전산천문학 HW6 Solution

June 24, 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import linalg
from scipy.odr import *

import warnings
warnings.filterwarnings('ignore')
```

1.

[2]:
```
lam=np.array([[0.      ,88.356E4,32.661E4,4931.1,0.    ],
              [88.356E4,0.      ,51.814E4,4958.9,2321.0],
              [32.661E4,51.814E4,0.      ,5006.9,2331.4],
              [4931.1  ,4958.9  ,5006.9  ,0.    ,4363.2],
              [0.      ,2321.0  ,2331.4  ,4363.2,0.    ]]).T

Eng=np.array([[0.   ,0.014,0.038,2.514,5.300],
              [0.014,0.   ,0.024,2.500,5.314],
              [0.038,0.024,0.   ,2.476,5.318],
              [2.514,2.500,2.476,0.   ,2.842],
              [5.300,5.314,5.318,2.842,0.   ]]).T

A=np.array([[0.     ,2.7E-05,3.1E-11,1.7E-06,0.     ],
            [0.     ,0.     ,9.7E-05,6.8E-03,2.3E-01],
            [0.     ,0.     ,0.     ,2.0E-02,6.1E-04],
            [0.     ,0.     ,0.     ,0.     ,1.6    ],
            [0.     ,0.     ,0.     ,0.     ,0.     ]]).T

Om1=np.array([[0.   ,0.55,0.27,0.254,0.032],
              [0.   ,0.  ,1.29,0.763,0.097],
              [0.   ,0.  ,0.  ,1.272,0.161],
              [0.   ,0.  ,0.  ,0.   ,0.58 ],
              [0.   ,0.  ,0.  ,0.   ,0.   ]]).T

b0=np.array([0.,0.,0.,0.,1.])
omega=np.array([1.,3.,5.,5.,1.])
```

(a)

[3]:
```
def O2plus_pop(Ne,T): # Ne:cm-3, T:K
    Ne,T=np.atleast_1d(Ne),np.atleast_1d(T)
    N1,N2=len(Ne),len(T)
    pop=np.zeros([N1,N2,5])
    for jt in range(N1):
        ne=Ne[jt]
        for it in range(N2):
            t=T[it]/1e4
            q=np.zeros([5,5])
            M=np.zeros([5,5])
            for i in range(4):
                for j in range(i,5): #i<j
                    q[j,i]=8.629e-8*Om1[j,i]/(np.sqrt(t)*omega[j])

            for i in range(1,5):
                for j in range(i): #i>j
                    coef2 = np.exp(-1.1605*Eng[i,j]/t)
                    q[j,i]=q[i,j]*omega[i]/omega[j]*coef2

            for i in range(4):
                qsum,asum=0,0
                for k in range(i): # k<i
                    M[i,k]=q[k,i]*ne
                    qsum += q[i,k]
                    asum += A[i,k]

                for k in range(i+1,5): #k>i
                    M[i,k]=q[k,i]*ne + A[k,i]
                    qsum += q[i,k]
                k=i
                M[i,k]=-(qsum*ne + asum)

            i=4
            for k in range(5):
                M[i,k]=1. # 전체의 합이 1이 되도록 함
            LU=linalg.lu_factor(M)     # calculate pivoted LU decomposition
            x =linalg.lu_solve(LU, b0) # solve Ax=b for given LU factorization
            pop[jt,it,:]=x

    return pop #pop[i,j,k]=i번째 Ne, j번째 T에서의 n_k
```

[4]:
```
Ne1=[1.0e2,1.0e2,1.0e1]
T1=[1.0e4,2.0e4,1.0e4]

for i in range(3):
    print ('Ne : {:.1e} cm-3, T : {:.1e} K'.format(Ne1[i],T1[i]))
```

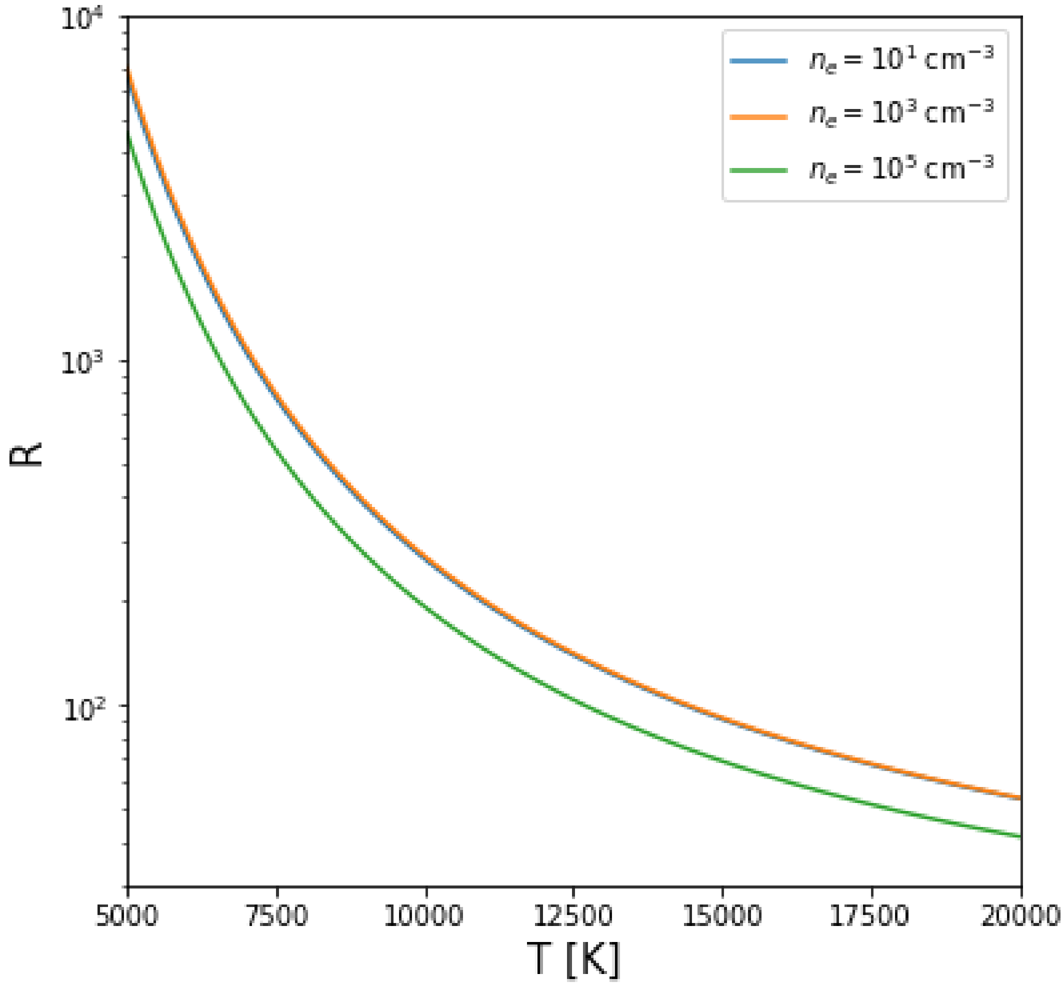
```
    pop_i = O2plus_pop(Ne1[i],T1[i])[0,0]
    print ('=> (n1,n2,n3,n4,n5) = ({:.4e}, {:.4e}, {:.4e}, {:.4e}, {:.4e})'.
↪format(pop_i[0],pop_i[1],pop_i[2],pop_i[3],pop_i[4]))
```

