实验二

张天乐 计96 2018011038

上机题 2

实验内容

编程实现牛顿法与牛顿下山法求解下面两个方程. 要求: (1) 设定合适的迭代判停准则; (2) 设置合适 的下山因子序列; (3) 打印每个迭代步的近似解及下山因子; (4) 请用其他较准确的方法 (如MATLAB软件中的 fzero 函 数) 验证牛顿法与牛顿下山法结果的正确性。最后，总结哪个问题需要用牛顿下山法求解，及采用它之后的效果

实验过程

- 使用残差判据 + 误差判据
- 设定下山因子初值，每次迭代下山因子以一定比例减小
- 打印每个迭代步的近似解及下山因子
- 使用 python scipy 模块求根

In [ ]:

```python
import numpy as np

def newton(fun, derive, x0, damp=False):
    epsilon = 1e-6
    lamb = 0.9
    k = 0
    x = xk =x0
    while np.abs(fun(x)) > epsilon or np.abs(x - xk) > epsilon:
        s = fun(x) / derive(x)
        xk = x
        x = xk - s
        print('Step {:2d}: x = {:.6f}, f(x) = {:.6f}'.format(k, x, fun(x)))
        if not damp:
            if x == x0:
                print('[err] 牛顿法发生跳动, x 重复, 请更换初值')
                break
        else:
            i = 0
            lamb_i = lamb
            while np.abs(fun(x)) > np.abs(fun(xk)):
                x = xk - lamb_i * s
                lamb_i *= 0.9
                i += 1
                print('- lambda {:.5f}, x = {:.6f}, f(x) = {:.6f}'.format(lamb_i
            if i > 0:
                print('         x = {:.6f}, f(x) = {:.6f}'.format(x, fun(x)))
        k += 1
    return x
```

对两个函数，分别使用牛顿法、牛顿下山法、scipy.optimize.root方法求根

In [ ]:
```python
from scipy.optimize import root

def test(fun, derive, x0):
    print('牛顿法')
    r1 = newton(fun, derive, x0)
    print('\n牛顿下山法')
    r2 = newton(fun, derive, x0, True)
    print('\nscipy.optimize.root 方法')
    r3 = root(fun, int(r2))
    print('Success: {}'.format(r3.success))
    print('\n牛顿法: {}, 牛顿下山法: {}, scipy.optimize.root 方法: {}'.format(r1

def f1(x):
    return x ** 3 - 2 * x + 2

def f1_derive(x):
    return 3 * x ** 2 - 2

def f2(x):
    return - x ** 3 + 5 * x

def f2_derive(x):
    return -3 * x ** 2 + 5
```

In [ ]:
```python
test(f1, f1_derive, 0)
```

牛顿法

```
Step  0: x = 1.000000, f(x) = 1.000000
Step  1: x = 0.000000, f(x) = 2.000000
[err] 牛顿法发生跳动, x 重复, 请更换初值
```

