전산천문학 HW4 Solution

May 13, 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import gamma
from scipy.integrate import odeint

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

1.

[2]:
```
def RanWalk2d(nstep):
    p = np.random.rand(nstep)
    x_p, x_m = np.where(p<1./6.), np.where(abs(p-1./3.)<1./6.)
    y_p, y_m = np.where(abs(p-5./8.)<1./8.), np.where(abs(p-7./8.)<1./8.)
    return (len(p[x_p])-len(p[x_m]),len(p[y_p])-len(p[y_m]))
```

(a)

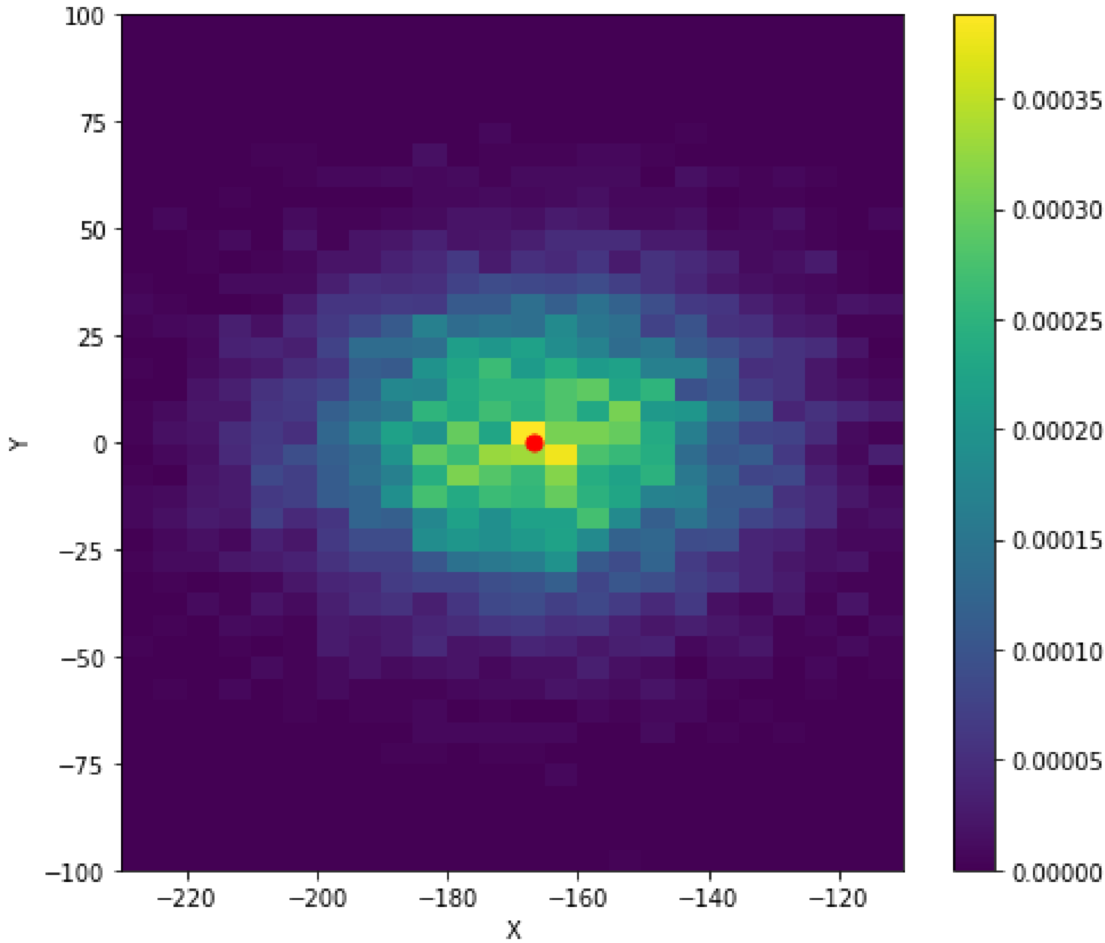
[3]:
```
nsample = 10000
ntrial = 10000

prob_list=np.zeros(nsample)
for i in range(nsample):
    dlist=np.zeros(ntrial)
    for n in range(ntrial):
        d_n = np.sqrt(np.sum(np.square(RanWalk2d(50))))
        dlist[n]=d_n
    prob_i = len(dlist[np.where(dlist>20.)])/ntrial
    prob_list[i] = prob_i

print ('The probability is %.4f±%.4e' % (np.mean(prob_list),np.std(prob_list)))
```

```
The probability is 0.0124±1.1172e-03
```

