数理演習課題3

最急降下法

```
$ python gradient_descent.py
== 最急降下法-1 ==
initial =  [0, 0, 0, 0]
f(x) =  111.37502690343712
[x1, x2, x3, x4] =  [2.2230467787130257, 0.31842608432360897,
-0.6933914143954055, 0.33254621440525983]
count =  3099

initial =  [-2, -1, 1, 2]
f(x) =  111.37502690209087
[x1, x2, x3, x4] =  [2.2230468970864004, 0.31842604030157473,
-0.6933913712408408, 0.3325462340996037]
count =  3313

initial =  [100, 200, 300, 400]
f(x) =  111.37502688479191
[x1, x2, x3, x4] =  [2.232507137168756, 0.3149078587902273,
-0.689942516702497, 0.3341201794610436]
count =  5608

== 最急降下法-2==
initial =  [0, 0, 0, 0]
f(x) =  -13.626788385938362
[x1, x2, x3, x4] =  [-3.4182403874698757, 5.225297552972747,
-4.9909652265925075, -5.481984012591073]
count =  18743

initial =  [-2, -1, 1, 2]
f(x) =  -13.626788339291316
[x1, x2, x3, x4] =  [-3.418239098718801, 5.225294940684059,
-4.99096262034507, -5.4819814620914]
count =  19264

initial =  [100, 200, 300, 400]
f(x) =  -13.626788337973949
[x1, x2, x3, x4] =  [-3.4182390623264873, 5.225294866917113,
-4.990962546748718, -5.481981390069281]
count =  30190

== 最急降下法-3 ==
initial =  [0, 0]
f(x) =  0.0001246375129096662
[x1, x2] =  [0.9889260340369366, 0.9775269518978035]
count =  8448

initial =  [-1, 1]
```

```
f(x) =  0.00012466010123508468
[x1, x2] =  [0.9889250307129775, 0.9775249266290891]
count =  9149

initial =  [10, 20]
Traceback (most recent call last):
  File "gradient_descent.py", line 131, in <module>
    (result, count) = gradient_descent(test, diff3, 0.001)
  File "gradient_descent.py", line 10, in gradient_descent
    next = next_x(current, diff_fn, alpha)
  File "gradient_descent.py", line 25, in next_x
    next.append(v - alpha * diff_fn(X, i))
  File "gradient_descent.py", line 71, in diff3
    return 40 * pow(X[0], 3) + 2 * X[0] - 40 * X[0] * X[1] - 2
OverflowError: (34, 'Result too large')
```

問１は、`[2.2230468970864004, 0.31842604030157473, -0.6933913712408408, 0.3325462340996037]` に収束した。 イテレーション回数は、3000 ~ 6000回

問２は、`[-3.4182403874698757, 5.225297552972747, -4.9909652265925075, -5.481984012591073]` に収束した。 イテレーション回数は、20000 ~ 30000回

問３は、`[0.9889260340369366, 0.9775269518978035]` に収束した。イテレーション回数は、10000回程度。しかし、初期値によって発散してしまい計算できないことがあった。 `[10, 20]` の場合は、発散しオーバーフローエラーが発生した。

