# 전산천문학 HW2 Solution

April 8, 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

## 1. Machine Epsilon ($\epsilon$)

### 1) 첫번째 방법

[2]:
```
def epsilon_loop(datatype):
    epsilon = datatype(1)
    while datatype(1)+datatype(epsilon) != datatype(1):
        epsilon_last = epsilon
        epsilon = datatype(epsilon) / datatype(2)
    return epsilon_last

print (epsilon_loop(np.float16))
print (epsilon_loop(np.float32))
print (epsilon_loop(float))
```

```
0.000977
1.1920929e-07
2.220446049250313e-16
```

### 2) 두번째 방법

[3]:
```
def epsilon_np(datatype):
    return np.finfo(datatype).eps
```

[4]:
```
print (epsilon_np(np.float16))
print (epsilon_np(np.float32))
print (epsilon_np(np.float))
```

```
0.000977
1.1920929e-07
2.220446049250313e-16
```

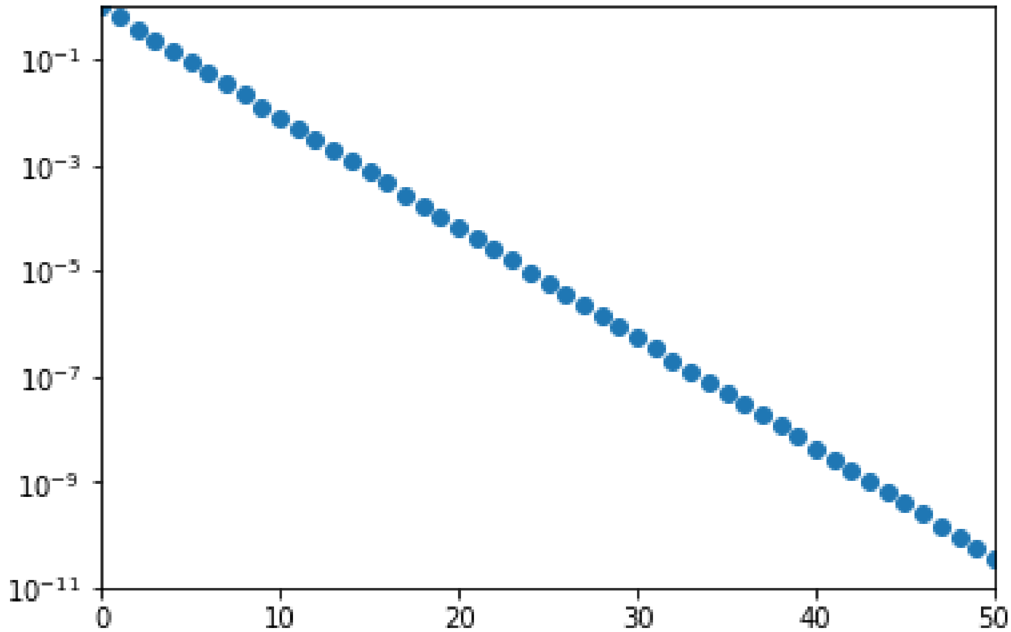
## 2. Golden Mean $\phi$

**(a)**

[5]:
```
gm=0.5*(pow(5.,0.5)-1.)
phi_mul=np.zeros(51)
phi_mul[0]=1.

for n in range(1,51):
    phi_mul[n]=gm*phi_mul[n-1]

print (phi_mul)
```

```
[1.00000000e+00 6.18033989e-01 3.81966011e-01 2.36067977e-01
 1.45898034e-01 9.01699437e-02 5.57280900e-02 3.44418537e-02
 2.12862363e-02 1.31556175e-02 8.13061876e-03 5.02499874e-03
 3.10562002e-03 1.91937873e-03 1.18624129e-03 7.33137436e-04
 4.53103854e-04 2.80033582e-04 1.73070272e-04 1.06963310e-04
 6.61069614e-05 4.08563490e-05 2.52506123e-05 1.56057367e-05
 9.64487568e-06 5.96086099e-06 3.68401469e-06 2.27684629e-06
 1.40716840e-06 8.69677897e-07 5.37490500e-07 3.32187398e-07
 2.05303102e-07 1.26884295e-07 7.84188071e-08 4.84654881e-08
 2.99533190e-08 1.85121692e-08 1.14411498e-08 7.07101942e-09
 4.37013034e-09 2.70088908e-09 1.66924125e-09 1.03164783e-09
 6.37593424e-10 3.94054407e-10 2.43539017e-10 1.50515390e-10
 9.30236269e-11 5.74917632e-11 3.55318637e-11]
```

