

**《数值计算方法》第二周作业**

**姓名与学号**  王德茂（3220105563）

**指导教师**  徐祖华

**年级与专业**  机器人2202

**所在学院**  控制科学学院

第二周作业

1. 问题提出：

工程师经常需要计算抛物的轨迹。假设坐标系如图所示，左边的球向右抛出后的轨迹可由如下公式计算：

假设：球抛出的初始速度为:30

抛出点的高度为:1.8

重力加速度为:9.81

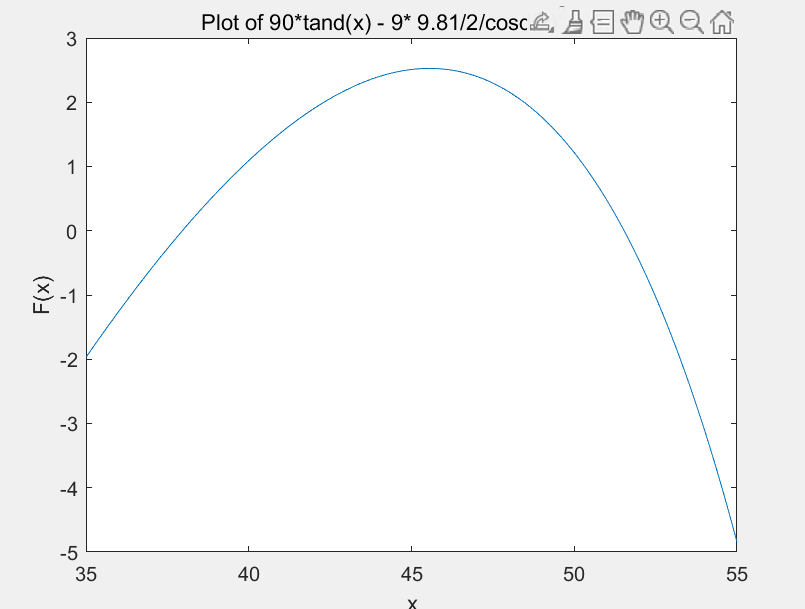
若球到达离抛出点水平距离处时90，球的高度为1

请：计算球的抛出角。要求分别用二分法、试位法、不动点迭代、Newton-Raphson法和割线法求解，并对各种方法进行比较。（提示：存在两个根)

1. 解决思路：

按照计算方程根的几种方法，分别执行。

在开始之前，由于我们已知表达式，所以我们可以通过图像进行直观的判断：



从中，我们可以看到根有两个，并且大约位于35-40，50-55之间。这为我们选择根的起始点有了依据。

1. 二分法

在确定方程感兴趣的根值区间后，找到一个两端点函数值异号的子区间，并考察该子区

间的中点的函数值，反复利用根的存在性定理，循环减半根估计区间，递归求得确定精度下的根的估计值。

在这里，我们确定当小数点后7位不再改变时，停止迭代。

程序实现如下：

format long;

F=@(x) 90\* tand(x)- 9 \* 9.81 / 2 / cosd(x)^2+ 0.8;

x1=35;

x2=40;

x3=50;

x4=52;

count1=0;

count2=0;

G=@(x) 90/cosd(x)^2 - 9 \* 9.81 \*sind(x)  / cosd(x)^3;

while count1 < 10

    x1 = x1 - F(x1)/G(x1);

    count1 = count1+1 ;

end

disp(x1)

while count2 < 10

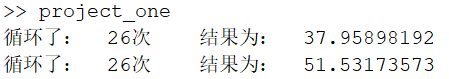
    x4 = x4 - F(x4)/G(x4);

    count2 = count2 + 1;

end

disp(x4)

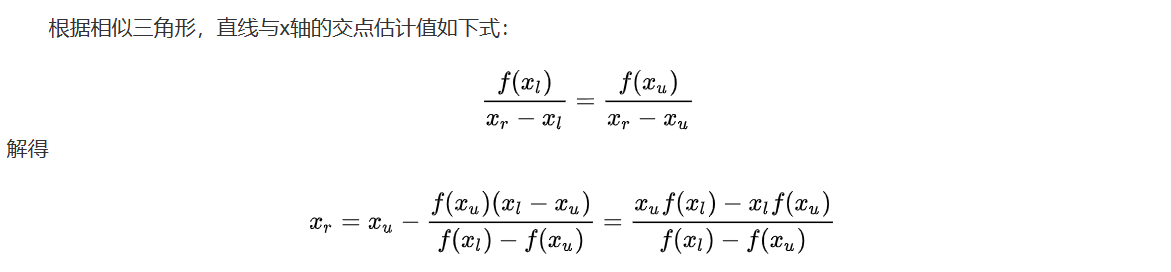
执行结果如下：



1. 试位法

试位法是计算作为一个方程的根的未知量的一种方法，是先做出一个或者一些估计，再从这个或者这些估计出发并根据未知量的性质求出代求的未知量。

试位法类似于二分法，也是将含根区间逐渐缩小，但它并不是单一的二分区间，而是利用区间两个端点的线性插值来求一个近似根。



在这里，我们确定当小数点后7位不再改变时，停止迭代。

程序实现如下：

format long;