```
Ne : 1.0e+02 cm-3, T : 1.0e+04 K
=> (n1,n2,n3,n4,n5) = (7.8287e-01, 1.9181e-01, 2.5313e-02, 4.4603e-06,
3.2121e-10)
Ne : 1.0e+02 cm-3, T : 2.0e+04 K
=> (n1,n2,n3,n4,n5) = (8.2436e-01, 1.5543e-01, 2.0204e-02, 1.3765e-05,
4.9272e-09)
Ne : 1.0e+01 cm-3, T : 1.0e+04 K
=> (n1,n2,n3,n4,n5) = (9.7246e-01, 2.5125e-02, 2.4131e-03, 4.4432e-07,
3.2155e-11)
```

(b)

[5]:
```
def O2plus_R(Ne,T):
    Ne,T=np.atleast_1d(Ne),np.atleast_1d(T)
    N1,N2=len(Ne),len(T)
    Ratio=np.zeros([N1,N2])
    pop=O2plus_pop(Ne,T)
    for jt in range(N1):
        for it in range(N2):
            x=pop[jt,it]
            Ratio[jt,it]=(x[3]/x[4])*(A[3,2]*lam[3,2]+A[3,1]*lam[3,1])/
↪(A[4,3]*lam[4,3])
    return Ratio # Ratio[i,j]=i번째 Ne, j번째 T에서의 R
```

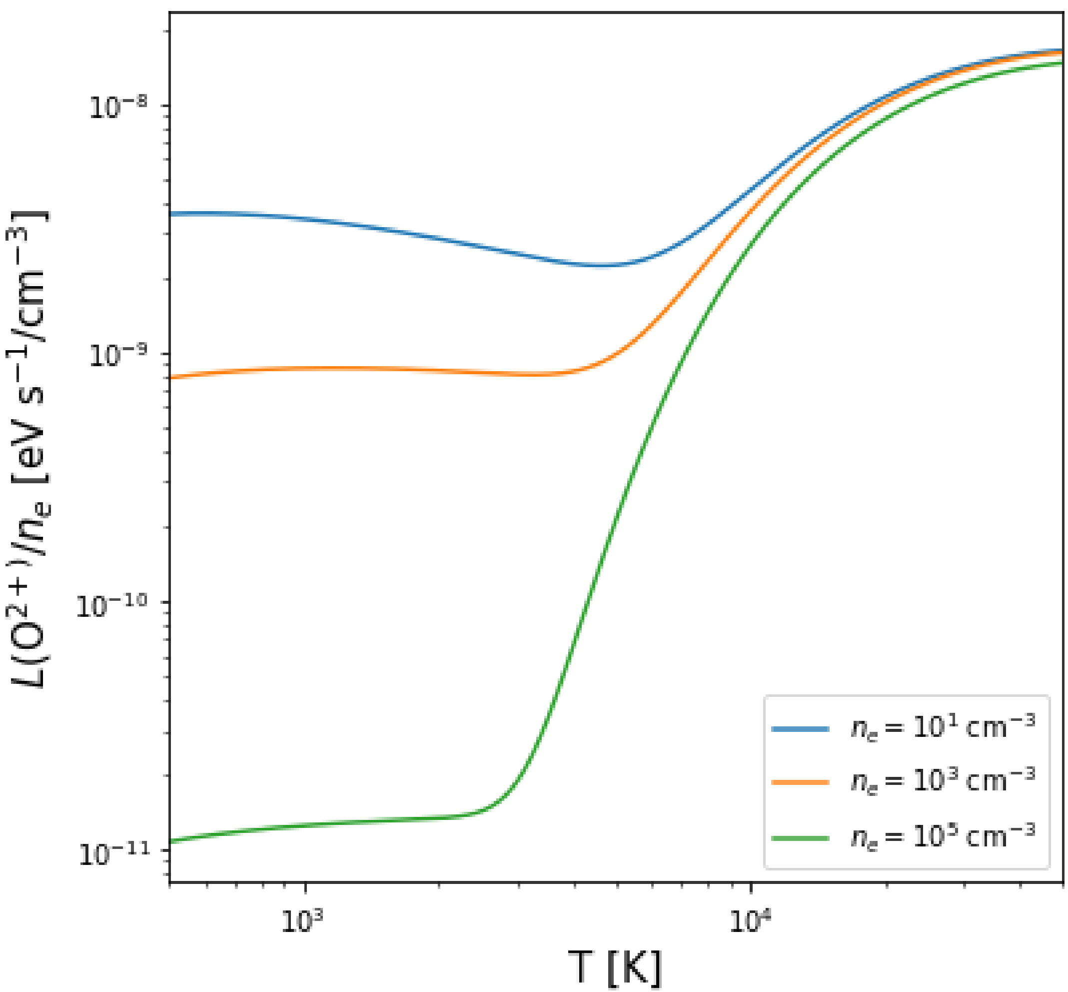
[6]:
```
Ne,Temp=[1e1,1e3,1e5],np.linspace(5e3,2e4,400)
Ratio=O2plus_R(Ne,Temp)