[6]:
```
plt.figure()
plt.scatter(np.arange(0,51),phi_mul)
plt.xlim(0,50)
plt.ylim(1.0e-11,1.0)
plt.yscale('log')
```




**(b)**

$\phi$는 다음과 같은 방정식의 해에 해당한다.

$$\phi^2 + \phi - 1 = 0 \tag{1}$$

그러므로,

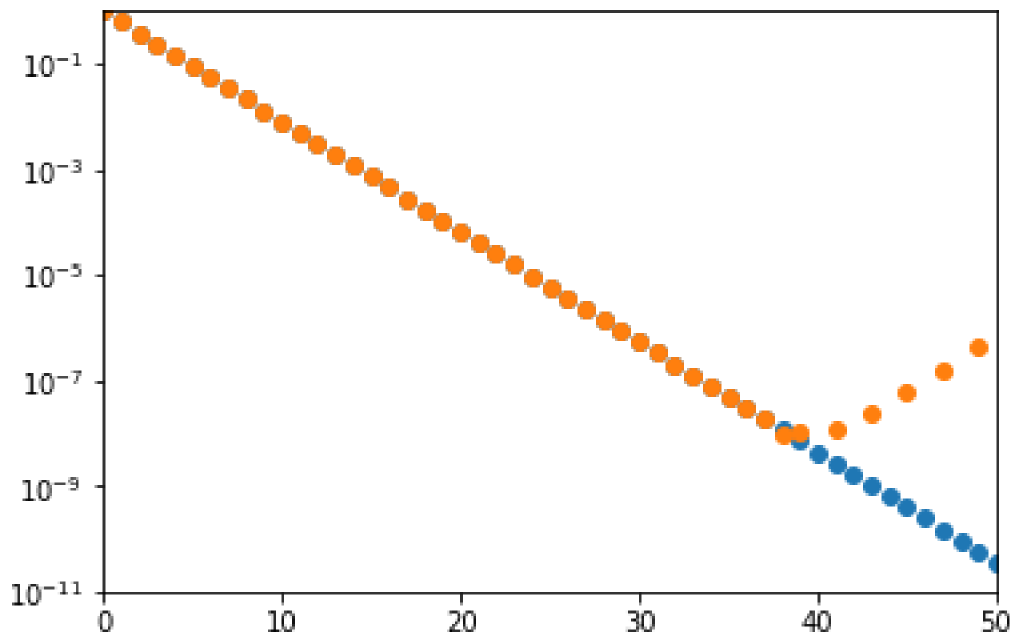
$$\phi^{n+1} = \phi^{n-1} - \phi^n \tag{2}$$

