Table of Contents

試験解答例18

簡単な行列計算

In [46]:
```
import numpy as np
from pprint import pprint
aa = np.array([[4,-1,-1], [1,2,-1],[3,-1,0]])
pprint(aa)
```

```
array([[ 4, -1, -1],
       [ 1,  2, -1],
       [ 3, -1,  0]])
```

In [47]:
```
import scipy.linalg as linalg
inv_a = linalg.inv(aa)
pprint(inv_a)
```

```
array([[-0.16666667,  0.16666667,  0.5       ],
       [-0.5       ,  0.5       ,  0.5       ],
       [-1.16666667,  0.16666667,  1.5       ]])
```

In [48]:
```
pprint(np.dot(inv_a,aa))
```

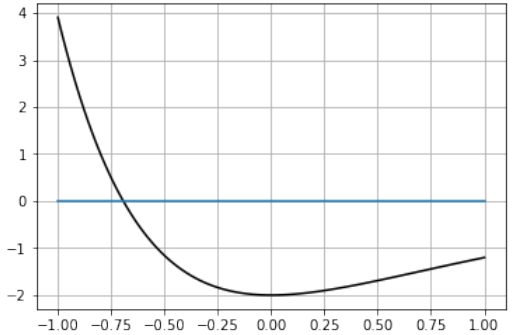
```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

数値解の収束性

In [49]:
```
import numpy as np

def func(x):
    return -4*np.exp(-x)+2*np.exp(-2*x)

def df(x):
    return 4*np.exp(-x) - 4*np.exp(-2*x)

from scipy.optimize import fsolve
x0 = fsolve(func, -1.0)[0]
print(x0)
```

```
-0.69314718056
```

In [50]:
```
import matplotlib.pyplot as plt

x1=-1.0
x2=1.0
x = np.linspace(x1, x2, 100)
y = func(x)
plt.plot(x, y, color = 'k')
plt.plot([x1,x2],[0,0])
plt.grid()
plt.show()
```



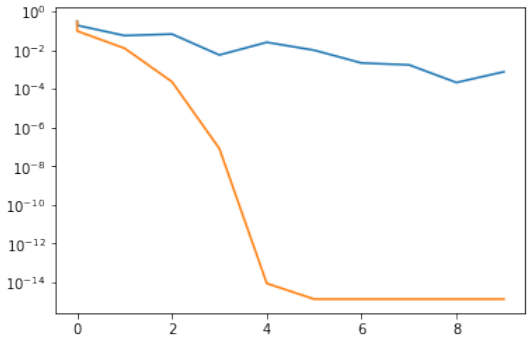
In [51]:
```
x1, x2 = -1.0, 0.0
f1, f2 = func(x1), func(x2)
print('%+15s %+15s %+15s %+15s'  % ('x1','x2','f1','f2'))
print('%+15.10f %+15.10f %+15.10f %+15.10f' % (x1,x2,f1,f2))

list_bisec = [[0],[abs(x1-x0)]]
for i in range(0, 10):
    x = (x1 + x2)/2
    f = func(x)
    if (f*f1>=0.0):
        x1, f1 = x, f
        list_bisec[0].append(i)
        list_bisec[1].append(abs(x1-x0))
    else:
        x2, f2 = x, f
        list_bisec[0].append(i)
        list_bisec[1].append(abs(x2-x0))

    print('%+15.10f %+15.10f %+15.10f %+15.10f' % (x1,x2,f1,f2))