plt.figure(figsize=(6,6))
plt.semilogy(Temp,Ratio[0,:],label=r'$n_e=10^1$ $\mathrm{cm}^{-3}$')
plt.semilogy(Temp,Ratio[1,:],label=r'$n_e=10^3$ $\mathrm{cm}^{-3}$')
plt.semilogy(Temp,Ratio[2,:],label=r'$n_e=10^5$ $\mathrm{cm}^{-3}$')
plt.xlabel(r'T [K]',fontsize=15)
plt.ylabel('R',fontsize=15)
plt.xlim(5000.,20000.)
plt.xticks(np.arange(5000.,20001.,2500.))
plt.ylim(3.0e1,1.0e4)
plt.legend()
plt.show()
```



(c)

[7]:
```
def O2plus_L(Ne,T):
    Ne,T=np.atleast_1d(Ne),np.atleast_1d(T)
    N1,N2=len(Ne),len(T)
    Loss=np.zeros([N1,N2])
    pop=O2plus_pop(Ne,T)
    for jt in range(N1):
        for it in range(N2):
            x=pop[jt,it]
            for i in range(1,5):
                for k in range(0,i):
                    Loss[jt,it]+=x[i]*A[i,k]*Eng[k,i]
    return Loss # Loss[i,j]=i번째 Ne, j번째 T에서의 L(O2+)
```

[8]:
```
Ne,Temp=[1e1,1e3,1e5],np.logspace(2+np.log10(5.),4.+np.log10(5.),1000)
Loss=O2plus_L(Ne,Temp)

plt.figure(figsize=(6,6))
plt.loglog(Temp,Loss[0,:]/Ne[0],label=r'$n_e=10^1$ $\mathrm{cm}^{-3}$')
plt.loglog(Temp,Loss[1,:]/Ne[1],label=r'$n_e=10^3$ $\mathrm{cm}^{-3}$')
plt.loglog(Temp,Loss[2,:]/Ne[2],label=r'$n_e=10^5$ $\mathrm{cm}^{-3}$')
plt.xlabel(r'T [K]',fontsize=15)
plt.ylabel(r'$L$(O$^{2+})$/$n_e$ [eV s$^{-1}$/cm$^{-3}$]',fontsize=15)
plt.xlim(500.,50000.)
# plt.ylim(3.0e1,1.0e4)
plt.legend(loc=4)
plt.show()
```



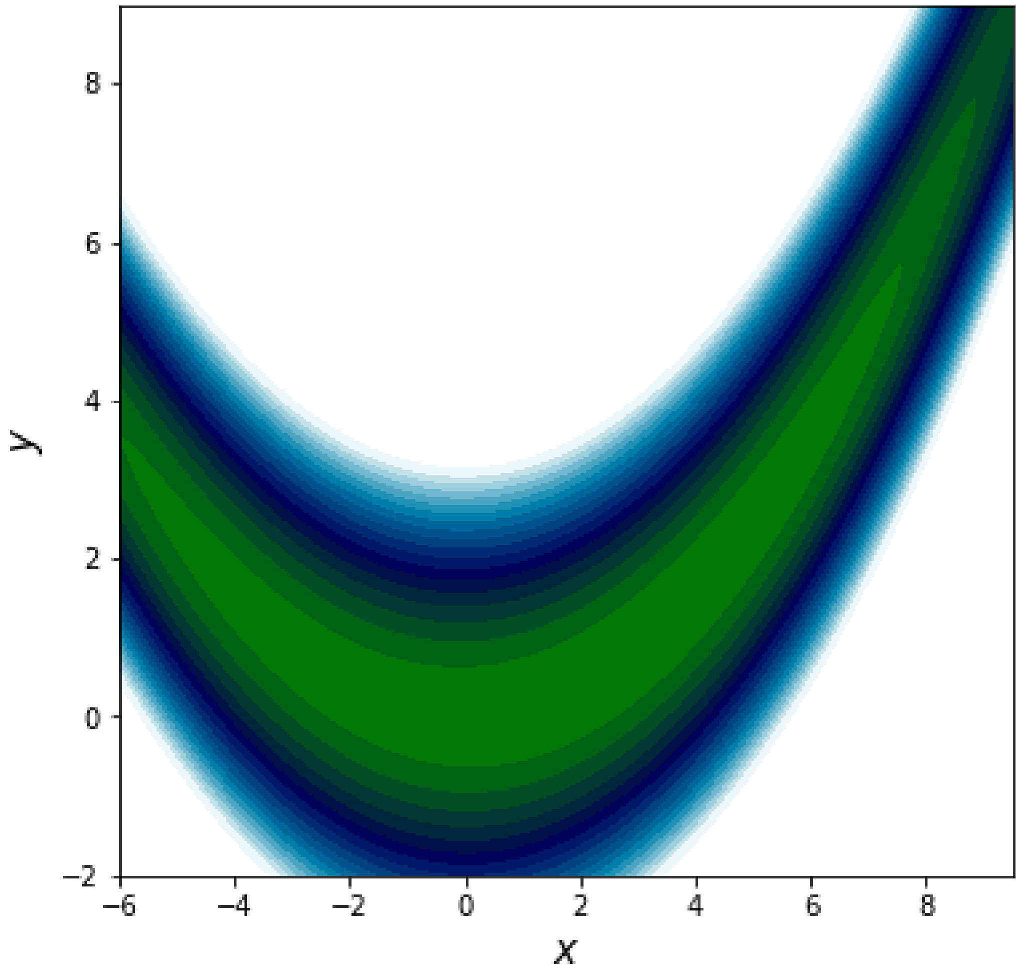
2.

[9]:
```
def func(x):
    return (((x[0]**2.)-10.*x[1])**2.)+2.*(x[1]**2.)-3.*x[0]