F=@(x) 90\* tand(x)- 9 \* 9.81 / 2 / cosd(x)^2+ 0.8;

x1=35;

x2=40;

x3=50;

x4=55;

count1=0;

count2=0;

err = 1e-7;

help = 0;

xr1=1;

xr2=1;

while abs(help-xr1) > err

    xr1=x2-F(x2) \* ( x1 - x2 ) / (F(x1) - F(x2));

    if (F(xr1) == 0)

        break;

    end

    if F(xr1)\*F(x2)<0

        help = x1;

        x1 = xr1;

    else

        help = x2;

        x2 = xr1;

    end

    count1 = count1 + 1 ;

end

s1="循环了：  "+num2str(count1)+"次    结果为：  "+num2str(xr1,10);

disp(s1)

help=0;

while abs(help - xr2)>err

    xr2=x4-F(x4) \* ( x3 - x4 ) / (F(x3) - F(x4));

    if (F(xr2) == 0)

        break;

    end

    if F(xr2)\*F(x4)<0

        help = x3;

        x3 = xr2;

    else

        help = x4;

        x4 = xr2;

    end

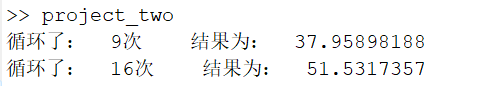
    count2 = count2 + 1 ;

end

s2="循环了：  "+num2str(count2)+"次    结果为：  "+num2str(xr2,10);

disp(s2)

输出结果：

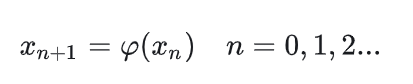


从中可以看出，在该问题中，试位法较快的得到了结果，效果优于二分法。

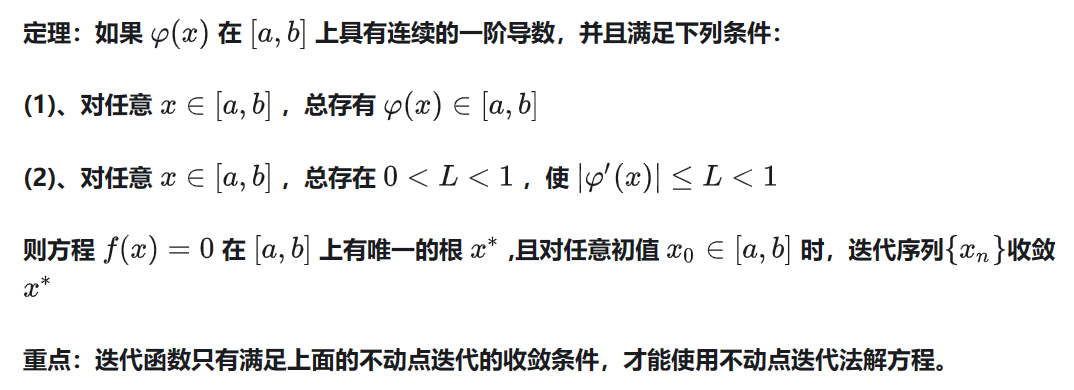
五、不动点迭代

不动点迭代法是求单变量线性方程近似根的一种重要方法，首先确定一个方程根附近的近似初始值，采用逐次逼近的方法，使用迭代公式不断地更新这个初始值，使这个初始值不断趋近于准确值。

不动点迭代法的思想原理：将方程f（x） 转换成一个等价的方程 ，以此构造一个迭代公式：



在这里，我们由于考虑到收敛性，构造了两个不同的迭代函数。



代码实现如下：

format long;

F=@(x) 90\* tand(x)- 9 \* 9.81 / 2 / cosd(x)^2+ 0.8;

x1=35;

x2=40;

x3=52;

x4=55;

count1=0;

count2=0;

err= 1e-7;

G1=@(x) atand( (9 \* 9.81 / 2 / cosd(x)^2 - 0.8) /90 );