print(list_bisec)
```

```
             x1              x2              f1              f2
  -1.0000000000   +0.0000000000   +3.9049848840   -2.0000000000
  -1.0000000000   -0.5000000000   +3.9049848840   -1.1583214259
  -0.7500000000   -0.5000000000   +0.4953780742   -1.1583214259
  -0.7500000000   -0.6250000000   +0.4953780742   -0.4922979148
  -0.7500000000   -0.6875000000   +0.4953780742   -0.0447964325
  -0.7187500000   -0.6875000000   +0.2128474183   -0.0447964325
  -0.7031250000   -0.6875000000   +0.0810265592   -0.0447964325
  -0.6953125000   -0.6875000000   +0.0173789137   -0.0447964325
  -0.6953125000   -0.6914062500   +0.0173789137   -0.0138911236
  -0.6933593750   -0.6914062500   +0.0016980959   -0.0138911236
  -0.6933593750   -0.6923828125   +0.0016980959   -0.0061079375
[[0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0.30685281944005338, 0.193147
18055994662, 0.056852819440053382, 0.068147180559946618, 0.0056471
805599466185, 0.025602819440053382, 0.0099778194400533815, 0.00216
53194400533815, 0.0017409305599466185, 0.00021219444005338151, 0.0
0076436805994661849]]
```


In [52]:
```
x1 = -1.0
f1 = func(x1)
list_newton = [[0],[abs(x1-x0)]]
print('%-15.10f %+24.25f' % (x1,f1))
for i in range(0, 10):
    x1 = x1 - f1 / df(x1)
    f1 =func(x1)
    print('%-15.10f %+24.25f' % (x1,f1))
    list_newton[0].append(i)
    list_newton[1].append(abs(x1-x0))

print(list_newton)
```

```
-1.0000000000   +3.90498488402512045070125 17
-0.7909883534   +0.90682330520595222367319 37
-0.7057281263   +0.10256563939231178039790 39
-0.6933803632   +0.00186611397431768466503 85
-0.6931472621   +0.00000065227027867820197 57
-0.6931471806   +0.00000000000007993605777 30
-0.6931471806   +0.00000000000000000000000 00
-0.6931471806   +0.00000000000000000000000 00
-0.6931471806   +0.00000000000000000000000 00
-0.6931471806   +0.00000000000000000000000 00
-0.6931471806   +0.00000000000000000000000 00
[[0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0.30685281944005338, 0.097841
172874716609, 0.012580945692322598, 0.00023318267075744803, 8.1533
773510500396e-08, 8.659739592076221e-15, 1.3322676295501878e-15, 1
.3322676295501878e-15, 1.3322676295501878e-15, 1.3322676295501878e
-15, 1.3322676295501878e-15]]
```

In [53]:
```
import matplotlib.pyplot as plt

X = list_bisec[0]
Y = list_bisec[1]
plt.plot(X, Y)

X = list_newton[0]
Y = list_newton[1]
plt.plot(X, Y)

plt.yscale("log") # y軸を対数目盛に
plt.show()
```



精度，誤差

In [54]:
```
from decimal import *

def pretty_p(result,a,b,operator):
    print('context.prec:{}'.format(getcontext().prec))
    print(' %20.14f' % (a))
    print( '%1s%20.14f' % (operator, b))
    print('-----------')
    print( ' %20.14f' % (result))
```


In [57]:
```
getcontext().prec = 5

aa = '0.80000'
bb = '3.1415'
cc = '3.1234'
a=Decimal(aa)
b=Decimal(bb)
c=Decimal(cc)
pretty_p(b-c,b,c,'-')

print(b-c)
print(a/(b-c))
```

```
context.prec:5
      3.14150000000000
-     3.12340000000000
-----------
      0.01810000000000
0.0181
44.199
```

In [59]:
```
TWOPLACES = Decimal(10) ** -3
getcontext().prec = 4
a=Decimal(aa).quantize(Decimal(10) ** -4)
b=Decimal(bb).quantize(Decimal('0.001'))
c=Decimal(cc).quantize(Decimal('0.001'))

pretty_p(b-c,b,c,'-')

print(b-c)
print(a/(b-c))
```

```
context.prec:4
      3.14200000000000
-     3.12300000000000
-----------
      0.01900000000000
0.019
42.11
```

In [60]:
```
TWOPLACES = Decimal(10) ** -2
getcontext().prec = 3
a=Decimal(aa).quantize(Decimal(10) ** -3)
b=Decimal(bb).quantize(Decimal('0.01'))
c=Decimal(cc).quantize(Decimal('0.01'))

pretty_p(b-c,b,c,'-')

print(b-c)
print(a/(b-c))
```

```
context.prec:3
     3.14000000000000
-    3.12000000000000
-----------
     0.02000000000000
0.02
40.0
```

In [61]:
```
TWOPLACES = Decimal(10) ** -1
getcontext().prec = 2
a=Decimal(aa).quantize(Decimal(10) ** -2)
b=Decimal(bb).quantize(Decimal('0.1'))
c=Decimal(cc).quantize(Decimal('0.1'))

pretty_p(b-c,b,c,'-')