共役勾配法

```
$ python fletcher_reeves.py
== 共役勾配法-1 ==
initial =  [0, 0, 0, 0]
f(x) =  111.37500015499927
[x1, x2, x3, x4] =  [ 2.22747106  0.31678353 -0.69172178  0.33331348]
count =  10

initial =  [-2, -1, 1, 2]
f(x) =  111.37500168732123
[x1, x2, x3, x4] =  [ 2.22662208  0.31707674 -0.69214659  0.33314667]
count =  11

initial =  [100, 200, 300, 400]
f(x) =  111.3753227586285
[x1, x2, x3, x4] =  [ 2.24417331  0.31081194 -0.68609827  0.33548768]
count =  15

== 共役勾配法-2 ==
initial =  [0, 0, 0, 0]
f(x) =  -13.626902587719027
[x1, x2, x3, x4] =  [-3.42194731  5.23364967 -4.99961872 -5.49008581]
count =  21
```

```
initial =  [-2, -1, 1, 2]
f(x) =  -13.626919595924221
[x1, x2, x3, x4] =  [-3.42327676  5.23553933 -5.00141927 -5.49189677]
count =  42

initial =  [100, 200, 300, 400]
f(x) =  -13.626944721768151
[x1, x2, x3, x4] =  [-3.42731763  5.24176063 -5.00675713 -5.49788326]
count =  19

== 共役勾配法-3 ==
initial =  [0, 0]
f(x) =  0.00031289101251119676
[x1, x2] =  [0.98292497 0.96468087]
count =  14

initial =  [-1, 1]
f(x) =  1.2818989709841442e-30
[x1, x2] =  [1. 1.]
count =  2

initial =  [10, 20]
fletcher_reeves.py:115: RuntimeWarning: invalid value encountered in
subtract
  return (X[0] - 1) ** 2 + 10 * (X[0] ** 2 - X[1]) ** 2
fletcher_reeves.py:19: RuntimeWarning: invalid value encountered in
true_divide
  s = -gx1 + s * np.dot(gx1.T, gx1) / np.dot(gx0.T, gx0)
f(x) =  nan
[x1, x2] =  [nan nan]
count =  100
```

共役勾配法でも、問1と2は、最急降下法と同じ値に収束した。イテレーション回数は減ったが、設計変数の数と同じにはならなかった。これは、αの値をラインサーチの手法に問題があったのではないかと考える。ラインサーチで探すときの範囲と間隔を、`-1 ~ 1` の範囲を `0.001` 感覚で行なっていたため十分ではなかったのではないかと考察する。

問3 では、それぞれ最急降下法の解に近い解を出すことはできたが、異なる解に収束した。`[10, 20]` を初期値にした場合は、発散した。100回を超えるイテレーションをしたため途中で終了させた。

準ニュートン法

```
== 準ニュートン法-1 ==
initial =  [0, 0, 0, 0]
f(x) =  111.37500000007378
[x1, x2, x3, x4] =  [ 2.22777688  0.31666556 -0.69166592  0.3333345 ]
count =  9

initial =  [-2, -1, 1, 2]
f(x) =  111.37500000000009
```

```
[x1, x2, x3, x4] =  [ 2.22777752  0.31666677 -0.69166677  0.3333333 ]
count =  10

initial =  [100, 200, 300, 400]
f(x) =  111.37500000070985
[x1, x2, x3, x4] =  [ 2.22775351  0.31667551 -0.69167576  0.33332936]
count =  11

== 準ニュートン法-2 ==
initial =  [0, 0, 0, 0]
f(x) =  -13.626953124621672
[x1, x2, x3, x4] =  [-3.42735623  5.24377732 -5.00940081 -5.50002608]
count =  12

initial =  [-2, -1, 1, 2]
f(x) =  -13.626953124999929
[x1, x2, x3, x4] =  [-3.42734375  5.24375    -5.009375   -5.5       ]
count =  9

initial =  [100, 200, 300, 400]
f(x) =  -13.626953125000039
[x1, x2, x3, x4] =  [-3.42734375  5.24375    -5.009375   -5.5       ]
count =  9

== 準ニュートン-3 ==
initial =  [0, 0]
f(x) =  1.6536615392938503e-09
[x1, x2] =  [0.99995934 0.99991889]
count =  19

initial =  [-1, 1]
f(x) =  4.409348198805778e-10
[x1, x2] =  [0.99998794 0.99997045]
count =  38

initial =  [10, 20]
f(x) =  80.97750192220795
[x1, x2] =  [ 9.99875002 99.975016  ]
count =  6
```

準ニュートン法でも、問1と2は、最急降下法と同じ値に収束した。イテレーション回数は共役勾配法の時よりも減った。

問３は、最急降下法、共役勾配法よりも小さい最小解に収束した。しかし、`[10, 20]` を初期値にした時は、全く違う解に収束した。