G2=@(x) 90\* tand(x)- 9 \* 9.81 / 2 / cosd(x)^2+ 0.8 + x;

while abs( G1(x1) - x1 )>err

    x1 = G1(x1);

    count1 = count1 + 1 ;

end

s1="循环了：  "+num2str(count1)+"次    结果为：  "+num2str(G1(x1),10);

disp(s1)

while abs(G2(x4)-x4)>err

    x4 = G2(x4);

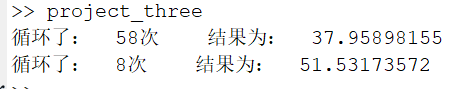
    count2 = count2 + 1 ;

end

s1="循环了：  "+num2str(count2)+"次    结果为：  "+num2str(G2(x4),10);

disp(s1)

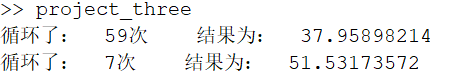
结果如下：



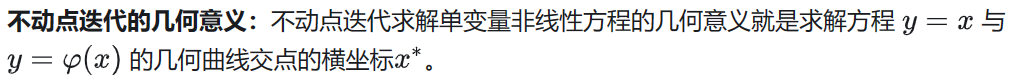
从中我们很明显的看出，给定不同的迭代函数，其收敛速度不同。

当我们改变初始值时，（x1=40 , x4 =50）

结果如下：



可以看到，初值和迭代函数都对不动点法有影响。



六、Newton-Raphson法

多数方程不存在求根公式，因此求精确根非常困难，甚至不可解，从而寻找方程的近似根就显得特别重要。方法使用函数的泰勒级数的前面几项来寻找方程的根。牛顿迭代法是求方程根的重要方法之一，其最大优点是在方程的单根附近具有平方收敛，而且该法还可以用来求方程的重根、复根，此时线性收敛，但是可通过一些方法变成超线性收敛。另外该方法广泛用于计算机编程中。

代码实现如下：

format long;

F=@(x) 90\* tand(x)- 9 \* 9.81 / 2 / cosd(x)^2+ 0.8;

x1=35;

x2=40;

x3=50;

x4=52;

count1=0;

count2=0;

err = 1e-7 ;

G=@(x) 90/cosd(x)^2 - 9 \* 9.81 \*sind(x)  / cosd(x)^3;

help = 0;

while abs(help - x1) > err

    help = x1;

    x1 = x1 - F(x1)/G(x1);

    count1 = count1+1 ;

end

s1="循环了：  "+num2str(count1)+"次    结果为：  "+num2str(x1,10);

disp(s1)

help = 0;

while abs(help - x4) >err

    help = x4;

    x4 = x4 - F(x4)/G(x4);

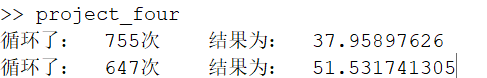
    count2 = count2 + 1;

end

s2="循环了：  "+num2str(count2)+"次    结果为：  "+num2str(x4,11);

disp(s2)

其结果令人吃惊：



可以看到，此时的Newton-Raphson法效率极低。

我们来分析一下，输出每一步的导数值以及函数值。

代码如下：

format long;

F=@(x) 90\* tand(x)- 9 \* 9.81 / 2 / cosd(x)^2+ 0.8;

x1=35;

x2=40;

x3=50;

x4=52;

count1=0;

count2=0;

G=@(x) 90/cosd(x)^2 - 9 \* 9.81 \*sind(x)  / cosd(x)^3;

while count1<10

    x1 = x1 - F(x1)/G(x1);

    count1 = count1+1 ;

    s1="循环了：  "+num2str(count1)+"次    结果为：  "+num2str(x1,10)...

        +"F(x1)为： "+num2str(F(x1)) +"   G(x1)为: "+ num2str(G(x1))   ;

    disp(s1)

end

while count2<10

    x4 = x4 - F(x4)/G(x4);

    count2 = count2 + 1;

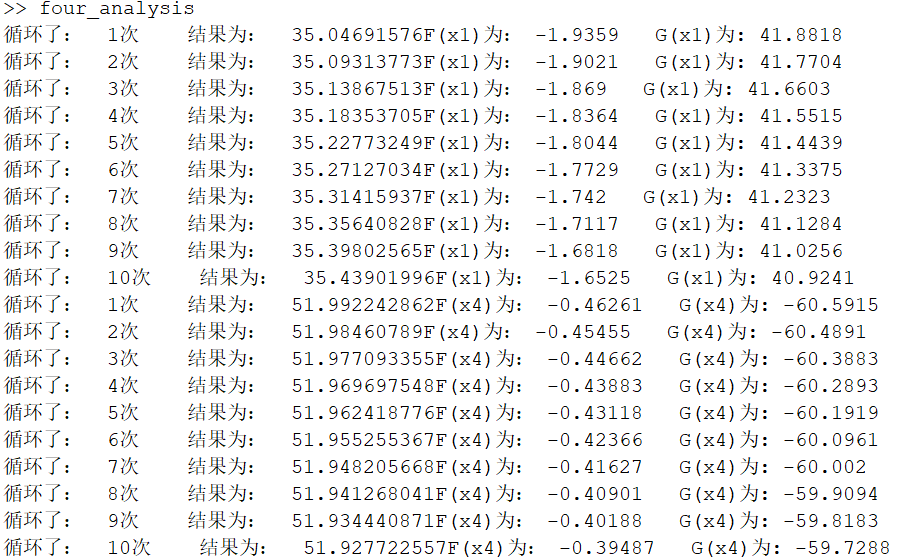
    s2="循环了：  "+num2str(count2)+"次    结果为：  "+num2str(x4,11)...

