数值分析上机实验

非线性方程组的迭代解法

1 问题描述

在本次实验中，我们的目的是使用迭代法求解非线性系统. 尤其是运用不动点迭代法及其 Steffensen 加速、Newton 迭代法.

我们的问题是求解如下的多项式方程：

$$x^3 + 2x^2 + 10x - 20 = 0, \tag{1}$$

记 $p(x) = x^3 + 2x^2 + 10x - 20$，求导数可得 $p'(x) = 3x^2 + 4x + 10 = 3(x + \frac{2}{3})^2 + \frac{26}{3} > 0$，所以 p 在 $\mathbb{R}$ 上单调递增，因此应当存在唯一解. 计算可得：

$$p(1) = -7, \quad p(2) = 16, \tag{2}$$

所以唯一解在 1 和 2 之间.

本次实验中我们统一采用 $x_0 = 1$ 为初值，并且希望得到精度为 10^{-7} 的近似解（准确解为 $x^* = 1.368808107\cdots$）.

2 实验内容

本次考虑两种不同的迭代函数 φ 和 ψ，定义如下：

$$\varphi(x) = \frac{20 - 2x^2 - x^3}{10}, \quad \psi(x) = \sqrt[3]{20 - 10x - 2x^2}. \tag{3}$$

这两种迭代模式都属于不动点迭代. 其推导过程简述如下. 对于 φ，我们对方程进行移项：

$$10x = 20 - x^3 - 2x^2, \tag{4}$$

两边除以 10 就得到 $x = \frac{20 - x^3 - 2x^2}{10}$，于是解原方程等价于解 $x = \varphi(x)$. 对于 ψ，我们将原方程改写成

$$x^3 = 20 - 10x - 2x^2, \tag{5}$$

两边同时开立方根即可得到 $x = \psi(x)$.

我们还可以考虑这两个迭代函数的 Steffensen 加速，相当于使用了新的迭代函数：

$$\widetilde{\varphi}(x) = x - \frac{(\varphi(x) - x)^2}{\varphi(\varphi(x)) - 2\varphi(x) + x}. \tag{6}$$

以及：

$$\widetilde{\psi}(x) = x - \frac{(\psi(x) - x)^2}{\psi(\psi(x)) - 2\psi(x) + x}. \tag{7}$$

根据 Steffensen 加速收敛性的充分条件，只要迭代函数 φ 或 ψ 在根 α 处的导数不等于 1，则 Steffensen 加速法至少有二阶收敛性，这属于超线性收敛，对于一般的问题来说已经足够令人满意. 我们使用 x^* 的估计值 1.368808107 来估算 φ' 和 ψ' 如下：

$$\varphi'(x) = \frac{1}{10}(-4x - 3x^2) \Rightarrow \varphi'(x^*) \approx -1.10961, \tag{8}$$

$$\psi'(x) = -\frac{4x + 10}{3(20 - 10x - 2x^2)^{2/3}} \Rightarrow \psi'(x^*) \approx -2.75316, \tag{9}$$

因此 Steffensen 加速迭代函数应当都具有超线性的收敛性能（尽管我们还不能保证 φ 和 ψ 本身是收敛的）.

对于 Newton 迭代法，我们考虑的迭代函数 $\mathcal{N}$ 则为：

$$\mathcal{N}(x) = x - \frac{p(x)}{p'(x)}, \tag{10}$$

首先 $p''(x) = 6x + 4$ 在 $[0, 2]$ 上不变号，其次 $p(x)$ 在 $[0, 2]$ 上有零点，最后 $|p'(0)| = \min(|p'(0)|, |p'(2)|)$ 满足：

$$|p'(0)| \geq \frac{|p(0)|}{2 - 0},$$

根据 Newton 迭代法收敛的充分条件可知，当初值选择在 $x_0 = 1 \in [0, 2]$ 时，Newton 迭代法将二阶收敛（超线性收敛）到 x^*.