def deriv(x):
    f1 = 4.*x[0]*((x[0]**2.)-10.*x[1])-3.
    f2 = -20.*((x[0]**2.)-10.*x[1])+4.*x[1]
    return np.array([f1,f2])
```

[10]:
```
def golden2D(x,dr,tol):
    R,err,loop=0.61803399,10.,-1
    a,b=x-dr,x+dr # initialize the range
    while(err > TOL):
        loop += 1
        x1,x2=b-R*(b-a),a+R*(b-a)
        f1,f2=func(x1),func(x2)
        if (f2>f1):
            b=x2
        else:
            a=x1
        err=np.sqrt(sum((a-b)**2))
    xmin=a
    return xmin,err,loop
```

[11]:
```
x_gd,y_gd=np.linspace(-6.5,9.5,500),np.linspace(-2.5,9.,500)
X, Y =np.meshgrid(x_gd,y_gd)
E=(((X**2.)-10.*Y)**2.)+2.*(Y**2.)-3.*X
dmax,dmin=1000.,-10.
levels=(dmax-dmin)*np.arange(20)/19.+dmin
cmap=plt.cm.ocean

plt.figure(figsize=(6,6))
plt.contourf(x_gd, y_gd, E, levels,cmap=cmap)
plt.xlim(-6.,9.5)
plt.ylim(-2.,9.)
plt.xlabel(r'$x$',fontsize=15)
plt.ylabel(r'$y$',fontsize=15)
```

[11]:
```
Text(0, 0.5, '$y$')
```



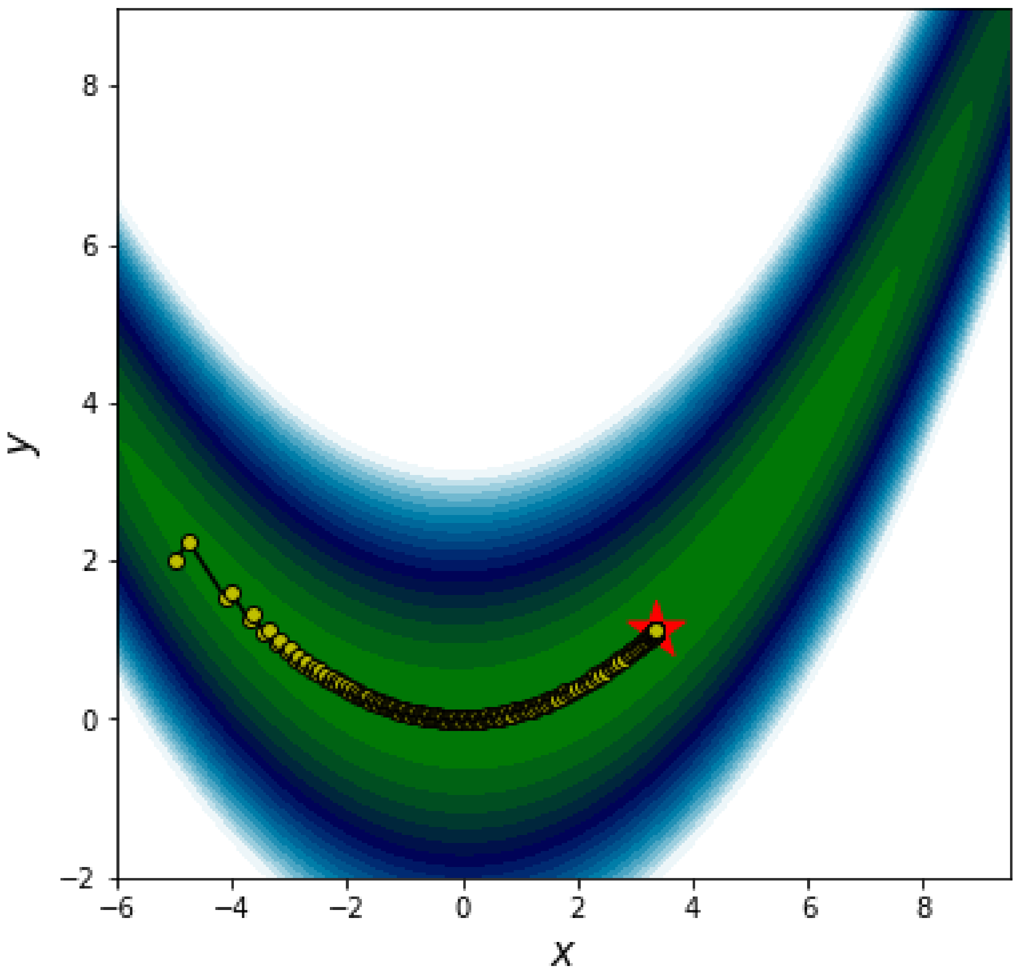
(a) The Steepest Descent Method

[12]:
```
err, TOL = 1.0, 1.0e-6
x=np.array([-5.,2.])
xt,yt=np.array([x[0]]),np.array([x[1]])

loop=0
while err>TOL:
    loop+=1
    dr = -deriv(x)/np.sqrt(np.sum(np.power(deriv(x),2.)))
    xmin,err_lp,loop_lp = golden2D(x,dr,1.0e-5)
    err = np.sqrt(sum((xmin-x)**2.))
    x = xmin
    xt,yt=np.append(xt,x[0]),np.append(yt,x[1])
print (x,loop)
```