牛顿下山法

```
Step  0: x = 1.000000, f(x) = 1.000000
Step  1: x = 0.000000, f(x) = 2.000000
- lambda 0.81000, x = 0.100000, f(x) = 1.801000
- lambda 0.72900, x = 0.190000, f(x) = 1.626859
- lambda 0.65610, x = 0.271000, f(x) = 1.477903
- lambda 0.59049, x = 0.343900, f(x) = 1.352872
- lambda 0.53144, x = 0.409510, f(x) = 1.249654
- lambda 0.47830, x = 0.468559, f(x) = 1.165753
- lambda 0.43047, x = 0.521703, f(x) = 1.098588
- lambda 0.38742, x = 0.569533, f(x) = 1.045672
- lambda 0.34868, x = 0.612580, f(x) = 1.004714
- lambda 0.31381, x = 0.651322, f(x) = 0.973660
         x = 0.651322, f(x) = 0.973660
Step  2: x = 1.989979, f(x) = 5.900396
- lambda 0.81000, x = 1.856114, f(x) = 4.682378
- lambda 0.72900, x = 1.735634, f(x) = 3.757203
- lambda 0.65610, x = 1.627203, f(x) = 3.054086
- lambda 0.59049, x = 1.529615, f(x) = 2.519644
- lambda 0.53144, x = 1.441786, f(x) = 2.113535
- lambda 0.47830, x = 1.362739, f(x) = 1.805208
- lambda 0.43047, x = 1.291597, f(x) = 1.571479
- lambda 0.38742, x = 1.227570, f(x) = 1.394719
- lambda 0.34868, x = 1.169945, f(x) = 1.261497
- lambda 0.31381, x = 1.118083, f(x) = 1.161560
- lambda 0.28243, x = 1.071407, f(x) = 1.087067
- lambda 0.25419, x = 1.029398, f(x) = 1.032016
- lambda 0.22877, x = 0.991590, f(x) = 0.991802
- lambda 0.20589, x = 0.957564, f(x) = 0.962890
         x = 0.957564, f(x) = 0.962890
Step  3: x = -0.324949, f(x) = 2.615586
- lambda 0.81000, x = -0.196698, f(x) = 2.385785
- lambda 0.72900, x = -0.081272, f(x) = 2.162006
- lambda 0.65610, x = 0.022612, f(x) = 1.954788
- lambda 0.59049, x = 0.116107, f(x) = 1.769351
- lambda 0.53144, x = 0.200253, f(x) = 1.607525
- lambda 0.47830, x = 0.275984, f(x) = 1.469053
- lambda 0.43047, x = 0.344142, f(x) = 1.352474
- lambda 0.38742, x = 0.405484, f(x) = 1.255701
- lambda 0.34868, x = 0.460692, f(x) = 1.176392
- lambda 0.31381, x = 0.510379, f(x) = 1.112189
- lambda 0.28243, x = 0.555098, f(x) = 1.060849
- lambda 0.25419, x = 0.595344, f(x) = 1.020322
- lambda 0.22877, x = 0.631566, f(x) = 0.988784
- lambda 0.20589, x = 0.664166, f(x) = 0.964643
- lambda 0.18530, x = 0.693506, f(x) = 0.946530
         x = 0.693506, f(x) = 0.946530
Step  4: x = 2.392385, f(x) = 10.908057
- lambda 0.81000, x = 2.222497, f(x) = 8.533012
- lambda 0.72900, x = 2.069598, f(x) = 6.725378
- lambda 0.65610, x = 1.931989, f(x) = 5.347324
- lambda 0.59049, x = 1.808140, f(x) = 4.295201
- lambda 0.53144, x = 1.696677, f(x) = 3.490890
- lambda 0.47830, x = 1.596360, f(x) = 2.875386
- lambda 0.43047, x = 1.506074, f(x) = 2.404019
```

```
- lambda 0.38742, x = 1.424817, f(x) = 2.042894
- lambda 0.34868, x = 1.351686, f(x) = 1.766233
- lambda 0.31381, x = 1.285868, f(x) = 1.554391
- lambda 0.28243, x = 1.226632, f(x) = 1.392358
- lambda 0.25419, x = 1.173319, f(x) = 1.268644
- lambda 0.22877, x = 1.125338, f(x) = 1.174436