            +"F(x4)为： "+num2str(F(x4)) +"   G(x4)为: "+ num2str(G(x4)) ;

    disp(s2)

end

结果如下：



从中我们可以明白，**导数值较大，而函数值大小，使得每一次变化很小，收敛很慢**。

七、割线法

割线法，又称弦割法、弦法，是基于牛顿法的一种改进，基本思想是用弦的斜率近似代替目标函数的切线斜率，并用割线与横轴交点的横坐标作为方程式的根的近似。 它是求解非线性方程的根的一种方法，属于逐点线性化方法。

代码实现：

format long;

F=@(x) 90\* tand(x)- 9 \* 9.81 / 2 / cosd(x)^2+ 0.8;

x1=35;

x2=40;

x3=50;

x4=55;

count1=0;

count2=0;

err= 1e-7;

help = 0;

while abs(x2-help) > err

    help=x2;

    x2 = x1 - F(x1) \* (x1 - x2) /(F(x1) - F(x2));

    count1 = count1 + 1 ;

end

s1="循环了：  "+num2str(count1)+"次    结果为：  "+num2str(x2,10);

disp(s1)

help = 0 ;

while abs(x4 - help) > err

    help =x4;

    x4 = x3 - F(x3) \* (x3 - x4) /(F(x3) - F(x4));

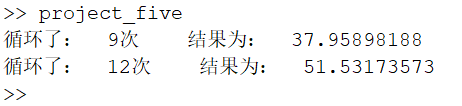
    count2 = count2 + 1 ;

end

s2="循环了：  "+num2str(count2)+"次    结果为：  "+num2str(x4,10);

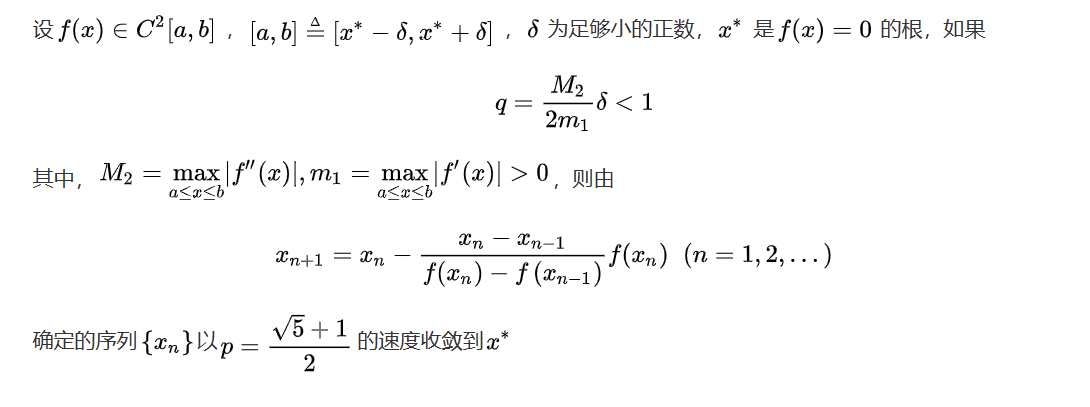
disp(s2)

获得结果如下：



可以看到，**割线法有较好的收敛速度**。

割线法收敛定理：



八、小结：

在这个问题中，我们通过二分法、试位法、不动点迭代、Newton-Raphson法和割线法分别求解了球的抛出角，并对各种方法进行了比较。通过实际计算和结果分析，我们发现试位法和割线法在这个问题中表现较好，收敛速度较快，而Newton-Raphson法的效率较低。并且我们进行了分析出现这一情况的原因。

在不动点迭代法中，不同的迭代函数和初始值会影响结果，需要谨慎选择。综合来看，选择合适的方法和参数对于求解抛物轨迹问题至关重要。