第五次作业

赖显松 2021214726

1 书面作业题目

P115 1 (2) 二分法求方程 $x^3-x-1=0$ 在 [1, 1.5] 根

解: $k > \log_2\frac{b-a}{\varepsilon} = \log_2\frac{0.5}{10^2} = 5.64$ 取 $k=6$ 到需要6次

$f(x) = x^3-x-1$

$f(1) = 1-1-1 = -1 < 0$

$f(1.5) = 1.5^3-1.5-1 = 0.875 > 0$

① $f(1.25) = 1.25^3-1.25-1 = -0.297 < 0$

② $f(1.375) = 1.375^3-1.375-1 = 0.225 > 0$

③ $f(1.3125) = 1.3125^3-1.3125-1 = -0.052 < 0$

④ $f(1.34375) = 0.0826 > 0$

⑤ $f(1.328125) = 0.0146 > 0$

⑥ $f(1.3203125) = -0.0187 < 0$

∴ 此根为 $x_1 = \frac{1.3203125 + 1.328125}{2} = 1.3242$

P115 3. 解: $x \in [1, 2]$

$\varphi_1(x) \in [1.25, 2]$ 收敛! $\varphi_1'(x) = -2x^{-3}$

$\varphi_2(x) \in [1.26, 1.7]$ 收敛! $\varphi_2'(x) = \frac{1}{3}(1+x^2)^{-\frac{2}{3}} \cdot 2x$

$\varphi_3(x) \in [1, \infty)$ 不收敛!

$\varphi_1'(x^*) \approx \varphi_1'(1.3) = -0.9$

$\varphi_2'(x^*) \approx \varphi_2'(1.3) = 0.448$ ∴ φ_2 收敛更快, 选 φ_2 进行迭代

$L \geq \frac{|\varphi(x)-\varphi(y)|}{|x-y|} = \frac{|1.7-1.26|}{1} = 0.45$

若要满足绝对误差界 $|x^*-x_k| \leq 10^{-2}$

则 $|x_k - x_{k-1}| < \frac{1-L}{L}10^{-2} = \frac{0.55}{0.45}\times 10^{-2} = 0.012$

$x_0 = 1.5$

$x_1 = \sqrt[3]{1+1.5^2} = 1.481248$ $|x_1 - x_0| = 0.12$

$x_2 = \sqrt[3]{1+x_1^2} = 1.4727057$ $|x_2 - x_1| = 0.0085427$ < 0.012 迭代结束

方程的根为 $x = 1.473$

P116 T11

$$x^3-3x+2=(x-1)x^2+(x-1)x+(x-1)\times 2$$
$$=(x-1)(x^2+x-2)$$
$$=(x-1)^2(x+2)$$

$\therefore x^*=1$ 是方程的二重根

(1) Newton 法

迭代式 $x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}=x_k-\frac{x_k^3-3x_k+2}{3x_k^2-3}$

$$x_1=x_0-\frac{x_0^3-3x_0+2}{3x_0^2-3}=1.103$$
$$x_2=x_1-\frac{x_1^3-3x_1+2}{3x_1^2-3}=1.052$$
$$x_3=x_2-\frac{x_2^3-3x_2+2}{3x_2^2-3}=1.0264$$

结论：
相对于普通的牛顿法，修改后的牛顿法迭代速度要更快，收敛更早。

(2) 迭代式 $x_{k+1}=x_k-2\frac{x_k^3-3x_k+2}{3x_k^2-3}$

$$x_1=x_0-2\frac{x_0^3-3x_0+2}{3x_0^2-3}=1.006$$
$$x_2=x_1-2\frac{x_1^3-3x_1+2}{3x_1^2-3}=1.0000061$$
$$x_3=x_2-2\frac{x_2^3-3x_2+2}{3x_2^2-3}=1$$

(3) 迭代式 $x_{k+1}=x_k-\frac{(x_k^3-3x_k+2)(3x_k^2-3)}{(3x_k^2-3)^2-(x_k^3-3x_k+2)\times(6x_k)}$

$$x_1=x_0-\frac{(x_0^3-3x_0+2)(3x_0^2-3)}{(3x_0^2-3)^2-(x_0^3-3x_0+2)\times 6x_0}=0.994152$$
$$x_2=x_1-\frac{(x_1^3-3x_1+2)(3x_1^2-3)}{(3x_1^2-3)^2-(x_1^3-3x_1+2)\times 6x_1}=0.9999942779615$$
$$x_3=x_2-\frac{(x_2^3-3x_2+2)(3x_2^2-3)}{(3x_2^2-3)^2-(x_2^3-3x_2+2)\times 6x_2}=1$$

