非线性方程求根

上机题2

本题使用牛顿法以及牛顿下山法来进行非线性方程的求根。

运行代码 `python P2.py` 后，可得到以下结果：

```
function 1: x^3 - 2x + 2
Newton method:
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
```

```
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
```

```
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
x = 1.000000
x = 0.000000
Result:  None
Newton down-hill method:
x = 1.000000, lamdas = []
x = 0.657000, lamdas = [0.7, 0.48999999999999994, 0.3429999999999999]
x = 0.888131, lamdas = [0.7, 0.48999999999999994, 0.3429999999999999,
0.24009999999999992, 0.16806999999999994]
x = -1.634938, lamdas = []
x = -1.784405, lamdas = []
x = -1.769454, lamdas = []
x = -1.769292, lamdas = []
Result:  -1.7692923542386316
function 2: -x^3 + 5x
Newton method:
x = 10.525668
x = 7.124287
x = 4.910781
x = 3.516911
x = 2.709743
x = 2.336940
x = 2.242244
x = 2.236093
x = 2.236068
Result:  2.23606797749979
Newton down-hill method:
x = 2.429508, lamdas = [0.7, 0.48999999999999994, 0.3429999999999999,
0.24009999999999992, 0.16806999999999994, 0.11764899999999995]
x = 2.256960, lamdas = []
x = 2.236355, lamdas = []
x = 2.236068, lamdas = []
Result:  2.2360679774997916
```

对于两种方法，设置的迭代判停准则为迭代100次或者两次迭代解的插值绝对值小于 10^{-6} 。

对于牛顿下山法中的下山因子，设置成 $\{\lambda_i\} = \{0.7, 0.7^2, \cdots\}$

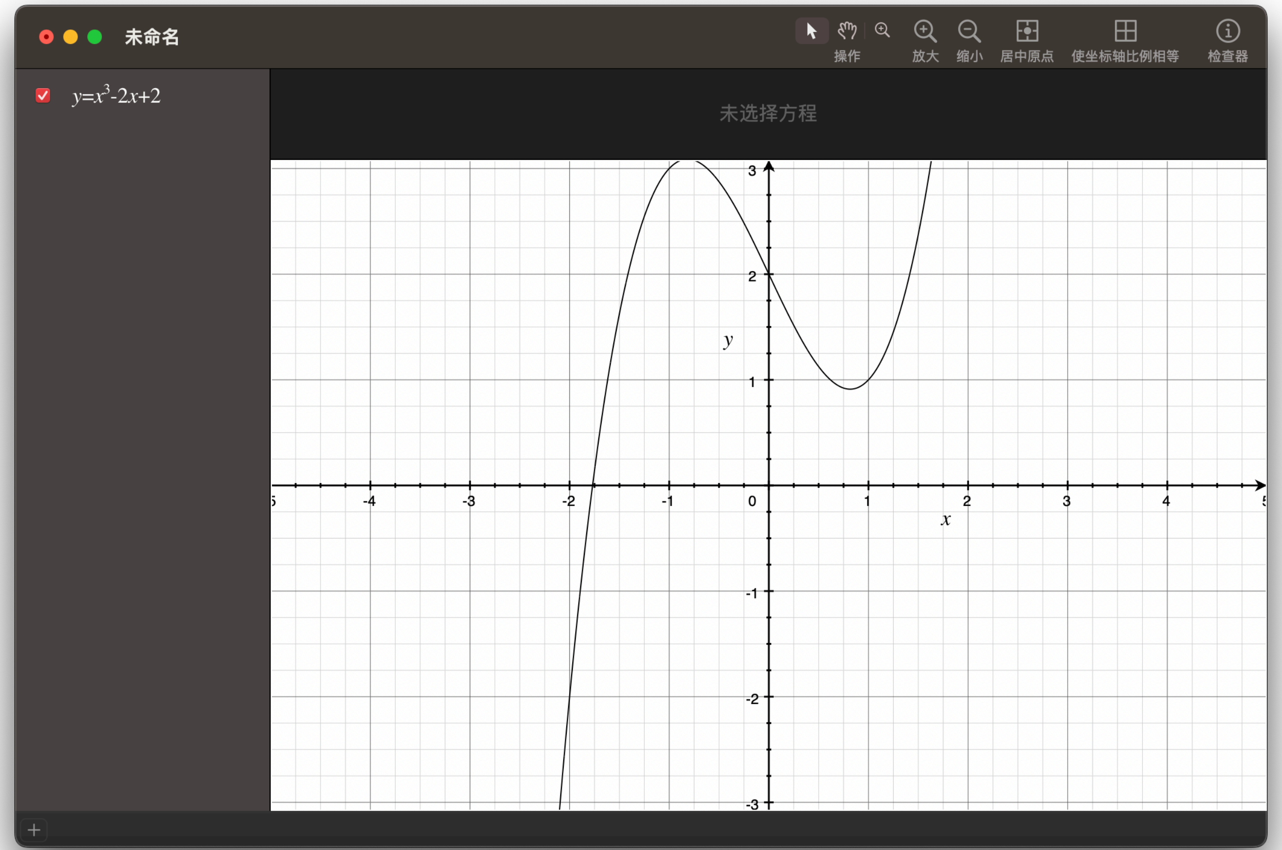
使用MATLAB运行 `P2_2.m` 后可得到如下输入：

```
ans =

   -1.7693


ans =

    2.2361