(b)

[4]:
```
ntrial = 10000
bins = np.arange(-250.,250.,5.)

plt.figure(figsize=(7,6))
xlist,ylist=[],[]
for n in range(ntrial):
    x_n,y_n = RanWalk2d(1000)
    xlist.append(x_n)
    ylist.append(y_n)
xlist,ylist = np.array(xlist),np.array(ylist)

plt.hist2d(xlist,ylist,bins,normed=True)
plt.xlim(-230.,-110.)
plt.ylim(-100.,100.)
plt.xlabel('X')
plt.ylabel('Y')
plt.colorbar()

print ('The most probable position is (x,y) = (%.2f±%.2f,%.2f±%.2f)' % (np.
 ↪mean(xlist),np.std(xlist),np.mean(ylist),np.std(ylist)))
plt.scatter(np.mean(xlist),np.mean(ylist),c='r',marker='o',s=50)

plt.tight_layout()
```

```
The most probable position is (x,y) = (-166.63±21.56,-0.10±22.65)
```



2.

[5]:
```
npeople_list = [10,30,50]
ntrial = 100000

for npeople in npeople_list:
    bday_overlap = 0
    for n in range(ntrial):
        bday_list=np.random.randint(1,366,size=npeople)
        bday_unique = np.unique(bday_list)
        if len(bday_list)-len(bday_unique) > 0. :
            bday_overlap+=1
    print ('The Probablity for %d people : %.3f' % (npeople,bday_overlap/ntrial))
```

```
The Probablity for 10 people : 0.116
The Probablity for 30 people : 0.706
The Probablity for 50 people : 0.970
```

[6]:
```python
npeople = 1
ntrial = 100000
prob=0.

while prob < 0.5:
    npeople+=1
    bday_overlap = 0
    for n in range(ntrial):
        bday_list=np.random.randint(1,366,size=npeople)
        bday_unique = np.unique(bday_list)
        if len(bday_list)-len(bday_unique) > 0. :
            bday_overlap+=1
    prob = bday_overlap / ntrial

print ("The Probability reaches half when n = %d" % npeople)
```

```
The Probability reaches half when n = 23
```

3.

(a)

[7]:
```python
def photon_prob(Np,tau_max):
    z_max=1.
    N=1
    count_ref,count_trans=0,0
    while N<=Np:
        chi_i=np.random.rand(2) # Step 1에 사용할 랜덤 변수

        # Step 1
        pos_i=np.zeros(3)
        theta_i,phi_i=np.arccos(np.sqrt(chi_i[0])),2.*np.pi*chi_i[1]
        v_i=np.array([np.sin(theta_i)*np.cos(phi_i),\
                      np.sin(theta_i)*np.sin(phi_i),\
                      np.cos(theta_i)])
        pos,v=pos_i,v_i

        while pos[2]<=z_max: # Photon이 Transmitted 될 때 까지 루프 진행
            chi=np.random.rand(3) # Step 2~ 에 사용할 랜덤 변수

            # Step 2
            tau=-np.log(chi[0])

            # Step 3
            L=tau*(z_max/tau_max)
            pos+=L*v

            # Step 4 : Reflected Photon 판정
```

