1. IVT: is a continuous function on the interval [0,1]. , , therefore and thus satisfies the IVT.  
     
   For 10 correct decimal places we do:  
   , but for rounding sake this sometimes yields a 4 in the 11th decimal place depending on the starting interval, so I am going to use   
     
   , the approximate root to 10 correct places and rounded is 0.4738158381.  
     
   Now we do the same thing for the interval [0,1.2]:  
   , the approximate root to 10 correct places and rounded is 0.4738158381.  
     
   Once more for the interval [0,1.4]:  
   , the approximate root to 10 correct places and rounded is 0.4738158381.  
     
   When using for some the new intervals I was getting 0.47381583804 instead of the expected 0.47381583806 or 0.47381583807. So up to the 10th decimal place was correct across the different starting intervals but the rounding wasn’t so I wanted to make sure all the rounding was the same so I used to fix the rounding issue.
2. The backward error is Since the backward error is so small, the estimated root is close to 0 which gives confidence that the root is accurate up to 10 decimal places rounded.
3. For 10 correct decimal places we do:  
   , the approximate root to 10 correct places and rounded is 0.1669921875.  
     
   Now we do the same thing for the interval [0,1.2]:  
   , the approximate root to 10 correct places and rounded is 0.1669921875.  
     
   Once more for the interval [0,1.4]:  
   , the approximate root to 10 correct places and rounded is 0.1667968750.  
     
   For both intervals [0,1] and [0,1.2] the roots are the same but for [0,1.4] the root is different. They are only the same up to 3 decimal places.  
     
   The backward error for [0,1] and [0,1.2] is   
   The backward error for [0,1.4] is   
   The backward error for both of the points is 0 which means that accuracy of these roots must be contingent only upon the first 3 decimal places and that must be the 3 decimal places are the rounded exact root where the exact root is 1/6 (0.1666666…).
4. The main difference between and is that all the intervals for get the same rounded 10 decimal place estimated root and intervals differ. Although has different estimated roots, their backwards errors are all 0. I believe that the estimated rounded root of is correct to 10 decimal places. With since the estimated roots are different after 3 decimal places depending on the interval one can’t really tell which is truly the correct 10 decimal place estimated root, but with the backwards error being 0 for each estimated root it seems that only the correct 3 decimal places are all we need.

**bisect.m**

function xc=bisect(f,a,b,tol)

if sign(f(a))\*sign(f(b)) >= 0

error('f(a)f(b)<0 not satisfied!')

end

fa=f(a);

fb=f(b);

while (b-a)/2>tol

c=(a+b)/2;

fc=f(c);

if fc == 0

break

end

if sign(fc)\*sign(fa)<0

b=c;fb=fc;

else

a=c;fa=fc;

end

end

xc=(a+b)/2;

**MatLab Session**

f=@(x)(432\*x.^4+72\*x.^2+16\*x+5)\*exp(1)-8\*exp(6\*x);

xc=bisect (f,0,1,0.5\*10.^(-11))

xc =

0.473815838067821

xc=bisect (f,0,1.2,0.5\*10.^(-11))

xc =

0.473815838071459

xc=bisect (f,0,1.4,0.5\*10.^(-11))

xc =

0.473815838069640

abs(f(0.4738158381))

ans =

2.932125653387629e-09

f=@(x)(432\*x.^4+72\*x.^2+16\*x+3)\*exp(1)-8\*exp(6\*x);

xc=bisect (f,0,1,0.5\*10.^(-11))

xc =

0.166992187500000

xc=bisect (f,0,1.2,0.5\*10.^(-11))

xc =

0.166992187500000

xc=bisect (f,0,1.4,0.5\*10.^(-11))

xc =

0.166796875000000

abs(f(0.1669921875))

ans =

0

abs(f(0.1667968750))

ans =

0

diary end