Lab 2 实验报告

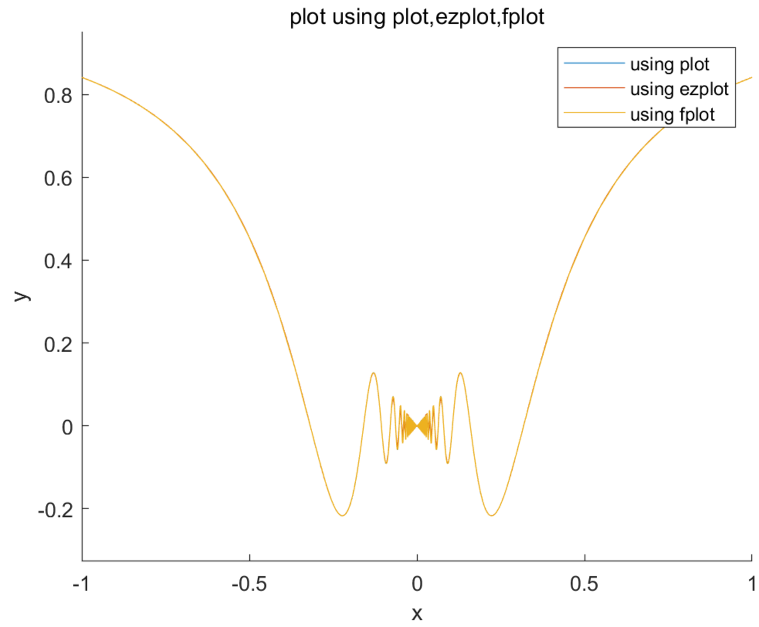
实验一

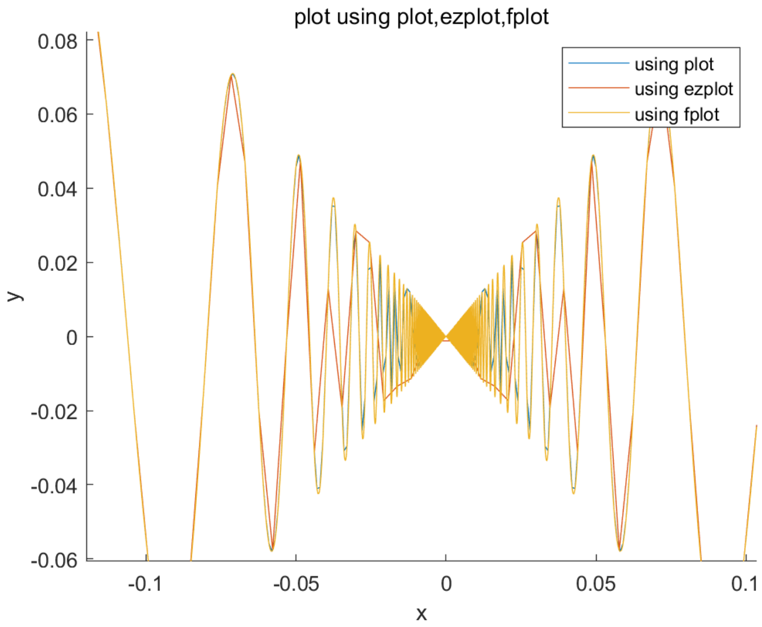
实验一分别用ezplot, fplot, plot画出了$x\sin(\frac{1}{x})$的图像。用到的代码如下

task1.m

```
close;
clear;
x=-1:0.001:1;
y=x.*sin(1./x);
hold on;
plot(x,y)

ezplot('x*sin(1/x)',[-1,1]);

fplot('x*sin(1/x)',[-1,1]);
title('plot using plot,ezplot,fplot');
xlabel('x');
ylabel('y');

legend('using plot','using ezplot','using fplot');
```

图像如下图所示





可以看出用三个function画出的图像基本重合，放大之后再看，可以看出fplot画出的光滑性最好，而ezplot和plot越接近0时就有越呈现不光滑的地方。

整个函数的特性可以看出越接近0时越密集，震荡的越厉害，并且也可以从图像中得出$\lim_{x\to 0} x\sin(\frac{1}{x}) = 0$

实验二

实验二是解一元二次方程。代码如下。

```
%Method 1
delta=A(2)^2-4*A(1)*A(3);
x1=(-A(2)+sqrt(delta))/(2*A(1));
x2=(-A(2)-sqrt(delta))/(2*A(1));
[x1,x2]

%Method 2
roots(A)

%Method 3
x1_old=-101;
x1_new=-100;
while(abs(x1_old-x1_new)>0.0001)
    x1_old=x1_new;
    y=A(1)*x1_old^2+A(2)*x1_old+A(3);
    b=y-(2*A(1)*x1_old+A(2))*x1_old;
    x1_new=-b/(2*A(1)*x1_old+A(2));
end
x2_old=-101;
x2_new=-100;
while(abs(x2_old-x2_new)>0.0001)
    x2_old=x2_new;
    y=A(1)*x2_old^2+A(2)*x2_old+A(3);
    b=y-(2*A(1)*x2_old+A(2))*x2_old;
    x2_new=-b/(2*A(1)*x2_old+A(2));
end
[x1_new,x2_new]
```

第一个方法就是用的二次方程求根公式，而matlab内置了虚数，所以可以直接求出虚数根。

第二个方法用的matlab内置的roots的方法，可以直接求解。

第三个方法是用的牛顿法，先确定一点（如代码中较小根找的是-100，较大根找的是100），再做这一点的切线，与x轴相交的点即为下一次更新的点。如此反复，直至此次与上一次的两点的横坐标之差小于0.0001，我们便可以得出最后的结果。