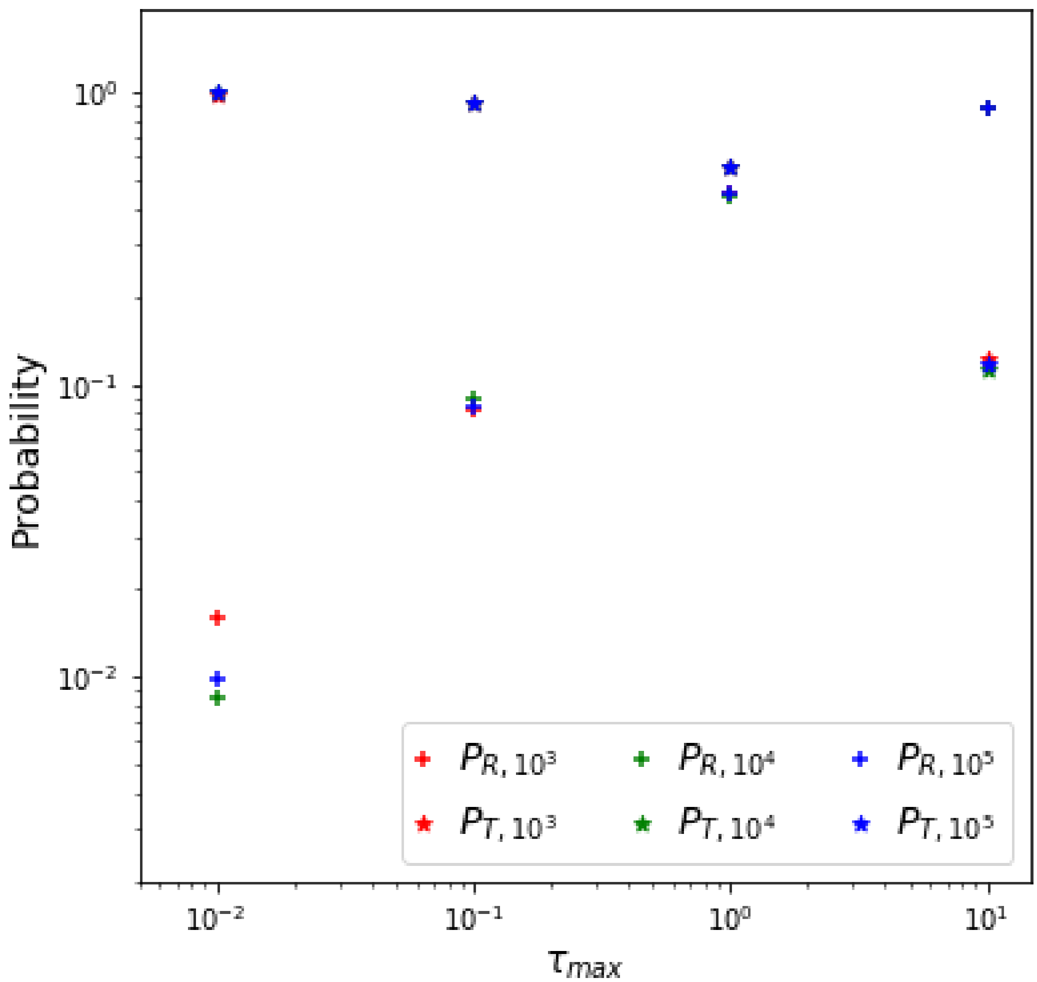
```python
            if pos[2]<0:
                count_ref+=1
                break

            # Step 5
            theta_f,phi_f=np.arccos(2.*chi[1]-1.),2.*np.pi*chi[2]
            v=np.array([np.sin(theta_f)*np.cos(phi_f),\
                        np.sin(theta_f)*np.sin(phi_f),\
                        np.cos(theta_f)])
        N+=1
    count_trans=Np-count_ref
    return count_ref/Np, count_trans/Np
```

[8]:
```python
tau_max=[0.01,0.1,1.,10.]

# N_particle : 10^3
r_m2_3,t_m2_3=photon_prob(1000,0.01)
r_m1_3,t_m1_3=photon_prob(1000,0.1)
r_0_3,t_0_3=photon_prob(1000,1.)
r_1_3,t_1_3=photon_prob(1000,10.)

prob_r_1e3=[r_m2_3,r_m1_3,r_0_3,r_1_3]
prob_t_1e3=[t_m2_3,t_m1_3,t_0_3,t_1_3]

# N_particle : 10^4
r_m2_4,t_m2_4=photon_prob(10000,0.01)
r_m1_4,t_m1_4=photon_prob(10000,0.1)
r_0_4,t_0_4=photon_prob(10000,1.)
r_1_4,t_1_4=photon_prob(10000,10.)

prob_r_1e4=[r_m2_4,r_m1_4,r_0_4,r_1_4]
prob_t_1e4=[t_m2_4,t_m1_4,t_0_4,t_1_4]

# N_particle : 10^5
r_m2_5,t_m2_5=photon_prob(100000,0.01)
r_m1_5,t_m1_5=photon_prob(100000,0.1)
r_0_5,t_0_5=photon_prob(100000,1.)
r_1_5,t_1_5=photon_prob(100000,10.)

prob_r_1e5=[r_m2_5,r_m1_5,r_0_5,r_1_5]
prob_t_1e5=[t_m2_5,t_m1_5,t_0_5,t_1_5]
```

