
```

（a）求精确根

```
%求函数精确解
syms x;
p = 816.*(x.^3)-3835.*(x.^2)+6000.*x-3125;
solve(p,x)
```

```
ans =

25/17
25/16
  5/3
```

由运行结果可得，

此函数的精确根为

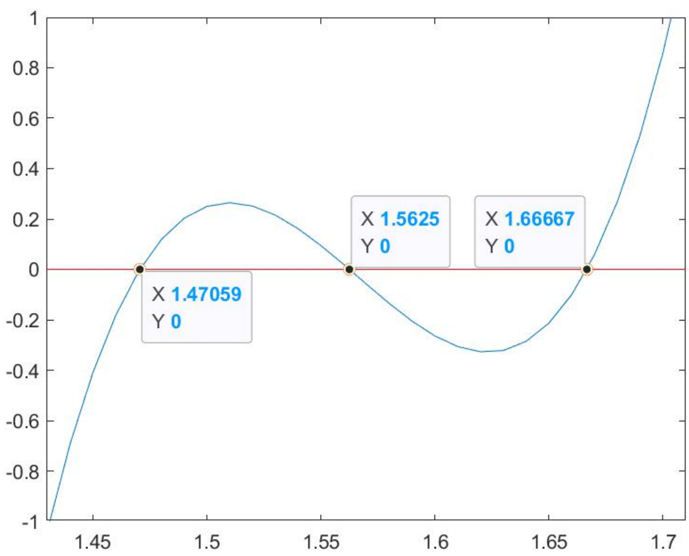
x1= 25/17 =1.4706

x2= 25/16 =1.5625

x3= 5/3　=1.6667

（b）绘制函数图形

```
%绘制函数图像
x=-20:0.01:20;
y = p(x);
figure(1);plot(x,y,'-',Ac,zeros(1,3),'o');
line([-50 50],[0 0],'color','red');%绘制p=0直线
axis([1.43 1.71 -1 1]);%限定区间范围
```



（c）牛顿法求解

```
%牛顿法
x0 = 1.5; x = x0;
pprime = inline('3*816.*(x.^2)-2*3835.*(x.^1)+6000','x');
xprev = x+1;
n1 = 0;
while abs(x - xprev) > eps*abs(x)
xprev = x;
x = x - p(x)/pprime(x);
n1 = n1 + 1;
end
x,n1

x =

   1.4706e+00


n1 =

    11
```

由运行结果可得，此函数零点的求解，若用牛顿法，则

循环次数：n = 11

求根零点：x = 1.4706

（d）割线法/弦截法求解

```
%割线法/弦截法
a = 1; b = 2;
n2 = 0;
while abs( b - a ) > eps*abs(b)
c = a;
a = b;
b = b - p(b)*((b - c)/(p(b)-p(c)));
n2 = n2 + 1;
end
b,n2
```

```
b =

   1.6667e+00


n2 =

    12
```

由运行结果可得，此函数零点的求解，若用二分法，则循环次数：n = 12

求根零点：x = 1.6667

（e）二分法求解

```
%二分法
a = 1; b = 2;
p(1),p(2)
n3 = 0;
while abs(b-a) > eps*abs(b)
x = (a + b)/2;
if sign(p(x)) == sign(p(b))
b = x;
else
a = x;
end
n3 = n3 + 1;
end
x,n3
```

```
ans =

  -144


ans =

    63


x =

   1.4706e+00


n3 =

    52
```

由运行结果可得，此函数零点的求解，若用二分法，则循环次数：n = 52

求根零点：x = 1.4706

（f）fzerotx 函数求解

```
%Zerion算法
z=fzerotx(p,[1 2])
```

```
z =

   1.6667e+00
```

由运行结果可得，运行 fzerotx(p,[1,2])函数得到一根：z = 1.6667，为三个根中最大的一个根。首先用 fzerotx 函数求根只会输出一个根，且其原理是 zeroin 算法。先进行一步割线法，在根据判断使用割线法或 IQI 算法，在进行 IQI 算法的拟合函数的根落在了 x2 和 x3 = z 之间的区域，使得算法最后收敛至最大根 x3 处。

Q2：T4.8 弦截法的反例函数验证

```
a=2;
f = inline('sign(x-a)*sqrt(abs(x-a))','x','a');
b0 =0; b = b0; c0 = 2;c = c0;
n = 0;
while abs( c - b ) > eps*abs(c)
d = b;
b = c;
c = c - f(c,a)*((c - d)/(f(c,a) - f(d,a)));
n = n + 1;
end
[a,b0,c0],n,c
```