[7]:
```
phi_add=np.zeros(51)
phi_add[0],phi_add[1]=1.,gm

for n in range(2,51):
    phi_add[n]=phi_add[n-2]-phi_add[n-1]

print (phi_add)
```

```
[ 1.00000000e+00  6.18033989e-01  3.81966011e-01  2.36067977e-01
  1.45898034e-01  9.01699437e-02  5.57280900e-02  3.44418537e-02
  2.12862363e-02  1.31556175e-02  8.13061876e-03  5.02499874e-03
  3.10562002e-03  1.91937873e-03  1.18624129e-03  7.33137436e-04
  4.53103854e-04  2.80033582e-04  1.73070272e-04  1.06963311e-04
  6.61069610e-05  4.08563496e-05  2.52506114e-05  1.56057382e-05
  9.64487316e-06  5.96086506e-06  3.68400810e-06  2.27685696e-06
  1.40715113e-06  8.69705831e-07  5.37445302e-07  3.32260528e-07
```

```
 2.05184774e-07  1.27075754e-07  7.81090197e-08  4.89667347e-08
 2.91422850e-08  1.98244496e-08  9.31783539e-09  1.05066142e-08
-1.18877885e-09  1.16953931e-08 -1.28841720e-08  2.45795650e-08
-3.74637370e-08  6.20433021e-08 -9.95070391e-08  1.61550341e-07
-2.61057380e-07  4.22607721e-07 -6.83665101e-07]
```

[8]:
```python
plt.figure()
plt.scatter(np.arange(0,51),phi_mul)
plt.scatter(np.arange(0,51),phi_add)
plt.xlim(0,50)
plt.ylim(1.0e-11,1.0)
plt.yscale('log')
```




**(c)**

$\phi^n$의 오차를 $\epsilon_n$이라 하고, $\tilde{\phi}^n = \phi^n + \epsilon_n$이라 놓자. 이 식을 문제의 식 (2)에 대입하면,

$$\phi^{n+1} + \epsilon_{n+1} = \phi^{n-1} + \epsilon_{n-1} - (\phi^n + \epsilon_n)$$

이다. 따라서 오차는

$$\epsilon_{n+1} = \epsilon_{n-1} - \epsilon_n$$

라는 식을 만족한다.

위 식을 풀기 위하여 오차가 $\epsilon_n = Ar^n$과 같이 공비 $r$의 등비수열로 주어진다고 가정하자($A$는 상수). 이를 위 식에 대입하면

$$r^{n+1} + r^n - r^{n-1} = 0,$$

즉,

$$r^2 + r - 1 = 0$$

을 얻는다. 위 식의 근은

$$r_\pm = \frac{-1 \pm \sqrt{5}}{2}$$

이므로

$$\epsilon_n = Ar_+^n + Br_-^n$$

로 적을 수 있다. 여기서 $A$와 $B$는 상수인데, $\epsilon_0 = A + B$는 machine epsilon $\epsilon$ 정도의 값을 가지므로 $A \sim B \sim \epsilon \sim 10^{-16}$으로 놓을 수 있다. $|r_-| > 1$이므로 $n$이 커지면서 오차가 기하급수적으로 증가한다.

문제에서 $r_+ = \phi$이므로 $\phi^n$의 상대오차는

$$\frac{|\epsilon_n|}{\phi^n} = A + B\left|\frac{r_-}{r_+}\right|^n \sim \epsilon \left(\frac{\sqrt{5}+1}{\sqrt{5}-1}\right)^n \sim 10^{-16} \times 2.618^n$$

으로 주어진다. 따라서

$$n \simeq \frac{16}{\log_{10} 2.618} \simeq 38$$

에서 상대오차가 1을 초과한다.

## 3. Planck Function

$$\begin{aligned} B_\lambda &= \frac{2hc^2/\lambda^5}{e^{hc/\lambda kT} - 1} \\ \frac{dB_\lambda}{d\lambda} &= \frac{2hc^2}{\lambda^6 (e^{hc/\lambda kT} - 1)} \left[-5 + \frac{hc}{\lambda kT} \frac{e^{hc/\lambda kT}}{e^{hc/\lambda kT} - 1}\right] \end{aligned} \tag{3}$$

[9]:
```
h,k,c=6.626e-34,1.381e-23,2.998e8

def B(l,T):
    return (2.*h*pow(c,2.)*pow(l,-5.))/(np.exp(h*c/(l*k*T))-1.)
```