```

可以发现得到的解均为正确解。

可以看到，对于第一个函数 $x^3-2x+2=0$，直接使用牛顿法，会导致迭代的解在 $0,1$ 之间反复横跳，因为原始的牛顿法没有阻尼因子。而使用牛顿下山法之后，则可以较好地收敛到一个正确解。因此，第一个函数需要用到牛顿下山法。

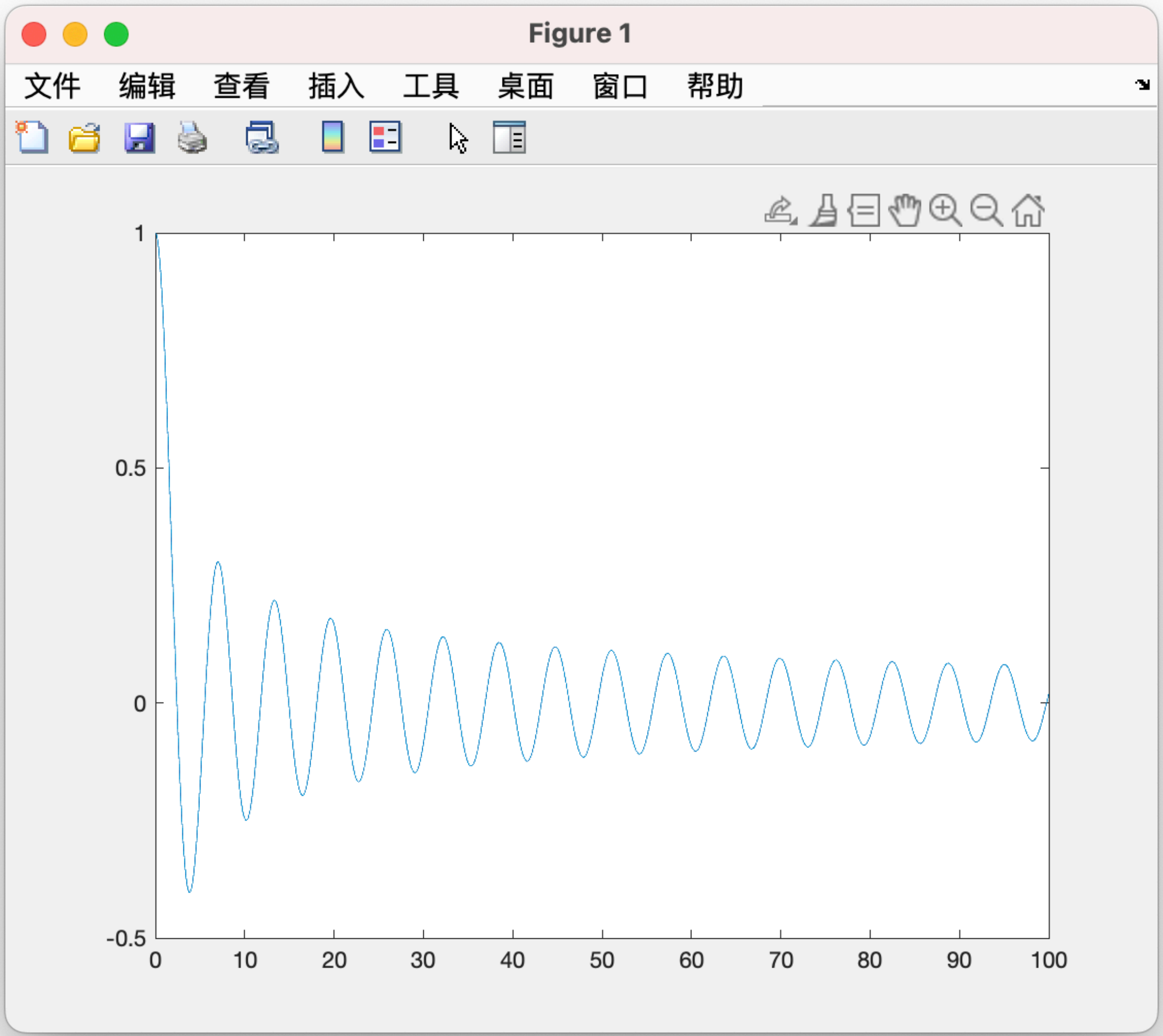
同时，对于第一个函数，当取下山因子为 $\{\lambda_i\}=\{0.5,0.5^2,\cdots\}$ 时，会发现牛顿下山法依然会困在局部，得不到正确解，观察输出发现，是由于下山因子减小得太快，无法迭代 x 到 x 轴左端 $[-2,-1]$ 中那一小段更优的区间内。



对于第二个函数，则两种方法都能收敛到一个解，且牛顿下山法的迭代次数更少，收敛更快。

上机题3

运行 `P3_2.m` 观察 $J_0(x)$ 在正半轴的图像：



可以发现零点之间相差都大于 1 ，且不为整数。因此在 `P3.m` 中，从 $[0,1]$ 开始，不断将区间向右位移 1 ，如果区间两端的函数值异号，则调用 `fzerotx.m` （该代码参考教材中2.6.3节的程序）来求区间内的零点，直到求得零点个数达到 10 则终止程序。运行 `P3.m` 后可以得到如下结果：

```
ans =

    2.4048


ans =

    5.5201
```

```
ans =

    8.6537


ans =

   11.7915


ans =

   14.9309


ans =

   18.0711


ans =

   21.2116


ans =

   24.3525


ans =

   27.4935


ans =

   30.6346
```

即为要求的前十个正零点。