- lambda 0.20589, x = 1.082155, f(x) = 1.102957
- lambda 0.18530, x = 1.043290, f(x) = 1.048993
- lambda 0.16677, x = 1.008311, f(x) = 1.008519
- lambda 0.15009, x = 0.976831, f(x) = 0.978429
- lambda 0.13509, x = 0.948498, f(x) = 0.956319
- lambda 0.12158, x = 0.922999, f(x) = 0.940330
         x = 0.922999, f(x) = 0.940330
Step  5: x = -0.768907, f(x) = 3.083222
- lambda 0.81000, x = -0.599716, f(x) = 2.983739
- lambda 0.72900, x = -0.447445, f(x) = 2.805308
- lambda 0.65610, x = -0.310400, f(x) = 2.590894
- lambda 0.59049, x = -0.187060, f(x) = 2.367575
- lambda 0.53144, x = -0.076054, f(x) = 2.151669
- lambda 0.47830, x = 0.023851, f(x) = 1.952312
- lambda 0.43047, x = 0.113766, f(x) = 1.773941
- lambda 0.38742, x = 0.194689, f(x) = 1.618001
- lambda 0.34868, x = 0.267520, f(x) = 1.484106
- lambda 0.31381, x = 0.333068, f(x) = 1.370813
- lambda 0.28243, x = 0.392061, f(x) = 1.276142
- lambda 0.25419, x = 0.445155, f(x) = 1.197904
- lambda 0.22877, x = 0.492939, f(x) = 1.133900
- lambda 0.20589, x = 0.535945, f(x) = 1.082053
- lambda 0.18530, x = 0.574651, f(x) = 1.040462
- lambda 0.16677, x = 0.609485, f(x) = 1.007436
- lambda 0.15009, x = 0.640837, f(x) = 0.981500
- lambda 0.13509, x = 0.669053, f(x) = 0.961383
- lambda 0.12158, x = 0.694448, f(x) = 0.946007
- lambda 0.10942, x = 0.717303, f(x) = 0.934463
         x = 0.717303, f(x) = 0.934463
Step  6: x = 2.764633, f(x) = 17.601358
- lambda 0.81000, x = 2.559900, f(x) = 13.655444
- lambda 0.72900, x = 2.375640, f(x) = 10.656037
- lambda 0.65610, x = 2.209806, f(x) = 8.371410
- lambda 0.59049, x = 2.060556, f(x) = 6.627783
- lambda 0.53144, x = 1.926231, f(x) = 5.294556
- lambda 0.47830, x = 1.805338, f(x) = 4.273362
- lambda 0.43047, x = 1.696534, f(x) = 3.489945
- lambda 0.38742, x = 1.598611, f(x) = 2.888121
- lambda 0.34868, x = 1.510480, f(x) = 2.425277
- lambda 0.31381, x = 1.431163, f(x) = 2.069020
- lambda 0.28243, x = 1.359777, f(x) = 1.794663
- lambda 0.25419, x = 1.295529, f(x) = 1.583352
- lambda 0.22877, x = 1.237707, f(x) = 1.420651
- lambda 0.20589, x = 1.185666, f(x) = 1.295482
- lambda 0.18530, x = 1.138830, f(x) = 1.199327
- lambda 0.16677, x = 1.096677, f(x) = 1.125620
- lambda 0.15009, x = 1.058740, f(x) = 1.069293
- lambda 0.13509, x = 1.024596, f(x) = 1.026426
- lambda 0.12158, x = 0.993867, f(x) = 0.993979
- lambda 0.10942, x = 0.966210, f(x) = 0.969597
- lambda 0.09848, x = 0.941320, f(x) = 0.951448
- lambda 0.08863, x = 0.918918, f(x) = 0.938108
- lambda 0.07977, x = 0.898756, f(x) = 0.928469
         x = 0.898756, f(x) = 0.928469
```

```
Step  7: x = -1.294709, f(x) = 2.419134
- lambda 0.81000, x = -1.075363, f(x) = 2.907171
- lambda 0.72900, x = -0.877951, f(x) = 3.079179
- lambda 0.65610, x = -0.700280, f(x) = 3.057148
- lambda 0.59049, x = -0.540376, f(x) = 2.922959
- lambda 0.53144, x = -0.396463, f(x) = 2.730609