### (a) Wien's Displacement Law

$x = hc/\lambda kT$로 정의하면, $dB_\lambda/d\lambda = 0$는 다음과 같이 쓸 수 있다.

$$xe^x - 5e^x + 5 = 0 \tag{4}$$

[10]:
```
def wien(x):
    f=x*np.exp(x)-5.*np.exp(x)+5.
    df=x*np.exp(x)-4.*np.exp(x)
    return f,df

# Newton Method 사용

TOL=1.e-6
```

```
x=5.
err=1
loop=0
while(err>TOL):
    loop += 1
    f,df=wien(x)
    err=abs(f/df)
    x = x - f/df
    print ("N={:2d}  f={:.5e}  df={:.5e}  err={:.5e}".format(loop,f,df,err))
print (x)
```

```
N= 1  f=5.00000e+00  df=1.48413e+02  err=3.36897e-02
N= 2  f=1.65643e-01  df=1.38662e+02  err=1.19458e-03
N= 3  f=2.01202e-04  df=1.38325e+02  err=1.45456e-06
N= 4  f=2.97860e-10  df=1.38325e+02  err=2.15333e-12
4.965114231744277
```

[11]:
```
wien_b=h*c/(x*k)
print (wien_b)
```

```
0.0028977657184466
```

**(b)** $B_\lambda = 1.0 \times 10^{13}$ **at** $T = 10^4 K$

$$B_\lambda = \frac{2hc^2/\lambda^5}{e^{hc/\lambda kT} - 1} \tag{5}$$

(a)와 같이 $x$를 정의하면 위 식을 다음과 같이 정의할 수 있다.

$$\frac{2x^5}{e^x - 1} = B_\lambda \frac{h^4 c^3}{(kT)^5} \tag{6}$$

[12]:
```
T = 1.0e4
B_ref = 1.0e13
RHS = B_ref/(pow(k*T,5.)*pow(h,-4.)*pow(c,-3.))
print (RHS) # B_lambda=1.0e13일때 우변의 값!
```

```
1.0340278091358226
```

[13]:
```
def g(x):
    return 2.*pow(x,5.)-RHS*(np.exp(x)-1.)
def dg(x):
    return 10.*pow(x,4.)-RHS*np.exp(x)
```

[14]:
```
# Newton Method 사용

x1 = 3.
TOL, err, loop = 1.e-5, 1,0
```

```python
while(err > TOL):
    loop += 1
    g1, dg1 = g(x1), dg(x1)
    err = abs(g1/dg1)
    x1 = x1 - g1/dg1
    print ("N={:2d}  f={:.5e}  df={:.5e}  err={:.5e}".format(loop,g1,dg1,err))
w1=(h*c)/(x1*k*T)
print ("wavelength 1 : {:.4e} m, B_1 : {:.4e} J s-1 m-3".format(w1,B(w1,T)))
```

```
N= 1  f=4.66265e+02  df=7.89231e+02  err=5.90784e-01
N= 2  f=1.51864e+02  df=3.25398e+02  err=4.66703e-01
N= 3  f=4.91363e+01  df=1.35169e+02  err=3.63518e-01
N= 4  f=1.56495e+01  df=5.71466e+01  err=2.73849e-01
N= 5  f=4.79431e+00  df=2.52022e+01  err=1.90234e-01
N= 6  f=1.32632e+00  df=1.22982e+01  err=1.07847e-01
N= 7  f=2.74981e-01  df=7.45493e+00  err=3.68858e-02
N= 8  f=2.48905e-02  df=6.13130e+00  err=4.05958e-03
N= 9  f=2.77280e-04  df=5.99500e+00  err=4.62520e-05
N=10  f=3.56738e-08  df=5.99345e+00  err=5.95213e-09
wavelength 1 : 1.4889e-06 m, B_1 : 1.0000e+13 J s-1 m-3
```