```
ans =

     2     0     2


n =

     1


c =

     2
```

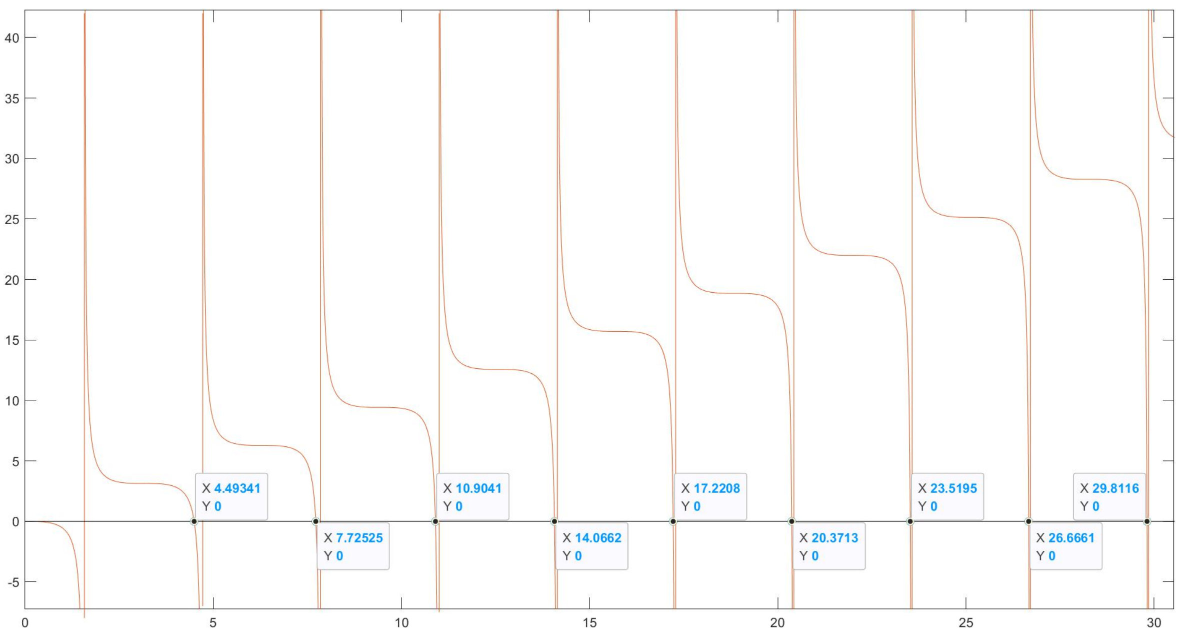
```
ans =

     2     3     4


n =

    74


c =

     2
```

此函数求根采用弦截法，若初值取值关于 x=a 对称分布或其中一初值点取 x=a 时，算法迭代次数 n=1 或 2 次，算法收敛性好。当初值点非以上两种情况，则算法迭代次数 n>70，算法收敛性差，因此对此函数用弦截法的算法效率取决于处置点的选取，稳定性低。

Q3：T4.9 求解 y=tanx 的前十个正数解

```
% Q3 4.9
a=0.0000000001;
for n=1:10
    z(n) = fzerotx('x - tan(x)',[(n - 1/(2+a)) (n + 1/(2+a))]*pi)
end
x = 0:pi/500:10.7*pi;
y = x - tan(x);
figure;plot(z,zeros(1,10),'o',x,y,'-');
line([0 10.7*pi],[0 0],'color','black');
axis([0 10.7*pi -5 40]);
```

```
z =

  列 1 至 3

   4.4934e+00   7.7253e+00   1.0904e+01

  列 4 至 6

   1.4066e+01   1.7221e+01   2.0371e+01

  列 7 至 9

   2.3519e+01   2.6666e+01   2.9812e+01

  列 10

   3.2956e+01
```



求得 x = tanx 的前十个正数解：

x1 = pi = 4.4934

x2 = 2pi = 7.7253

x3 = 3pi = 10.904

x4 = 4pi = 14.066

x5 = 5pi = 17.221

x6 = 6pi = 20.371

x7 = 7pi = 23.519

x8 = 8pi = 26.666

x9 = 9pi = 29.812

x10 = 10pi = 32.956

Q4：用牛顿法、割线法、逆二次插值发求根

```
% %Q4
p = inline('x^3 - 3*x - 1','x');
```

（1）牛顿法求根

```
%牛顿法
pprime = inline('3*x^2-3','x');
x = 2;
xprev = x+1;
n1 = 0;
while abs(x - xprev) > eps*abs(x)
xprev = x;
x = x - p(x)/pprime(x);
n1 = n1 + 1;
end
x,n1
```

```
x =

  1.8794e+00


n1 =

    5
```

由运行结果可得，此函数零点的求解，若用牛顿法，则循环次数：n = 5

求根零点：x = 1.8794

（2）割线法求根

```
%割线法
a = 2; b = 1.9;
n2 = 0;
while abs( b - a ) > eps*abs(b)
c = a;
a = b;
b = b - p(b)*((b - c)/(p(b)-p(c)));
n2 = n2 + 1;
end
b,n2
```

```
b =

  1.8794e+00


n2 =

    6
```

由运行结果可得，此函数零点的求解，若用割线法，则循环次数：n = 6

求根零点：x = 1.8794

（3）逆二次插值求根

```
%逆二次插值法
a = 1; b = 3; c = 2;
n3 = 0;
while abs(c-b) > 0.001
    x = polyinterp([p(a),p(b),p(c)],[a,b,c],0);
    a = b;
    b = c;
    c = x;
    n3 = n3 + 1;
end
x,n3
```

```
x =

  1.8794e+00


n3 =

    4
```

由运行结果可得，此函数零点的求解，若用逆二次插值法，则循环次数 n = 4

求根可得零点为 x = 1.8794

综上，对该函数求根，算法效率：逆二次插值法好于牛顿法好于割线法