- lambda 0.47830, x = -0.266941, f(x) = 2.514861
- lambda 0.43047, x = -0.150371, f(x) = 2.297343
- lambda 0.38742, x = -0.045459, f(x) = 2.090823
- lambda 0.34868, x = 0.048963, f(x) = 1.902192
- lambda 0.31381, x = 0.133942, f(x) = 1.734519
- lambda 0.28243, x = 0.210424, f(x) = 1.588470
- lambda 0.25419, x = 0.279257, f(x) = 1.463264
- lambda 0.22877, x = 0.341207, f(x) = 1.357310
- lambda 0.20589, x = 0.396962, f(x) = 1.268629
- lambda 0.18530, x = 0.447141, f(x) = 1.195117
- lambda 0.16677, x = 0.492303, f(x) = 1.134710
- lambda 0.15009, x = 0.532948, f(x) = 1.085479
- lambda 0.13509, x = 0.569529, f(x) = 1.045676
- lambda 0.12158, x = 0.602452, f(x) = 1.013755
- lambda 0.10942, x = 0.632082, f(x) = 0.988370
- lambda 0.09848, x = 0.658750, f(x) = 0.968366
- lambda 0.08863, x = 0.682750, f(x) = 0.952762
- lambda 0.07977, x = 0.704351, f(x) = 0.940734
- lambda 0.07179, x = 0.723791, f(x) = 0.931593
- lambda 0.06461, x = 0.741288, f(x) = 0.924768
         x = 0.741288, f(x) = 0.924768
Step  8: x = 3.372379, f(x) = 33.609115
- lambda 0.81000, x = 3.109270, f(x) = 25.840518
- lambda 0.72900, x = 2.872472, f(x) = 19.956095
- lambda 0.65610, x = 2.659354, f(x) = 15.488669
- lambda 0.59049, x = 2.467547, f(x) = 12.089276
- lambda 0.53144, x = 2.294921, f(x) = 9.496733
- lambda 0.47830, x = 2.139558, f(x) = 7.515153
- lambda 0.43047, x = 1.999731, f(x) = 5.997308
- lambda 0.38742, x = 1.873886, f(x) = 4.832287
- lambda 0.34868, x = 1.760627, f(x) = 3.936348
- lambda 0.31381, x = 1.658693, f(x) = 3.246112
- lambda 0.28243, x = 1.566952, f(x) = 2.713495
- lambda 0.25419, x = 1.484386, f(x) = 2.301926
- lambda 0.22877, x = 1.410076, f(x) = 1.983522
- lambda 0.20589, x = 1.343197, f(x) = 1.736973
- lambda 0.18530, x = 1.283006, f(x) = 1.545951
- lambda 0.16677, x = 1.228834, f(x) = 1.397913
- lambda 0.15009, x = 1.180080, f(x) = 1.283206
- lambda 0.13509, x = 1.136201, f(x) = 1.194379
- lambda 0.12158, x = 1.096709, f(x) = 1.125672
- lambda 0.10942, x = 1.061167, f(x) = 1.072620
- lambda 0.09848, x = 1.029179, f(x) = 1.031758
- lambda 0.08863, x = 1.000390, f(x) = 1.000391
- lambda 0.07977, x = 0.974480, f(x) = 0.976417
- lambda 0.07179, x = 0.951161, f(x) = 0.958200
- lambda 0.06461, x = 0.930173, f(x) = 0.944460
- lambda 0.05815, x = 0.911285, f(x) = 0.934198
- lambda 0.05233, x = 0.894285, f(x) = 0.926631
- lambda 0.04710, x = 0.878985, f(x) = 0.921147
         x = 0.878985, f(x) = 0.921147
Step  9: x = -2.019105, f(x) = -2.193245
- lambda 0.81000, x = -1.729296, f(x) = 0.287195
         x = -1.729296, f(x) = 0.287195
```

```
Step 10: x = -1.770492, f(x) = -0.008874
Step 11: x = -1.769293, f(x) = -0.000008
Step 12: x = -1.769292, f(x) = -0.000000
Step 13: x = -1.769292, f(x) = 0.000000

scipy.optimize.root 方法
Success: True

牛顿法: 0.0, 牛顿下山法: -1.7692923542386314, scipy.optimize.root 方法: -1.769292
3542386312
```

