1-1 P136 6

In [1]:

```
import math

x = [0.5]
i = 0

def g(x):
    return 4 + math.cos(x)*2/3

while math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [0.5, 4.585055041260248, 3.915339919553916, 3.523135312410598, 3.381272448
6310506, 3.352390640923876, 3.3480904966766905, 3.3474966835384796, 3.34741563089545
44, 3.3474045854461716]
精确值: 3.34740458545

1-2 P136 7

In [3]:

```
import math

x = [0.5]
i = 0

def g(x):
    return 1 / (1 + x)

while math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
print (math.sqrt(5) - 1) / 2
```

迭代过程: [0.5, 0.6666666666666666, 0.6000000000000001, 0.625, 0.6153846153846154,
0.6190476190476191, 0.6176470588235294, 0.6181818181818182, 0.6179775280898876, 0.61
80555555555556, 0.6180257510729613, 0.6180371352785146]
精确值: 0.618037135279
0.61803398875

1-3 P136 10

In [4]:

```
import math

x = [1.5]
i = 0

def g(x):
    return math.log(4 - x) / math.log(2)

while math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [1.5, 1.3219280948873624, 1.421194696941818, 1.366702855362277, 1.39687032613424, 1.3802471808097676, 1.3894306959797775, 1.3843644591030435, 1.3871615316726822, 1.3856179366952321, 1.3864699906196314, 1.3859997248748772, 1.3862592929778867, 1.386116027419562, 1.386195102916846, 1.3861514576921052, 1.386175547563411, 1.3861622512696667]
精确值: 1.38616225127

1-4 P137 17

Newton 法

In [5]:

```
import math

x = [math.pi / 4]
i = 0

def g(x):
    return x - (f(x) / f_(x))

def f(x):
    return math.exp(x) - 4 * math.cos(x)

def f_(x):
    return math.exp(x) + 4 * math.sin(x)

while f_(x[i]) != 0.0 and math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [0.7853981633974483, 0.9118784721476161, 0.9048101871468585, 0.9047882180853895]
精确值: 0.904788218085

弦截法

In [9]:

```
import math

x = [math.pi / 4, math.pi / 4 + 0.0001]
i = 1

def g(x):
    return x - (f(x) * (x - x0) / (f(x) - f(x0)))

def f(x):
    return math.exp(x) - 4 * math.cos(x)

while math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [0.7853981633974483, 0.7854981633974483, 0.9118721484748704, 0.9043946276039738, 0.9048101698844864, 0.9047869937871547]
精确值: 0.904786993787

快速弦截法

In [11]:

```
import math

x = [math.pi / 4, math.pi / 2]
i = 2

def g(i):
    return x[i - 1] - (f(x[i - 1]) * (x[i - 1] - x[i - 2]) / (f(x[i - 1]) - f(x[i - 2])))

def f(x):
    return math.exp(x) - 4 * math.cos(x)

while math.fabs(x[i - 1] - g(i)) >= 1e-5:
    x.append(g(i))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [0.7853981633974483, 1.5707963267948966, 0.8770025972944069, 0.8985446421856659, 0.9048658349261991, 0.9047880039957831]
精确值: 0.904788003996

1-5 P137 18

Newton 法

In [12]:

```
import math

x = [1.3]
i = 0

def g(x):
    return x - (f(x) / f_(x))

def f(x):
    return x*x*x + 2*x*x + 10*x - 20

def f_(x):
    return 3*x*x + 4*x + 10

while f_(x[i]) != 0.0 and math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [1.3, 1.3702022693635916, 1.3688086702377473]
精确值: 1.36880867024

弦截法

In [13]:

```
import math

x = [1.3, 1.3 + 0.0001]
i = 1

def g(x):
    return x - (f(x) * (x - x0) / (f(x) - f(x0)))

def f(x):
    return x*x*x + 2*x*x + 10*x - 20

while math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [1.3, 1.3001, 1.3813021530440508, 1.3665593821049666, 1.3692136076263877,
1.3687350111854906, 1.3688212852515538, 1.3688057322982383]
精确值: 1.3688057323

快速弦截法

In [14]:

```
import math

x = [1.3, 1.3 + 0.0001]
i = 2

def g(i):
    return x[i - 1] - (f(x[i - 1]) * (x[i - 1] - x[i - 2]) / (f(x[i - 1]) - f(x[i - 2])))

def f(x):
    return x*x*x + 2*x*x + 10*x - 20

while math.fabs(x[i - 1] - g(i)) >= 1e-5:
    x.append(g(i))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [1.3, 1.3001, 1.3702002260072024, 1.368780188645968, 1.3688080965730385]
精确值: 1.36880809657

2 P134 例5

In [1]:

```
import math

x = [0.5]
i = 0

def f(x):
    return math.exp(-1 * x)

def g(x):
    return math.exp(-1 * f(x))

def h(x):
    return g(x) - (g(x) - f(x))*(g(x) - f(x)) / (g(x) - 2*f(x) + x)

while i < 5 and math.fabs(x[i] - h(x[i])) >= 1e-5:
    x.append(g(i))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [0.5, 0.36787944117144233, 0.6922006275553464, 0.8734230184931167, 0.95143
19929004534, 0.9818510730616665]
精确值: 0.981851073062

In [ ]:

```

```