以上迭代函数可以通过以下 `Python` 代码实现：

```python
import numpy as np

#def fixed point iterative function phi(x)
def phi(x):
    return (20 - 2*x**2 - x**3)/10

#def fixed point iterative function psi(x)
def psi(x):
    return np.cbrt(20 - 10*x - 2*x**2)

#def stfs speed-up of phi(x)
def spdphi(x):
    return x - (phi(x)-x)**2/(phi(phi(x))-2*phi(x)+x)

```

```
#def stfs speed-up of psi(x)
def spdpsi(x):
    return x - (psi(x)-x)**2/(psi(psi(x))-2*psi(x)+x)

#def polynomial p(x)
def poly(x):
    return x**3 + 2*x**2 + 10*x -20

#def derpolynomial p'(x)
def der(x):
    return 3*x**2 + 4*x + 10

#def newton iterative function
def newt(x):
    return x-poly(x)/der(x)
```

我们定义如下的函数用于执行 “迭代过程”，该函数接受一个初值 x_0，一个迭代函数，以及一个收敛阈值（计算迭代到某一步时的 $p(x)$ 的绝对值，若小于收敛阈值，则停止迭代，认为迭代已经达到收敛）. 注意对于修正前的迭代函数 φ 和 ψ 并不能保证其收敛性，因此我们还接受一个 “最大步数” 参数，即到达该步数时无论是否收敛，都结束迭代过程. 该函数返回迭代过程结束时所得到的近似解 x 以及迭代总次数.

```
#def iteration process
def iter_process(x0,fn,threshold,max_steps):
    x = x0
    cnt_iter = 0
    for i in range(max_steps):
        cnt_iter += 1
        x = fn(x)
        if np.abs(poly(x)) < threshold:
            break
    return x,cnt_iter
```

3 实验结果与讨论

我们首先设置初值为 1.0，阈值统一设定为 10^{-13}，最大步数设定为 100000，运行如下代码：

```
x0 = 1.0
for fn in [phi,psi,spdphi,spdpsi,newt]:
    solution, steps = iter_process(x0,fn,1e-13,100000)
    print("{} iteration ends after {} steps, with output value {}".format(fn.
    __name__,steps,np.real(solution)))
```

输出结果为：

```
phi iteration ends after 100000 steps, with output value 0.548946478054766
psi iteration ends after 100000 steps, with output value -3.162277660168379
spdphi iteration ends after 4 steps, with output value 1.3688081078213719
spdpsi iteration ends after 8 steps, with output value 1.3688081078213725
newt iteration ends after 4 steps, with output value 1.3688081078213745
```

我们看到，前两组迭代和后三组迭代出现了明显的差异，对于前两组迭代，100000 步之后尚未达到收敛，而且结果距离真实值相差甚远. 而对于后三组迭代（分别是 φ 的 Steffensen 加速、ψ 的 Steffensen 加速以及 Newton 迭代函数 $\mathscr{N}$）则用寥寥数次（分别为 $4, 8, 4$）次迭代就使得 $p(x)$ 的绝对值小于 10^{-13}，从计算结果也可以看出，所得近似解已经有至少 10 位有效数字. 这与我们的预期也是一致的，可见超线性的迭代效率真的很高.

我们还想尝试，当初值选择得和准确解足够接近时，前两种不动点迭代是否得到改善. 为此，我们修改初值为 1.3688081078，再次运行程序可得：

```
phi iteration ends after 100000 steps, with output value 0.548946478054766
psi iteration ends after 100000 steps, with output value -3.162277660168379
spdphi iteration ends after 1 steps, with output value 1.3688081078213727
spdpsi iteration ends after 1 steps, with output value 1.3688081078213727
newt iteration ends after 1 steps, with output value 1.3688081078213727
```

可见，尽管我们选取的初值距离准确解已经非常接近，但是 ϕ 和 ψ 竟然仍给出了和选取 $x_0 = 1$ 时完全相同的结果. 为了研究这两种迭代格式的行为，我们运行如下代码，将整个迭代过程用散点图描绘出来：

```
import matplotlib.pyplot as plt

#def iteration process
def plot_iter_process(x0,fn,threshold,max_steps):
x = x0
cnt_iter = 0
scatters = []
for i in range(max_steps):
    cnt_iter += 1
    x = fn(x)
    scatters.append([cnt_iter,x])
    if np.abs(poly(x)) < threshold:
        break
cnt, value = zip(*scatters)
plt.scatter(cnt, value, s = 10)
plt.title('$\{}$'.format(fn.__name__))
plt.xlabel('iteration steps')
plt.ylabel('$x_k$ value')
```

```
plt.show()
```

