科学計算研究室 Python ゼミ　フォローアップ

～2. Bairstow 法～

2021-02-24 福田　浩

1　Python スクリプト

基本的には C++のソースファイルを 1 行毎 Python に変換しているので，C++で使っている閉じカッコの数だけ Python のコードは行数が減少．Python の変換する上での注意点は以下．

- Python は複数の返り値を一括で `return` 出来るので，ポインタを使わずに下記のように記述．
  - CALLER(呼び出し元) `x, y, z = ユーザ定義関数名 (a, b, c)`
  - CALLEE(呼び出し先) `return x, y, z`
- C++で `for (int i=2;i<n+1;i++)` は，Python では `for i in range(2,n+1):` で表現できる．
- 0 ～ 1 の一様乱数は `random` パッケージを `import` し，`random.random()` で導入できる．

```
 1  import random
 2  import math
 3
 4  def quadratic(p, q):
 5      # Quadratic equation solver
 6      if p*p-4*q>0:
 7          print(-p/2 + math.sqrt(p*p-4*q)/2)
 8          print(-p/2 - math.sqrt(p*p-4*q)/2)
 9      else:
10          print(-p/2, "+", math.sqrt(-p*p+4*q)/2, "i")
11          print(-p/2, "-", math.sqrt(-p*p+4*q)/2, "i")
12
13  def bs(n, a, p, q):
14      # Bairstow method
15      EPS = 0.00001
16      p = random.random() # initialize p as a random value
17      q = random.random() # initialize q as a random value
18      while 1:
19          b[0] = 1 # !!warning !! potential bug
20          b[1] = a[1]-p*a[0]
21          for i in range(2,n+1):
22              b[i] =  a[i]   -p*b[i-1] -q*b[i-2]
23          c[1] = -1
24          c[2] = -b[1]-p*c[1]
25          for i in range(3,n+1):
26              c[i] = -b[i-1] -p*c[i-1] -q*c[i-2]
27          d[1] = 0
28          d[2] = -1
29          for i in range(3,n+1):
```

```
30              d[i] = -b[i-2] -p*d[i-1] -q*d[i-2]
31          dp = (-b[n-1]*d[n]+b[n]*d[n-1])         /(c[n-1]*d[n]-d[n-1]*(c[n]+b[n-1]))
32          dq = (-b[n]*c[n-1]+b[n-1]*(c[n]+b[n-1]))/(c[n-1]*d[n]-d[n-1]*(c[n]+b[n-1]))
33          if math.fabs(dp)>EPS or math.fabs(dq)>EPS:
34              p += dp
35              q += dq
36          else:
37              break
38      for i in range(0,n):
39          a[i]=b[i]
40      return n, a, p, q
41
42  a = [0,0,0,0,0,0,0,0,0,0]
43  b = [0,0,0,0,0,0,0,0,0,0]
44  c = [0,0,0,0,0,0,0,0,0,0]
45  d = [0,0,0,0,0,0,0,0,0,0]
46  p = 0
47  q = 0
48  n = int(input())
49  for i in range(n):
50      a[i] = int(input())
51  n = n-1
52  while 1:
53      if n==0:
54          break
55      if n==1:
56          print(-a[1]/a[0])
57          break
58      if n==2:
59          quadratic(a[1], a[2]) # !!warning !! potential bug
60          break
61      if n>2:
62          n, a, p, q = bs(n, a, p, q)
63          quadratic(p, q) # !!warning !! potential bug
64          n += -2
```

2　課題解答

$x^4 + 4x^3 + 8x^2 + 8x - 5 = 0$ の解答

```
5
1
4
8
8
-5
0.4142136373045954
-2.4142135624523973
-1.000000037426099 + 2.0000000265203246 i
-1.000000037426099 - 2.0000000265203246 i
```

$x^4 + 4x^3 + 8x^2 + 8x - 6 = 0$ の解答

```
5
1
4
8
8
-6
0.4704685175120369
-2.470474905372696
-0.9999968060696705 + 2.040168388680857 i
-0.9999968060696705 - 2.040168388680857 i
```

$x^5 + x^4 + x^3 + x^2 + x + 1 = 0$ の解答

```
6
1
1
1
1
1
1
0.4999999524753935 + 0.8660253398041317 i
0.4999999524753935 - 0.8660253398041317 i
-0.4999992131148653 + 0.8660253540929682 i
-0.4999992131148653 - 0.8660253540929682 i
-1.0000014787210565
```

3　潜在的バグとその対応

C++のソースコードでも，Python のソースコードでも `a[0]` を `1` としているところが元凶．例えば $x^2 - 2x + 1 = 0$ は解けるが，数学的に等価な方程式 $2x^2 - 4x + 2 = 0$ はまともに解けない．対応例はすべての係数 `a[i]` を `a[0]` で割って規格化してしまうこと．前記ソースコードの 50 行目に以下の 3 行を追加すれば良い．

```
1  aa = a[0]
2  for i in range(n):
3      a[i] = a[i]/aa
```

4　折角なので可視化

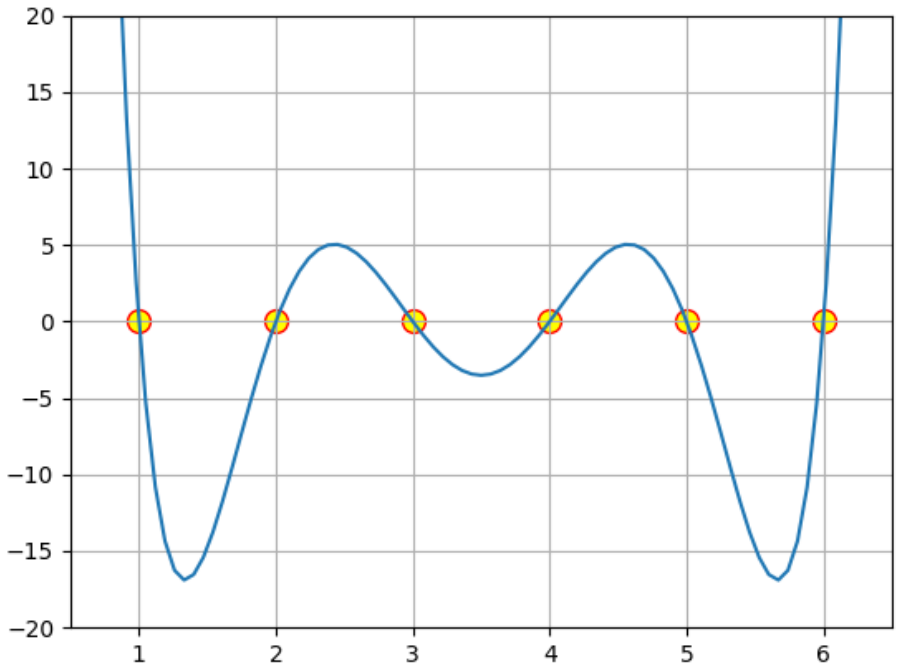

Figure 1: $y = x^6 - 21x^5 + 175x^4 - 735x^3 + 1624x^2 - 1764x + 720$ のグラフ