In [ ]:
```
test(f2, f2_derive, 1.35)
```

```
牛顿法
Step  0: x = 10.525668, f(x) = -1113.507269
Step  1: x = 7.124287, f(x) = -325.975011
Step  2: x = 4.910781, f(x) = -93.873337
Step  3: x = 3.516911, f(x) = -25.914942
Step  4: x = 2.709743, f(x) = -6.348134
Step  5: x = 2.336940, f(x) = -1.078004
Step  6: x = 2.242244, f(x) = -0.062019
Step  7: x = 2.236093, f(x) = -0.000254
Step  8: x = 2.236068, f(x) = -0.000000
Step  9: x = 2.236068, f(x) = -0.000000

牛顿下山法
Step  0: x = 10.525668, f(x) = -1113.507269
- lambda 0.81000, x = 9.608102, f(x) = -838.937314
- lambda 0.72900, x = 8.782291, f(x) = -633.454764
- lambda 0.65610, x = 8.039062, f(x) = -479.341330
- lambda 0.59049, x = 7.370156, f(x) = -363.490205
- lambda 0.53144, x = 6.768140, f(x) = -276.192417
- lambda 0.47830, x = 6.226326, f(x) = -210.245240
- lambda 0.43047, x = 5.738694, f(x) = -160.296674
- lambda 0.38742, x = 5.299824, f(x) = -122.363080
- lambda 0.34868, x = 4.904842, f(x) = -93.473901
- lambda 0.31381, x = 4.549358, f(x) = -71.409704
- lambda 0.28243, x = 4.229422, f(x) = -54.508834
- lambda 0.25419, x = 3.941480, f(x) = -41.524526
- lambda 0.22877, x = 3.682332, f(x) = -31.519168
- lambda 0.20589, x = 3.449099, f(x) = -23.785955
- lambda 0.18530, x = 3.239189, f(x) = -17.790739
- lambda 0.16677, x = 3.050270, f(x) = -13.128808
- lambda 0.15009, x = 2.880243, f(x) = -9.492702
- lambda 0.13509, x = 2.727219, f(x) = -6.648199
- lambda 0.12158, x = 2.589497, f(x) = -4.416370
- lambda 0.10942, x = 2.465547, f(x) = -2.660134
         x = 2.465547, f(x) = -2.660134
Step  1: x = 2.264582, f(x) = -0.290613
Step  2: x = 2.236598, f(x) = -0.005298
Step  3: x = 2.236068, f(x) = -0.000002
Step  4: x = 2.236068, f(x) = -0.000000

scipy.optimize.root 方法
Success: True

牛顿法: 2.23606797749979, 牛顿下山法: 2.2360679774998133, scipy.optimize.root 方
法: 2.2360679774997894
```