```
[3.36926465 1.11293537] 220
```

[13]:
```
plt.figure(figsize=(6,6))
plt.plot(xt,yt,marker='o',c='k',markerfacecolor='y')
plt.contourf(x_gd, y_gd, E, levels,cmap=cmap)
plt.xlim(-6.,9.5)
plt.ylim(-2.,9.)
plt.xlabel(r'$x$',fontsize=15)
plt.ylabel(r'$y$',fontsize=15)
plt.scatter(x[0],x[1],c='r',marker='*',s=500)
```

[13]:
```
<matplotlib.collections.PathCollection at 0x1c357329fd0>
```



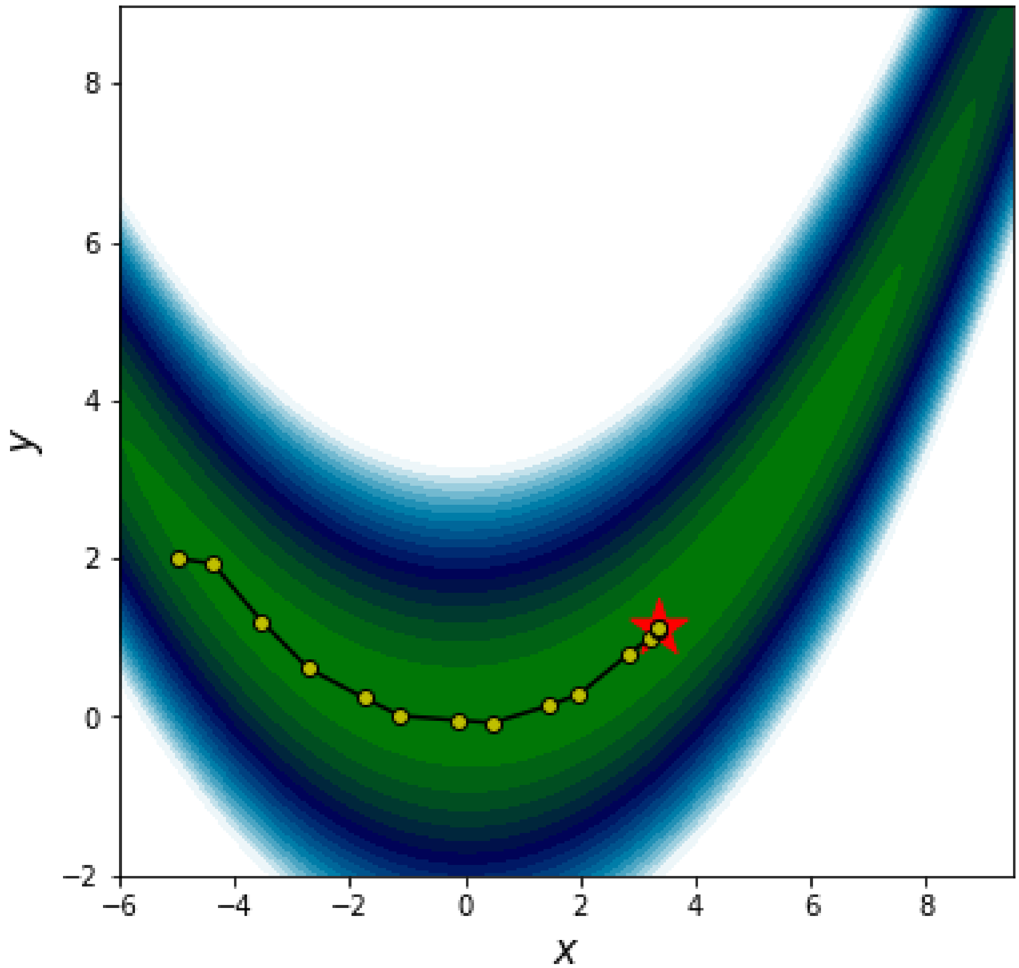
(b) Powell's Method

[14]:
```
P0=np.array([-5.,2.])
xt,yt=np.array([P0[0]]),np.array([P0[1]])

u=np.array([[1.,0.],[0.,1.]]) # Cartesian unit vectors
ut = u

loop=0
err, TOL = 1.0, 1.0e-6
while err>TOL:
    loop+=1
    u1,u2=u[0],u[1] # Initialize set of directions

    #Step 2
    P1,err1,loop1 = golden2D(P0,u1,1.0e-6)
    P2,err2,loop2 = golden2D(P1,u2,1.0e-6)

    #Step 3
    u1 = u2
    u2 = (P2-P0)/np.sqrt(sum((P2-P0)**2.)) # Unit vector로 변환
    u  = np.vstack((u1,u2))
    ut = np.vstack((ut,u2))

    #Step 4
    newP0,err_lp,loop_lp = golden2D(P2,u2,1.0e-6)
    err = np.sqrt(sum((newP0-P0)**2.))
    P0 = newP0
    xt,yt=np.append(xt,P0[0]),np.append(yt,P0[1])
print (P0,loop)
```

```
[3.36933163 1.11298012] 14
```

[15]:
```
plt.figure(figsize=(6,6))
plt.plot(xt,yt,marker='o',c='k',markerfacecolor='y')
plt.contourf(x_gd, y_gd, E, levels,cmap=cmap)
plt.xlim(-6.,9.5)
plt.ylim(-2.,9.)
plt.xlabel(r'$x$',fontsize=15)
plt.ylabel(r'$y$',fontsize=15)
plt.scatter(x[0],x[1],c='r',marker='*',s=500)
```

[15]:
```
<matplotlib.collections.PathCollection at 0x1c359506be0>
```



3.

1)

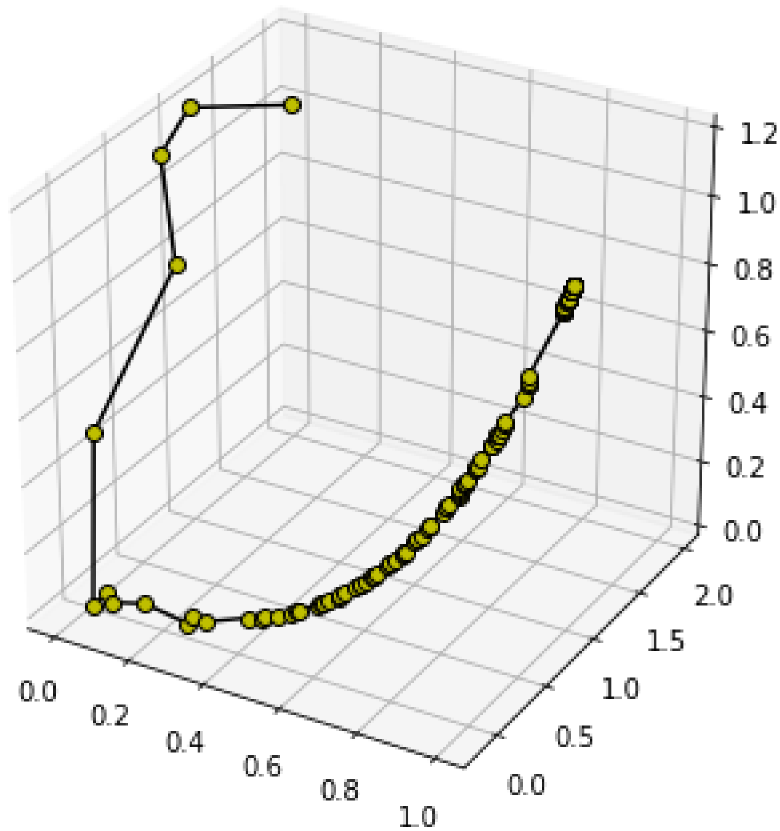
[16]:
```
def func(x):
    return 100*(x[1]-x[0]**2)**2 + (1-x[0])**2+100*(x[2]-x[1]**2)**2 +␣
 ↪(1-x[2])**2