2 编程题目

2.1 P116 T1

不动点迭代法函数

编写不动点迭代法函数（输入：x 初值，误差要求，迭代函数 Fx；返回：迭代求解结果）。代码如下：

```python
# fixed-point Iteration
def fixedPtIter(x_old, eps, Fx, *args, **kwargs):
    max_iter = 100
    err = 10
    it_num = 1
    while err > eps and it_num <= max_iter:
        x_new = Fx(x_old)
        err = abs(x_new-x_old)
        x_old = x_new
        it_num += 1
    print ('iteration times: ', it_num)
    return x_new
```

编写(1)、(2)、(5)方法的迭代公式：

(1):

```python
def iterFx1(x):
    return 20 / (x**2 + 2*x +10)
```

(2):

```python
def iterFx2(x):
    return (20 - 2*x**2 - x**3)/10
```

(5):

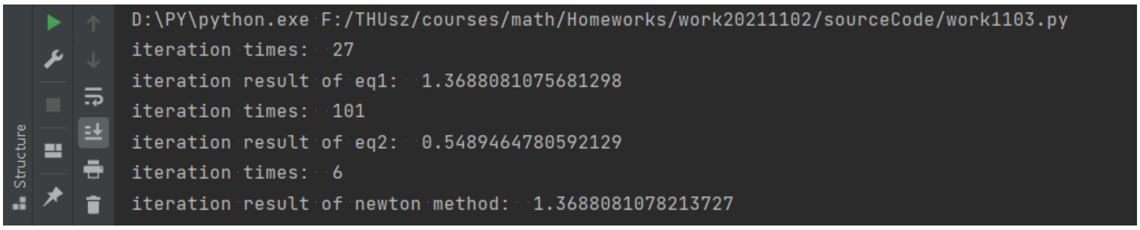
```python
def Fx(x):
    return x**3 + 2*x**2 + 10*x - 20


def dFx(x):
    return 3*x**2 + 4*x + 10


def newtonIter(x):
    return x - Fx(x) / dFx(x)
```

结果

待验证的迭代结果约为 1.368808107，所以迭代精度设置为 10^{-9}。初值取 1，求得不同迭代方法的迭代次数以及结果如图 2. 1 所示：



```
D:\PY\python.exe F:/THUsz/courses/math/Homeworks/work20211102/sourceCode/work1103.py
iteration times:  27
iteration result of eq1:  1.3688081075681298
iteration times:  101
iteration result of eq2:  0.5489464780592129
iteration times:  6
iteration result of newton method:  1.3688081078213727
```

图 2. 1 3 种方法的迭代结果

证明 Leonardo 所得到结果是准确的。

从结果我们能够看出，迭代法(1)、(5)牛顿法都是可以收敛的，且牛顿法的收敛速度要快得多。而(2)迭代式最终并没有成功收敛。

2.2 附加题

已知根多项式方程可以表示为(2- 1)：

$$\omega(n)=\prod_{i=1}^{n}(x-x_i)=0 \tag{2- 1}$$

根据乘法求导法则，其一阶导数为(2- 2)：

$$[\omega(n)]'=\sum_{j=1}^{n}\prod_{i=1,i\neq j}^{n}(x-x_i)=0 \tag{2- 2}$$

以此类推，这种多项式的导数相当于每次求导，将连乘项减少一项进行组合，并将组合相加，例如(2- 3)：

$$((x-1)(x-2)(x-3))'=(x-2)(x-3)+(x-1)(x-3)+(x-1)(x-2) \tag{2- 3}$$

编程细节

通过编程实现这种多项式的生成及求导，采用查表的方式：

一个二维数组表示用来生成已知根多项式及其导数，数组中的元素代表 x_i 这些根，表格中的每一行元素相乘，最后将所有行相加。

例如：$(x-2)(x-3)+(x-1)(x-3)+(x-1)(x-2)$ 可以表示为：$[[2,3],[1,3],[1,2]]$。这种表示方法我称为“乘加表”。

函数 1：除外组合

这个函数输入一个这样的乘加表，输出对应多项式导数的乘加表。原理在于，对于源数组的每一行，求其减少一个元素的组合（暗顺序依次去掉其中的一个元素，剩下的元素就是一个组合），最后将所有组合作为一行按照从上至下的顺序排列。若输入是 $m \times n$ 维的数组，则输出为 $(m \cdot n) \times (n-1)$ 维的数组。代码如下：

```
def exceptTable(na):
    m, n= na.shape[0], na.shape[1]
    nb = np.zeros((n*m, n-1), dtype=int)
    for i1 in range(m):
        for j1 in range(n):
            ej = 0
            j2 = 0
            while ej < n:
                if ej != j1:
                    nb[i1*n+j1, j2] = na[i1, ej]
                    j2 += 1
                ej += 1
    return nb
```

函数 2： 乘加表变为多项式

行元素相乘，最后将每一行相加，代码如下：

```
def mulPlus(x, na):
    y = 0
    for i in range(na.shape[0]):
        yr = 1
        for j in range(na.shape[1]):
            yr = yr * (x-na[i, j])
        y += yr
    return y
```