[9]:
```python
plt.figure(figsize=(6,6))
plt.scatter(tau_max,prob_r_1e3,c='r',marker='+',label=r'$P_{R,10^3}$')
plt.scatter(tau_max,prob_t_1e3,c='r',marker='*',label=r'$P_{T,10^3}$')
plt.scatter(tau_max,prob_r_1e4,c='g',marker='+',label=r'$P_{R,10^4}$')
plt.scatter(tau_max,prob_t_1e4,c='g',marker='*',label=r'$P_{T,10^4}$')
plt.scatter(tau_max,prob_r_1e5,c='b',marker='+',label=r'$P_{R,10^5}$')
```

```
plt.scatter(tau_max,prob_t_1e5,c='b',marker='*',label=r'$P_{T,10^5}$')
plt.xlim(0.005,15.)
plt.xlabel(r'$\tau_{max}$',fontsize=14)
plt.xscale('log')
plt.ylim(2.0e-3,1.9)
plt.ylabel(r'Probability',fontsize=14)
plt.yscale('log')
plt.legend(loc=4,fontsize=14,ncol=3,handlelength=0.3)
```

[9]: <matplotlib.legend.Legend at 0x24120393668>



Incident photon의 수를 $N = 10^4$에서 $N = 10^5$로 올려도 계산된 확률에는 거의 변화가 없는 것으로 볼 때, 결과값이 충분히 수렴된 것을 확인할 수 있다.

τ_{max}가 작을 때에는, photon이 거의 충돌을 겪지 않기때문에 대부분이 Transmitted 되지만, τ_{max}가 커질 수록 Photon이 자주 충돌을 일으키기에 Reflected될 확률이 점차 증가하게 된다.

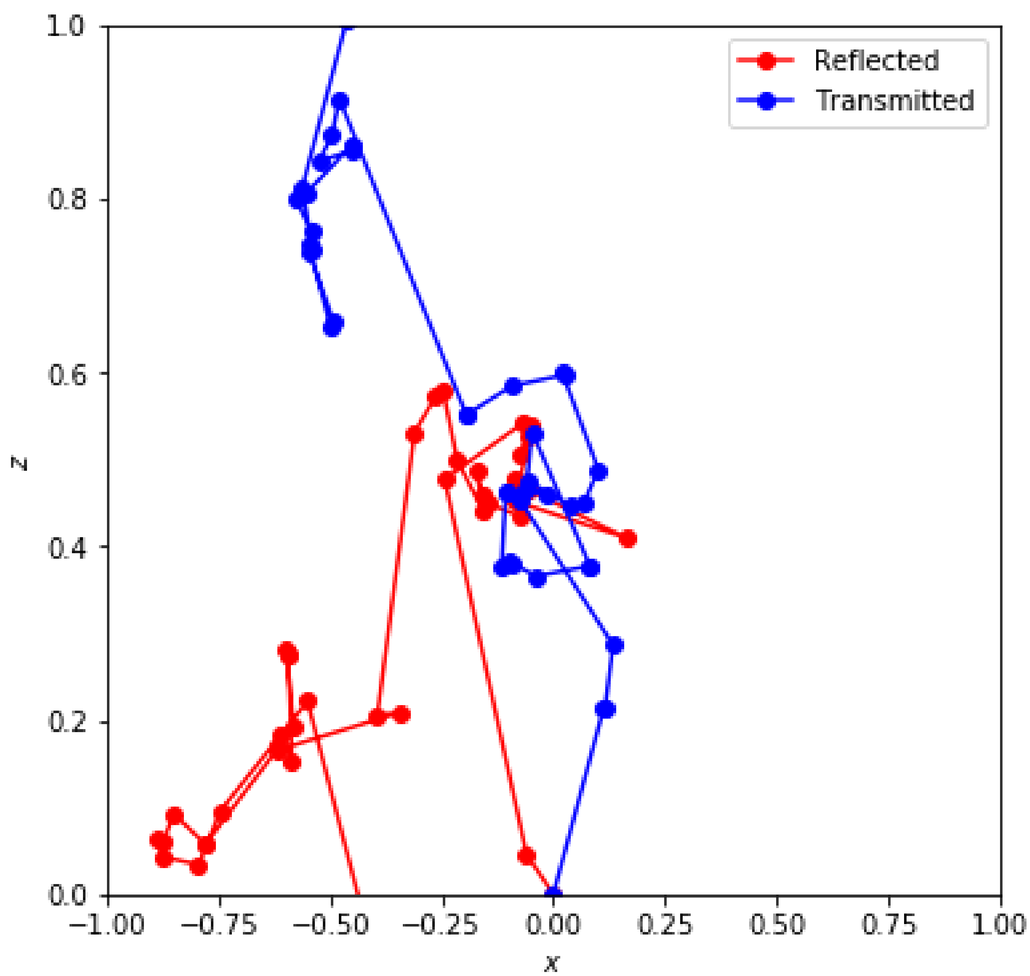
(b)