def deriv(x):
    f1=-400*x[0]*(x[1]-x[0]**2)-2*(1-x[0])
    f2=200*(x[1]-x[0]**2)-400*x[1]*(x[2]-x[1]**2)
    f3=200*(x[2]-x[1]**2)-2*(1-x[2])
    return np.array([f1,f2,f3])

err,TOL=1.,1.e-6
x=np.array([0.,2.,1.])
```

```python
xt,yt,zt=[x[0]],[x[1]],[x[2]]

loop=0
while err>TOL:
    loop+=1
    if loop==1:
        g_now=-deriv(x)
        dr_now=g_now/np.sqrt(sum(g_now**2))
        xmin,err_lp,loop_lp=golden2D(x,dr_now,1.e-5)
        err=np.sqrt(sum((xmin-x)**2))
        x=xmin
        xt,yt,zt=np.append(xt,x[0]),np.append(yt,x[1]),np.append(zt,x[2])
        g_prev=g_now # 이전 단계의 gradient 저장
        dr_prev=dr_now # 이전 단계의 direction 저장
    else:
        g_now=-deriv(x)
        l=sum(g_now**2)/sum(g_prev**2) # Fletcher-Reeves Method
#         l=sum(g_now*(g_now-g_prev))/sum(g_prev**2) # Polak-Ribière Method
        dr_now=g_now+l*dr_prev
        dr_now=dr_now/np.sqrt(sum(dr_now**2)) # dr을 단위벡터로 변환
        xmin,err_lp,loop_lp=golden2D(x,dr_now,1.e-6)
        err=np.sqrt(sum((x-xmin)**2))
        x=xmin
        xt,yt,zt=np.append(xt,x[0]),np.append(yt,x[1]),np.append(zt,x[2])
        g_prev=g_now
        dr_prev=dr_now
print (xmin,loop)
```

```
[0.99991815 0.99989568 0.9998397 ] 154
```

[17]:
```python
from mpl_toolkits import mplot3d

fig = plt.figure(figsize=(6,6))
ax = plt.axes(projection='3d')
ax.plot3D(xt,yt,zt,c='k',marker='o',markerfacecolor='y')
```

[17]:
```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x1c358f3c6a0>]
```



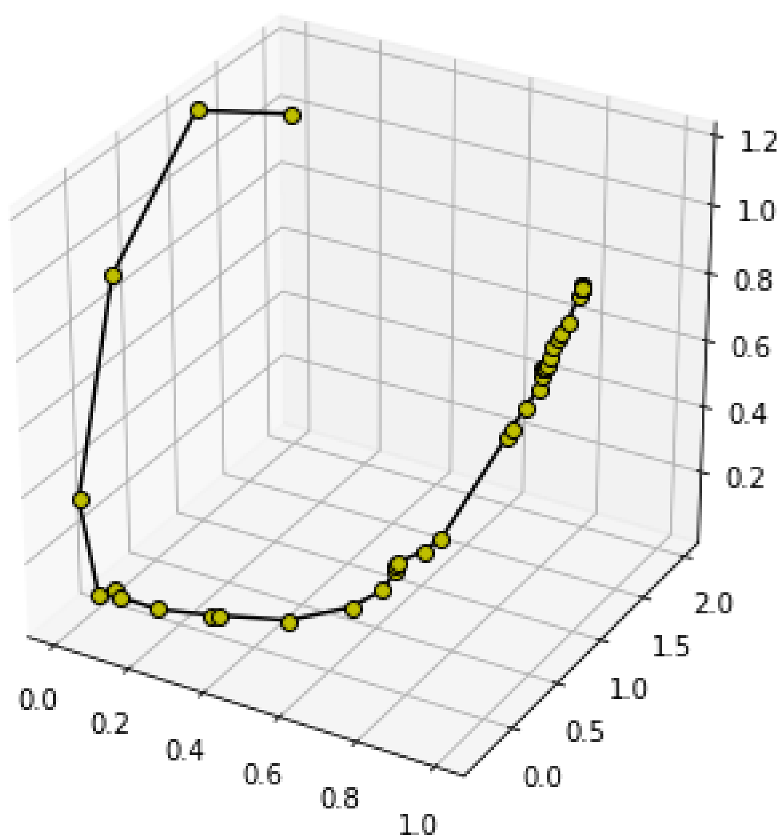
2) scipy 사용

[18]:
```
from scipy.optimize import minimize
x0=np.array([0.,2.,1.])
xt,yt,zt=[x0[0]],[x0[1]],[x0[2]]

def rosen(x):
    return 100*(x[1]-x[0]**2)**2+(1-x[0])**2+100*(x[2]-x[1]**2)**2+(1-x[2])**2
def callbackF(x):
    global xt,yt,zt
    xt.append(x[0])
    yt.append(x[1])
    zt.append(x[2])

res = minimize(rosen, x0, method='CG',\
               options={'gtol': 1e-5, 'disp': True},callback=callbackF)
print (res.x)
```