函数 3： 生成已知根多项式及其导数

输入自变量、最初的已知根多项式系数序列、求导次数，输出代入自变量的多项式计算结果，没求一次导，相当于多调用一次除外组合函数。代码如下：

```
def omigaPolynomial(x, xi, d_times):
    for k in range(d_times):
        xi = exceptTable(xi)
    return mulPlus(x, xi)
```

多根多项式求解需要用到修正的牛顿迭代函数。进入实现环节，先定义原多项式，再定义其一阶导、二阶导函数。代码如下：

```
def MrFx(x, xi):
    return nleqi.omigaPolynomial(x, xi, 0)
```

```python


def dMrFx(x, xi):
    return nleqi.omigaPolynomial(x, xi, 1)


def d2MrFx(x, xi):
    return nleqi.omigaPolynomial(x, xi, 2)
```

再定义不动点迭代法迭代函数 $\varphi(x)$，代码如下：

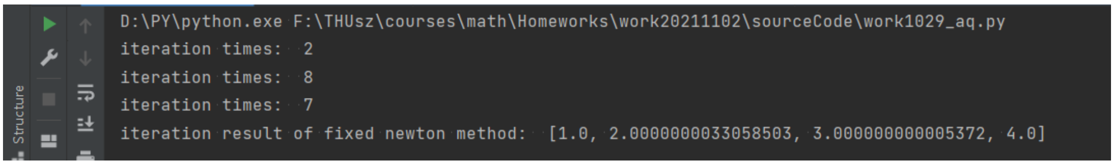
```python
def mrNewtonIterF(x, xi: np.ndarray):
    return x - (MrFx(x, xi)*dMrFx(x, xi)) / ((dMrFx(x, xi)**2)-(MrFx(x, xi)*d2MrFx(x, xi)))
```

定义多项式根序列及初值，每次求解得到一个根之后，更新根序列表，最后一个根为一次函数，直接输出根的值。代码如下：

```python
xi = np.array([[1, 2, 3, 4]])
root_nums = xi.shape[1]
x_0 = 28
x_solve_mn = []
for i in range(root_nums):
    # only one root left
    if xi.shape[1] == 1:
        x_solve_mn.append(xi[0, 0])
        break
    else:
        x_star = nleqi.mrNewtonFP(x_0, 10**(-3), mrNewtonIterF, xi)
        x_solve_mn.append(x_star)
        temp = np. zeros((1, xi.shape[1]-1))
        k = 0
        for j in range(xi.shape[1]):
            if round(xi[0, j]) != round(x_star):
                temp[0, k] = xi[0, j]
                k += 1
            if k == temp.shape[1]:
                break
        xi = temp.copy()
print('iteration result of fixed newton method: ', x_solve_mn)
```

结果

代入初值为 0 时的求解结果为如图 2. 2：



```
D:\PY\python.exe F:\THUsz\courses\math\Homeworks\work20211102\sourceCode\work1029_aq.py
iteration times:  2
iteration times:  8
iteration times:  7
iteration result of fixed newton method:  [1.0, 2.0000000033058503, 3.000000000005372, 4.0]
```

图 2. 2 初值为零时的多根多项式修正牛顿法迭代结果

求出了四个根[1, 2, 3, 4]。

初值敏感性分析

以 0.1 为间隔，多次取不同的初值，得到四个解随初值求解先后顺序，画出变化图线。如图 2. 3：

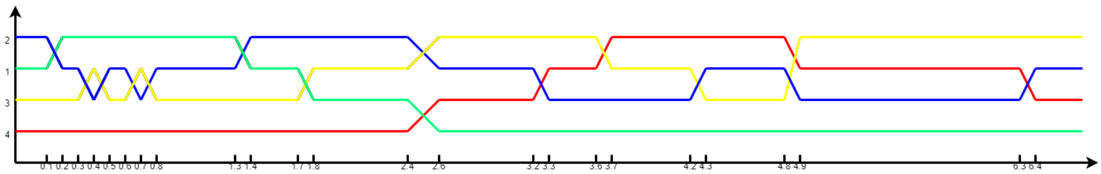


图 2. 3 求根先后顺序随初值变化曲线

X 轴为初值，先求出的根在 Y 轴的高度更高。在不同初值情况下，根被计算出来的先后次序不同。在初值很小的时候，先被计算出来的根为 2，最后直接得出的是 4；在初值很大的时候，先被计算出来的根为 3，最后直接得出的是 1。所以 1 和 4 的初值敏感度较大，初值不好时不易得到，粗略比较计算得出顺序，可以认为 4 的初值敏感性大于 1，而 2 的初值敏感性小于 3。最后的初值敏感性顺序为：2<3<1<4。

参考文献

[1] 关冶．数值方法[M]．北京：清华大学出版社，2006：66-92．

附录

见附件 python 源代码。