[10]:
```python
def photon_path(tau_max):
    z_max=1.
    chi_i=np.random.rand(2)  # Step 1에 사용할 랜덤 변수

    # Step 1
    pos_i=np.zeros(3)
    theta_i,phi_i=np.arccos(np.sqrt(chi_i[0])),2.*np.pi*chi_i[1]
    v_i=np.array([np.sin(theta_i)*np.cos(phi_i),\
                  np.sin(theta_i)*np.sin(phi_i),\
                  np.cos(theta_i)])
    pos,v=pos_i,v_i

    x_hist,z_hist=[],[] # Photon의 경로가 저장되는 곳
    while pos[2]<=z_max:
        x_hist.append(pos[0])
        z_hist.append(pos[2])

        chi=np.random.rand(3)

        # Step 2
        tau=-np.log(chi[0])

        # Step 3
        L=tau*(z_max/tau_max)
        pos+=L*v


        # Step 4 : Reflected Photon 판정
        if pos[2]<0:
            break

        # Step 5
        theta_f,phi_f=np.arccos(2.*chi[1]-1.),2.*np.pi*chi[2]
        v=np.array([np.sin(theta_f)*np.cos(phi_f),\
                    np.sin(theta_f)*np.sin(phi_f),\
                    np.cos(theta_f)])

    #최종 Photon의 위치 저장
    x_hist.append(pos[0])
    z_hist.append(pos[2])
    return x_hist,z_hist
```

[11]:
```python
plt.figure(figsize=(6,6))
z_r_f,z_t_f=10.,-10.

while z_r_f>0.: #Reflected Photon 경로 확인
```

```
    path_x_r,path_z_r=photon_path(10.)
    path_x_r,path_z_r=np.array(path_x_r),np.array(path_z_r)
    z_r_f=path_z_r[-1]
while z_t_f<1.: #Transmitted Photon 경로 확인
    path_x_t,path_z_t=photon_path(10.)
    path_x_t,path_z_t=np.array(path_x_t),np.array(path_z_t)
    z_t_f=path_z_t[-1]

plt.plot(path_x_r,path_z_r,marker='o',c='r',label='Reflected')
plt.plot(path_x_t,path_z_t,marker='o',c='b',label='Transmitted')

plt.xlabel(r'$x$')
plt.ylabel(r'$z$')
plt.xlim(-1.0,1.0)
plt.ylim(0.0,1.0)
plt.legend()
```