```
Warning: Desired error not necessarily achieved due to precision loss.
         Current function value: 0.000000
         Iterations: 39
         Function evaluations: 707
         Gradient evaluations: 139
[0.99999935 0.99999874 0.99999744]
```

[19]:
```
from mpl_toolkits import mplot3d

fig = plt.figure(figsize=(6,6))
ax = plt.axes(projection='3d')
ax.plot3D(xt,yt,zt,c='k',marker='o',markerfacecolor='y')
```

[19]:
```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x1c358f98da0>]
```



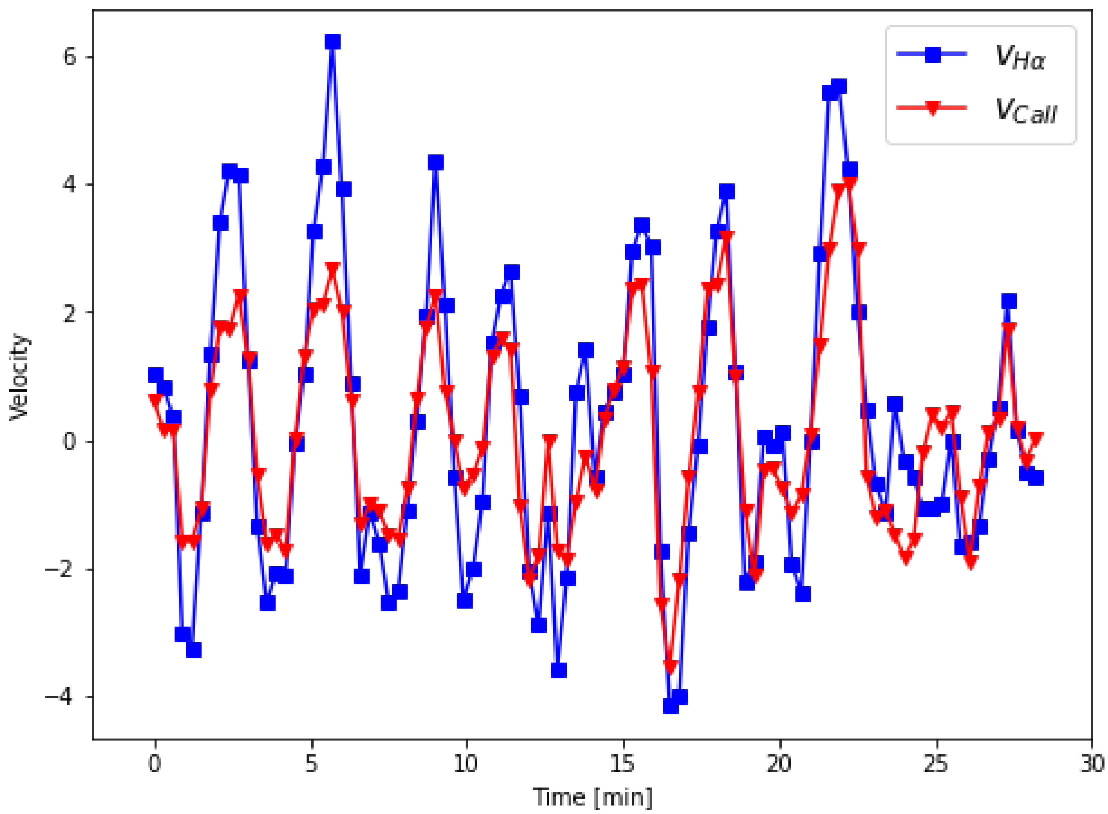
4.

[20]:
```
t,v_ha,v_caii=np.loadtxt('sol_vel.dat',unpack=True,usecols=(0,1,2))
```

(a)

[21]:
```
plt.figure(figsize=(8,6))

plt.plot(t,v_ha,c='blue',marker='s',label=r'$v_{H\alpha}$')
plt.plot(t,v_caii,c='red',marker='v',label=r'$v_{CaII}$')
plt.xlabel('Time [min]')
plt.ylabel('Velocity')
plt.xlim(-2.,30.)
plt.legend(fontsize=15)
```

[21]:
```
<matplotlib.legend.Legend at 0x1c34beff550>
```

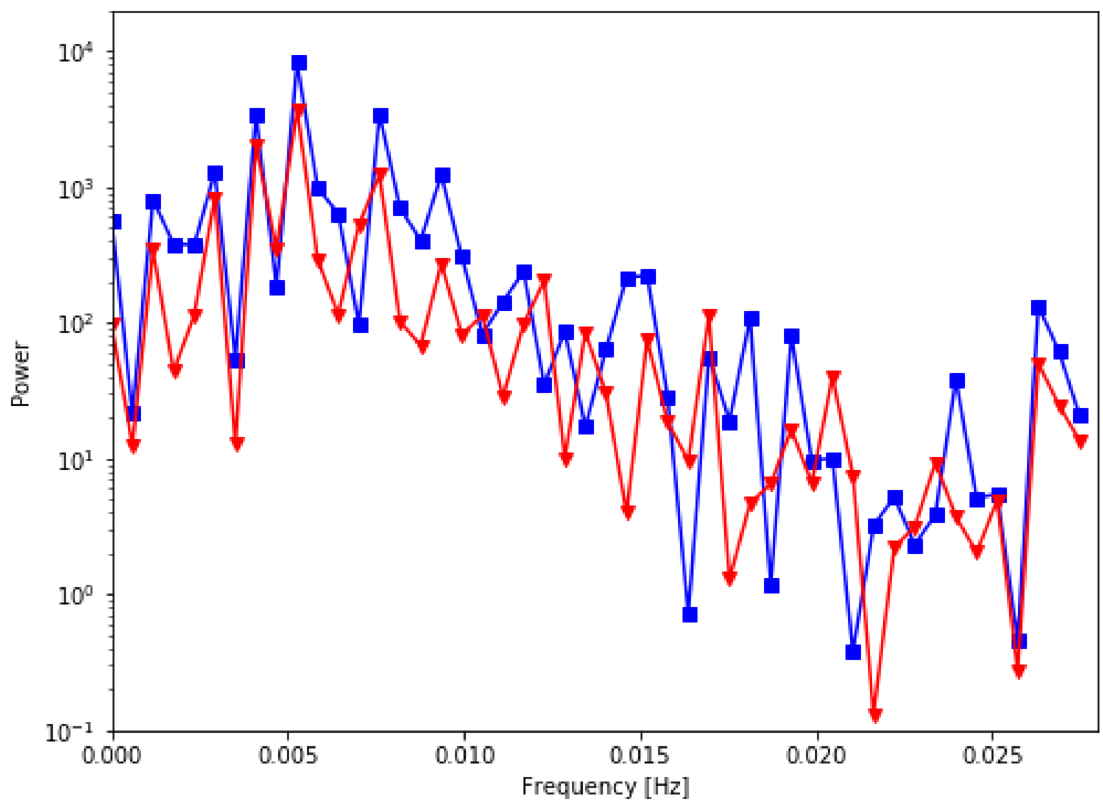


(b)

[22]:
```
# H alpha
ha_fft=np.fft.fft(v_ha)
ha_fft=np.fft.fftshift(ha_fft)