实验结论

函数 (1) 的牛顿法失败，迭代解在 0, 1 之间跳转。牛顿下山法迭代 13 步，解为 -1.7692923542386314。scipy.optimize.root 解为 -1.7692923542386312。

函数 (2) 的牛顿法迭代 9 步， 解为2.23606797749979。牛顿下山法迭代 4 步，解为 2.2360679774998133。scipy.optimize.root 解为 2.2360679774997894。

可以看出牛顿法存在问题：局部收敛, 依赖于初始解的设定，存在不收敛的情况。牛顿下山法使用一系列下山因子，保证 $|f(x_{k+1})| < |f(x_k)|$ ，在一定程度上防止牛顿法迭代过程发散。

上机题 3

实验内容

利用 2.6.3 节给出的 fzerotx 程序，编程求第一类的零阶贝塞尔函数 $J_0(x)$ 的零点。试求 $J_0(x)$ 的前10个正的零点, 并绘出函数曲线和零点的位置

实验过程

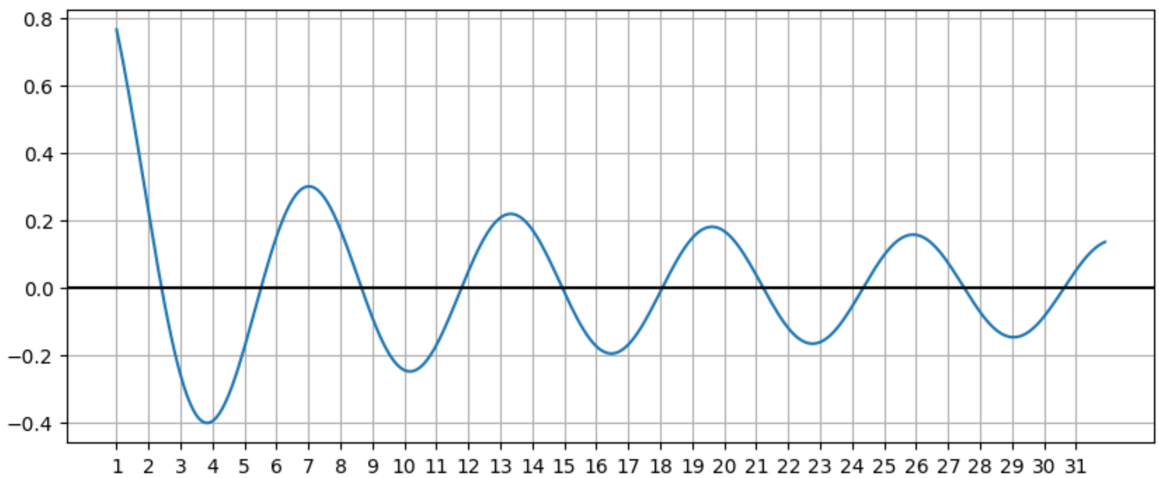
In [ ]:

```
def fzerotx(func, a, b, eps):
        fa, fb = func(a), func(b)
        if np.sign(fa) == np.sign(fb):
                print('Function must change sign on the interval')
                return

        c = a
        fc = fa
        d = b - c
        e = d

        while fb != 0:
                if np.sign(fa) == np.sign(fb):
                        a, fa = c, fc
                        d = b - c
                        e = d

                if abs(fa) < abs(fb):
                        c, fc = b, fb
                        b, fb = a, fa
                        a, fa = c, fc

                m = 0.5 * (a - b)
                tol = 2 * eps * max(abs(b), 1)

                if abs(m) <= tol or fb == 0:     # 收敛测试
                        break

                if abs(e) < tol or abs(fc) <= abs(fb):  # 二分法
                        d = e = m
                else:
                        s = fb / fc
                        if a == c:       # 割线法
                                p = 2 * m * s
```

```
                        q = 1 - s
                else:   # 逆二次插值
                        q = fc / fa
                        r = fb / fa
                        p = s * (2 * m * q * (q - r) - (b - c) * (r - 1)
                        q = (q - 1) * (r - 1) * (s - 1)

                if p > 0:
                        q = -q
                else:
                        p = -p

                if 2 * p < 3 * m * q - abs(tol * q) and p < abs(0.5 * e
                        e = d   # 判断逆二次插值 / 割线法的结果是否可接受
                        d = p / q
                else:
                        d = e = m

        c = b   # 准备下一个迭代步
        fc = fb
        if abs(d) > tol:
                b = b + d
        else:
                b = b - np.sign(b - a) * tol
        fb = func(b)
return b
```

绘制 $J_0(x)$ 曲线，估算 $J_0(x)$ 的前10个正的零点区间

In [ ]:
```
import matplotlib.pyplot as plt
from scipy.special import j0

x = np.arange(1, 32, 0.1)
y = j0(x)

plt.figure(figsize=(10, 4))
ax = plt.subplot()
plt.plot(x, y)
plt.grid(True)
plt.xticks(np.arange(1, 32, 1))
plt.axhline(0, color='black')
plt.show()
```



$J_0(x)$ 的前10个正的零点区间为

In [ ]:
```python
intervals = [
    (2, 3),
    (5, 6),
    (8, 9),
    (11, 12),
    (14, 15),
    (17, 19),
    (21, 22),
    (24, 25),
    (27, 28),
    (30, 31),
]
```

In [ ]:
```python
result = []

for interval in intervals:
    result.append(fzerotx(j0, *interval, 1e-8))

print(result)
```

```
[2.404825568995971, 5.5200781104384005, 8.653727917682568, 11.79153449631245, 14.
930917714231663, 18.071063996037388, 21.21163665052287, 24.352471608227958, 27.49
3479147827202, 30.63460648844871]
```

使用 fzeroin 计算零点并绘制图像

In [ ]:
```python
plt.figure(figsize=(10, 4))
ax = plt.subplot()
plt.plot(x, y)
plt.grid(True)
plt.xticks(np.arange(1, 32, 1))
plt.axhline(0, color='black')
for zero in result:
    plt.scatter(zero, 0)
plt.show()
```