[15]:
```python
x2 = 15.
TOL, err, loop = 1.e-5, 1, 0
while(err > TOL):
    loop += 1
    g2, dg2 = g(x2), dg(x2)
    err = abs(g2/dg2)
    x2 = x2 - g2/dg2
    print ("N={:2d}  f={:.5e}  df={:.5e}  err={:.5e}".format(loop,g2,dg2,err))
w2=(h*c)/(x2*k*T)
print ("wavelength 2 : {:.4e} m, B_2 : {:.4e} J s-1 m-3".format(w2,B(w2,T)))
```

```
N= 1  f=-1.86150e+06  df=-2.87400e+06  err=6.47704e-01
N= 2  f=-5.50732e+05  df=-1.34439e+06  err=4.09652e-01
N= 3  f=-1.20415e+05  df=-7.96307e+05  err=1.51217e-01
N= 4  f=-1.15470e+04  df=-6.47649e+05  err=1.78291e-02
N= 5  f=-1.42831e+02  df=-6.31679e+05  err=2.26114e-04
N= 6  f=-2.26748e-02  df=-6.31478e+05  err=3.59074e-08
wavelength 2 : 1.0444e-07 m, B_2 : 1.0000e+13 J s-1 m-3
```

## 4. Kepler's Equation

[16]:
```python
def KeplerEq(y,x,eps):
    f = y-eps*np.sin(y)-x # 해를 구하고자 하는 방정식 (Kepler's Equation)
    df = 1.-eps*np.cos(y) # Kepler's Equation을 y에 대해서 미분
    return f, df
```

```python
e = 0.9
xlist = np.linspace(0.,np.pi,30) # 0<= x <= pi 구간에 대해 30등분
ylist = np.array([]) # 방정식의 해를 작성할 Array

print (xlist)
```

```
[0.         0.10833078 0.21666156 0.32499234 0.43332312 0.54165391
 0.64998469 0.75831547 0.86664625 0.97497703 1.08330781 1.19163859
 1.29996937 1.40830016 1.51663094 1.62496172 1.7332925  1.84162328
 1.94995406 2.05828484 2.16661562 2.2749464  2.38327719 2.49160797
 2.59993875 2.70826953 2.81660031 2.92493109 3.03326187 3.14159265]
```

[17]:
```python
TOL = 1.0e-10
y_guess = 1. # y를 구하는 데 사용할 예측값

for x_i in xlist:
    y_i = y_guess # 이전 Step에서 구한 y값을 초기값으로 사용
    err = 1.
    loop = 0
    while (err > TOL):
        loop += 1
        f, df = KeplerEq(y_i,x_i,e)
        err = abs(f/df)
        y_i = y_i - f/df
        progress = [loop, y_i, f, err]
    y_guess = y_i # 구한 해를 다음 step에서 초기값으로 사용
    ylist = np.append(ylist,y_i) # 구한 해를 ylist에 넣음
```

[18]:
```python
table = np.column_stack([xlist,ylist]) # x를 1열, y를 2열에 배치함.
print (np.array_str(table,precision=5,suppress_small=True))
```

```
[[0.      0.     ]
 [0.10833 0.66047]
 [0.21666 0.94734]
 [0.32499 1.14441]
 [0.43332 1.30069]
 [0.54165 1.43314]
 [0.64998 1.54979]
 [0.75832 1.65512]
 [0.86665 1.75192]
 [0.97498 1.84207]
 [1.08331 1.92686]
 [1.19164 2.00726]
 [1.29997 2.08402]
 [1.4083  2.1577 ]
 [1.51663 2.22875]
```

```
[1.62496 2.29756]
[1.73329 2.36443]
[1.84162 2.42962]
[1.94995 2.49336]
[2.05828 2.55583]
[2.16662 2.61722]
[2.27495 2.67766]
[2.38328 2.7373 ]
[2.49161 2.79626]
[2.59994 2.85465]
[2.70827 2.91258]
[2.8166  2.97015]
[2.92493 3.02744]
[3.03326 3.08456]
[3.14159 3.14159]]
```