前两个方法都可以很标准的解出方程的解，第三个方法则不是那么精确，主要可能遇到的问题有：

- 如果两个解都位于-100的左侧或者100的右侧，则无法找到另一个解，这可以从牛顿法的原理的去解释
- 解的结果精确度不高，这跟设置的精度有关。如上的代码在运行$x^2+4x+4=0$的方程的解时得到的答案就（如图所示）可能存在较小的误差。
- 牛顿法解不了虚数根，因为牛顿法本质是寻找曲线与x轴的交点，而没有交点，则解不出虚数根。

```
ans =

   -2.0001   -2.0001
```

最后，附上一些方程的解

```
Input three numbers are:[1 4 4]

ans =

    -2    -2


ans =

    -2
    -2


ans =

   -2.0001   -2.0001
```

```
Input three numbers are:[1 4 3]

ans =

    -1    -3


ans =

    -3
    -1


ans =

   -3.0000   -3.0000
```

```
Input three numbers are:[1 2 3]

ans =

  -1.0000 + 1.4142i  -1.0000 - 1.4142i


ans =

  -1.0000 + 1.4142i
  -1.0000 - 1.4142i
```

实验三

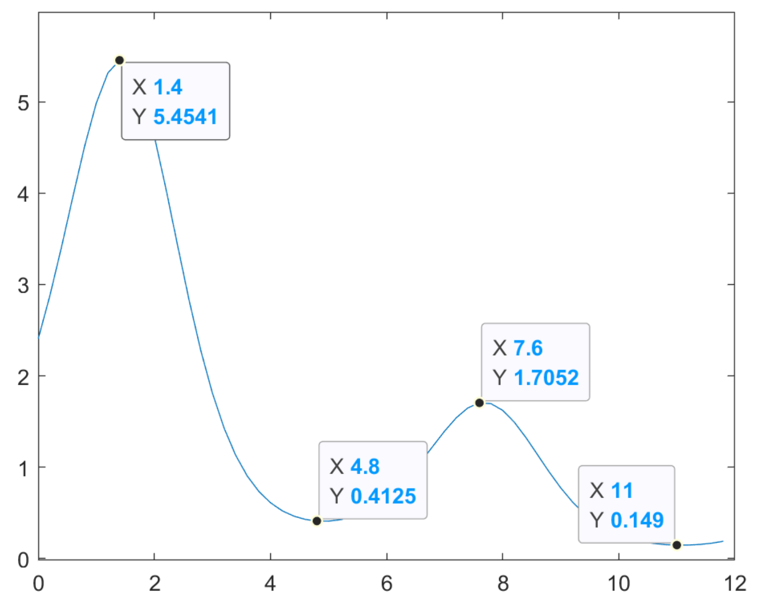
实验三是寻找拐点。代码如下。

```
close;
clear;
clc;

load('HW2Exp3data.mat');
data=table2array(HW2Exp3data);
plot(data(:,1),data(:,2));

X=data(:,1);
y=data(:,2);
grad=gradient(y,X);
point=[];
for i=1:59
    if grad(i)*grad(i+1)<0
        pt=0.2*(i-1);
        point(end+1)=pt;
    end
end
point

```

基本思路是算出 y 对于 x 在每一点的梯度，并在梯度值符号发生变化时记录（即相乘小于0），此时就是拐点。

得出的答案如图，并在图上也可以清晰的看出。

```
point =

    1.4000    4.8000    7.6000   11.0000
```

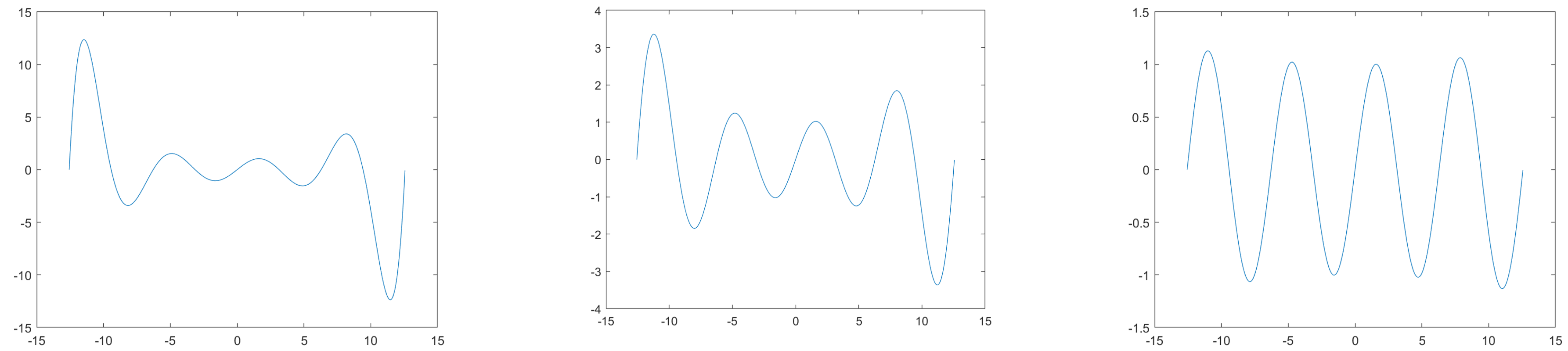


实验四

实验四的代码如下。

```
close;
clear;

x=-4*pi:0.01:4*pi;

n=5;

y=x;
for i=1:n
    y=y.*(1-x.^2/(i^2*pi^2));
end

plot(x,y);
```

基本思路就是不断累乘，得出最后的图像。



最后从左到右分别得到的是n=5,10,100的图像，可以看出随着n的增大，图像越逼近$\sin(x)$的图像。