分别绘制 φ 和 ψ 的收敛行为：

```
plot_iter_process(x0,phi,1e-13,1000)
```

```
plot_iter_process(x0,psi,1e-13,80)
```

散点图如下：从图中可以看出，尽管我们的初值已经选择得相当接近于真实值，当迭代次数

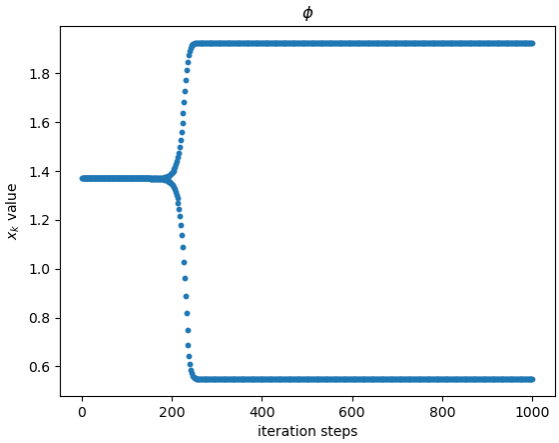

图 1: φ 的收敛行为

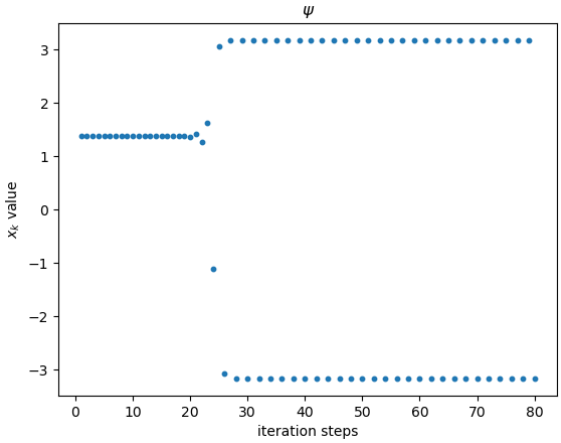

图 2: ψ 的收敛行为

大于某个数后，两种迭代格式迅速发生 “分支现象”，最终在两个值之间来回震荡并达成 “死循环”，不再能够收敛到真正的根. 对于 φ 收敛格式，大约在 200 多步开始出现该现象，ψ 则在第二十步左右就开始发散. 这是因为，φ 在 x^* 附近导数的绝对值比 ψ 小，因此发散较慢.

φ 和 ψ 经过 Steffensen 加速处理后，其收敛都较好. 其中 $\widetilde{\varphi}$ 的收敛速度大约是 $\widetilde{\psi}$ 的两倍. 这和前面对于 φ 和 ψ 的收敛行为某种意义下是相符合的：φ 本身发散较慢，因此做过 Steffensen 改善后，其收敛也更快.

4 总结

通过这次实验我们可以看出，在进行非线性方程组迭代求解之前，我们首先需要在理论上分析所使用的方法是否收敛. 比如本实验中涉及到的 Steffensen 收敛的充分条件、Newton 收敛的充分条件等. 而一般的不动点方法需要严格验证才能使用，在本例子中，φ 和 ψ 都没有明确的条件用以判断其收敛性，因此在实验中果然出现了问题.

另外，初值选取的好坏对于收敛速度、数值精度的影响也很明显. 在实际问题中，我们应当首先估计所使用的方法收敛速度如何（例如是线性收敛还是超线性收敛等）、数值精度如何，再据此设计收敛阈值、最大步长等. 一个好的初值可以很大程度改善计算的效率和精度，因此，先对解所在的区间进行估计，从而确定初值大致范围，也是很重要的.