print(b-c)
print(a/(b-c))
```

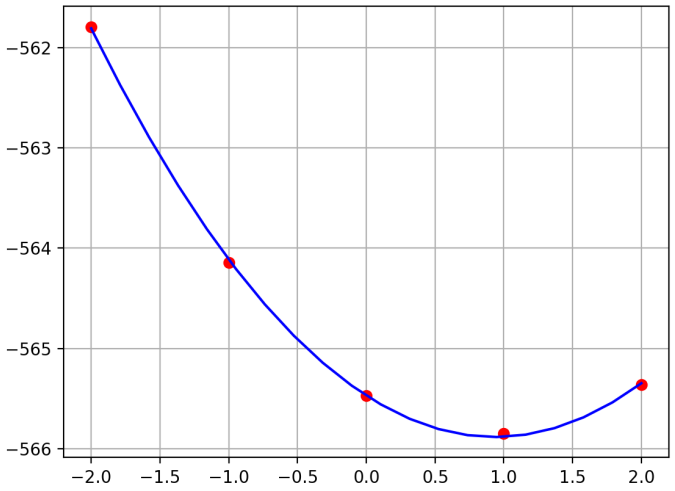
```
context.prec:2
     3.10000000000000
-    3.10000000000000
-----------
     0.00000000000000
0.0

---------------------------------------------------------------------
---------
DivisionByZero                            Traceback (most recent c
all last)
<ipython-input-61-c01b8733d81c> in <module>
      8
      9 print(b-c)
---> 10 print(a/(b-c))

DivisionByZero: [<class 'decimal.DivisionByZero'>]
```

最小2乗法

In [68]:
```
import numpy as np
from pprint import pprint

aa = np.array([[-2.0, -561.78952],
               [-1.0, -564.14261],
               [0.0, -565.47273],
               [1.0, -565.8513],
               [2.0, -565.36457]])

at = np.transpose(aa)
print(at)
```

```
[[  -2.        -1.         0.         1.         2.     ]
 [-561.78952 -564.14261 -565.47273 -565.8513  -565.36457]]
```

In [78]:
```
%matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

def f(x, a0, a1, a2):
    return a0 + a1*x + a2*x**2

xdata = np.array(at[0])
ydata = np.array(at[1])
plt.plot(xdata,ydata, 'o', color='r')

params, cov = curve_fit(f, xdata, ydata)
print(params)

x =np.linspace(-2,2,20)
y = f(x,params[0],params[1],params[2])
plt.plot(x,y, color='b')

plt.grid()
plt.show()
```

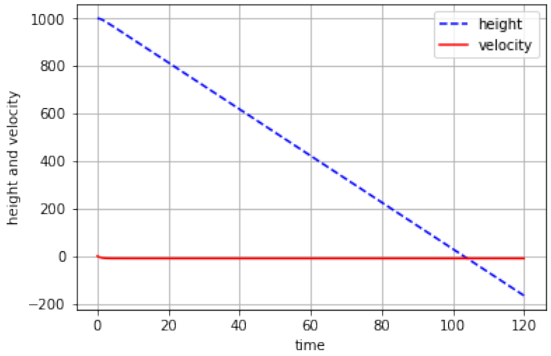



```
[ -5.65471459e+02  -8.85879000e-01   4.73656429e-01]
```

常微分方程式

In [89]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

def my_plot(xx, vv, tt):
    plt.plot(tt, xx, color = 'b', linestyle='--',label="height")
    plt.plot(tt, vv, color = 'r', label="velocity")
    plt.legend()
    plt.xlabel('time')
    plt.ylabel('height and velocity')
    plt.grid()
    plt.show()

def euler2(x0, v0):
  v1 = v0 + (-cc * v0- g) * dt
  x1 = x0 + v0 * dt
  return [x1, v1]
```


In [90]:
```
g, dt, cc=9.8, 0.1, 1.0
# tt,xx,vv=[0.0],[0.0],[-10]
tt,xx,vv=[0.0],[1000.0],[0.0]
t = 0.0
for i in range(0,1200):
  t += dt
  x, v = euler2(xx[-1],vv[-1])
  tt.append(t)
  xx.append(x)
  vv.append(v)

my_plot(xx, vv, tt)
```



In [84]:
```
vv[-1]*3600/1000
```

Out[84]: -35.27999999999968

In [ ]:
```

```