# Ca II
caii_fft=np.fft.fft(v_caii)
```

```
caii_fft=np.fft.fftshift(caii_fft)

# Sample Frequency
t_fft=np.fft.fftfreq(len(t),18) # 0.3분 간격으로 측정된 값!
t_fft=np.fft.fftshift(t_fft)

plt.figure(figsize=(8,6))
plt.semilogy(t_fft,np.absolute(ha_fft)**2,c='blue',marker='s',label=r'H$\alpha$')
plt.semilogy(t_fft,np.absolute(caii_fft)**2,c='red',marker='v',label=r'Ca II')
plt.xlim(0.0,0.028)
plt.ylim(1.0e-1,2.0e4)
plt.xlabel('Frequency [Hz]')
plt.ylabel('Power')
```

[22]:
```
Text(0, 0.5, 'Power')
```



(c)

[23]:
```
t_fft_pos=t_fft[t_fft>0.] # frequency>0만 뽑아냄!

# Power Spectrum에서 최대값을 가지는 Frequency를 주기로 변환
```

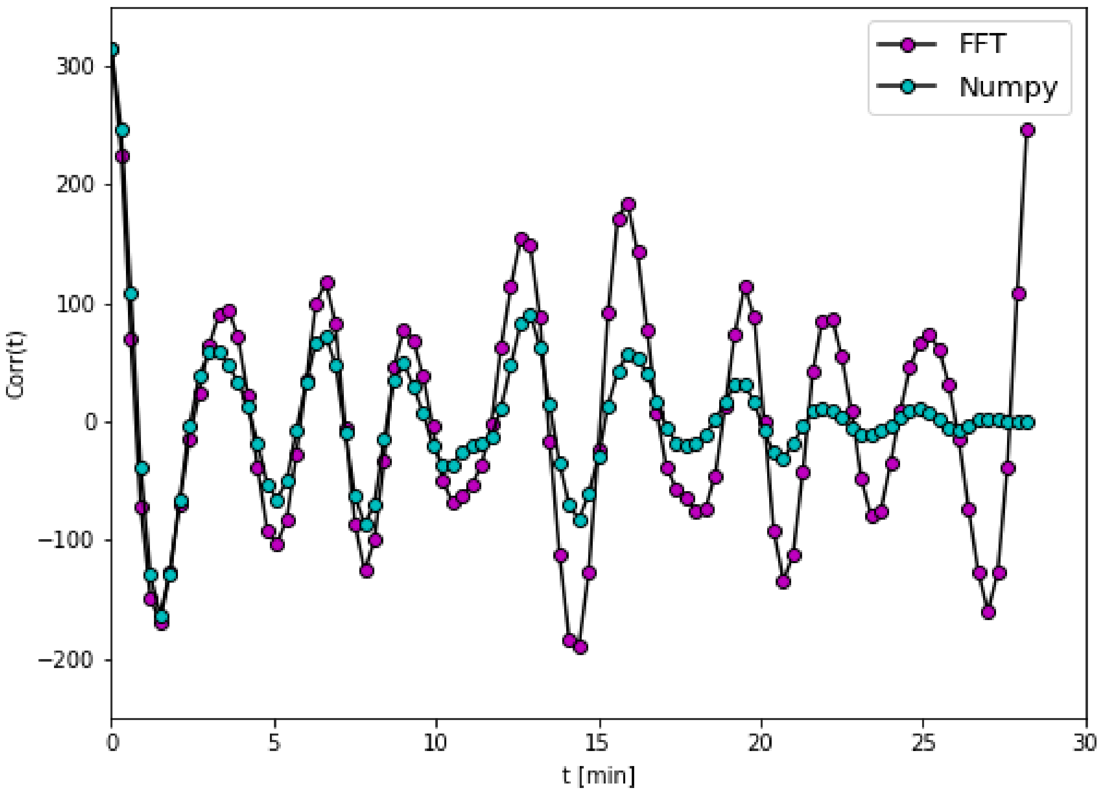
```
P_ha   =1./(t_fft_pos[np.argmax(np.absolute(ha_fft[t_fft>0.])**2)])
P_caii =1./(t_fft_pos[np.argmax(np.absolute(caii_fft[t_fft>0.])**2)])

print ('H alpha : {:.2e} sec, Ca II : {:.2e} sec'.format(P_ha,P_caii))
```

```
H alpha : 1.90e+02 sec, Ca II : 1.90e+02 sec
```

(d)

[24]:
```
# 1. FFT를 이용
ha_fft=np.fft.fft(v_ha)
caii_fft=np.fft.fft(v_caii)
Corr=np.fft.ifft(np.conj(ha_fft)*caii_fft) # Correlation Theorem!

# 2. np.correlate 내장함수 사용
Corr2=np.correlate(v_ha,v_caii,'full')
lag=np.arange(-t[-1],t[-1]+0.01,0.3)

plt.figure(figsize=(8,6))
plt.plot(t,Corr,c='k',marker='o',markerfacecolor='m',label='FFT')
plt.plot(lag,Corr2,c='k',marker='o',markerfacecolor='c',label='Numpy')
plt.xlim(0.,30.)
plt.ylim(-250.,350.)
plt.xlabel('t [min]')
plt.ylabel('Corr(t)')
plt.legend(fontsize=13)
```

[24]:
```
<matplotlib.legend.Legend at 0x1c3593ac8d0>
```



데이터의 범위가 $(-\infty, \infty)$가 아니기에 두 결과가 다르게 나타난다. 하지만 그 경향성은 일치하는 것을 볼 수 있다.