[11]: <matplotlib.legend.Legend at 0x2412045ba20>



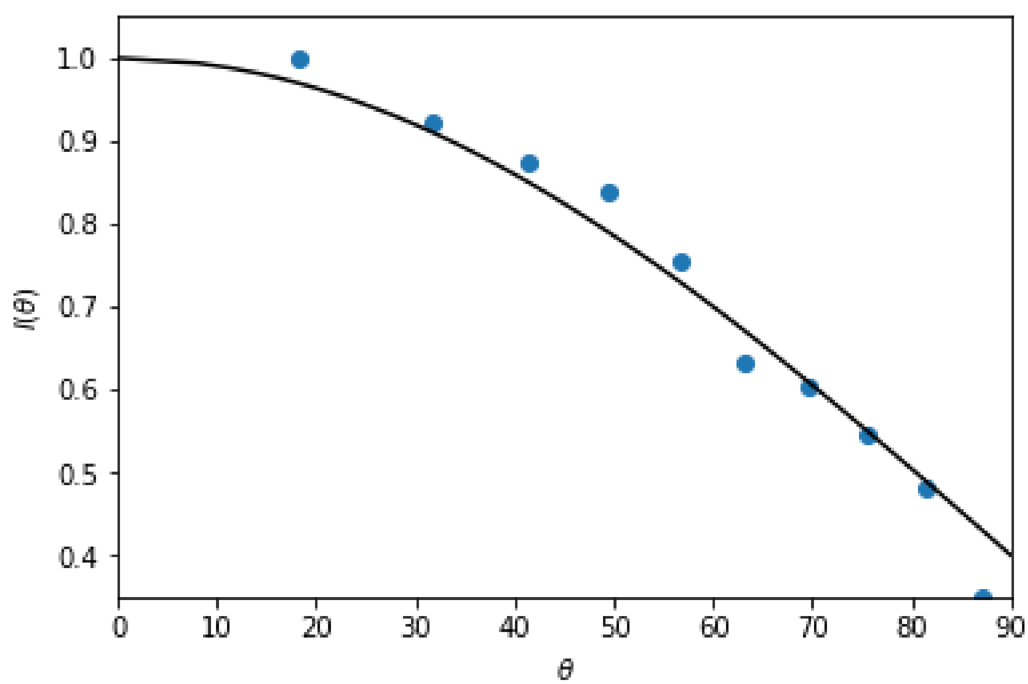
(c)

[12]:
```python
def EmergentIntensity(Np,tau_max):
    z_max=1.
    N=1
    count_ref,count_trans=0,0
    I=np.zeros(10)
    while N<=Np:
        chi_i=np.random.rand(2) # Step 1에 사용할 랜덤 변수

        # Step 1
        pos_i=np.zeros(3)
        theta_i,phi_i=np.arccos(np.sqrt(chi_i[0])),2.*np.pi*chi_i[1]
        v_i=np.array([np.sin(theta_i)*np.cos(phi_i),\
                      np.sin(theta_i)*np.sin(phi_i),\
                      np.cos(theta_i)])
        pos,v=pos_i,v_i

        trans = True
        theta_f, phi_f = theta_i, phi_i
        while pos[2]<=z_max: # Photon이 Transmitted 될 때 까지 루프 진행
            chi=np.random.rand(3) # Step 2~ 에 사용할 랜덤 변수

            # Step 2
            tau=-np.log(chi[0])

            # Step 3
            L=tau*(z_max/tau_max)
            pos+=L*v

            # Step 4 : Reflected Photon 판정
            if pos[2]<0:
                trans = False
                count_ref+=1
                break

            # Step 5

            # 현재 mu가 속한 bin의 위치 확인
            mu_f = 0.1*np.floor(10*np.cos(theta_f))+0.05
            mu_f_idx = int(np.floor(10*np.cos(theta_f)))

            theta_f,phi_f=np.arccos(2.*chi[1]-1.),2.*np.pi*chi[2]
            v=np.array([np.sin(theta_f)*np.cos(phi_f),\
                        np.sin(theta_f)*np.sin(phi_f),\
```

```
                            np.cos(theta_f)])

        if trans:
            I[mu_f_idx]+=1./(2.*mu_f*0.1)
        N+=1
    count_trans=Np-count_ref
    return I/count_trans
```

[13]:
```
I_MC = EmergentIntensity(100000,10.)

theta = np.arccos(np.arange(0.05,1.0,0.1))
theta_plt = np.arccos(np.linspace(0.,1.,101))

plt.scatter(np.rad2deg(theta),I_MC/I_MC[-1])
plt.plot(np.rad2deg(theta_plt),0.4+0.6*np.cos(theta_plt),c='k')
plt.xlim(0.0,90.0)
plt.ylim(0.35,1.05)
plt.xlabel(r'$\theta$')
plt.ylabel(r'$I(\theta)$')
```

[13]:
```
Text(0, 0.5, '$I(\\theta)$')
```



4.

[14]:
```
def RK4(func,x,y,h): #dy/dx=func(x)에 대한 RK4 Method
    hh=0.5*h
    V1=func(x,y)
    V2=func(x+hh,y+hh*V1)
    V3=func(x+hh,y-hh*V2)
    V4=func(x+h,y+h*V3)
    return y+h*(V1+2.*V2+2.*V3+V4)/6.
```

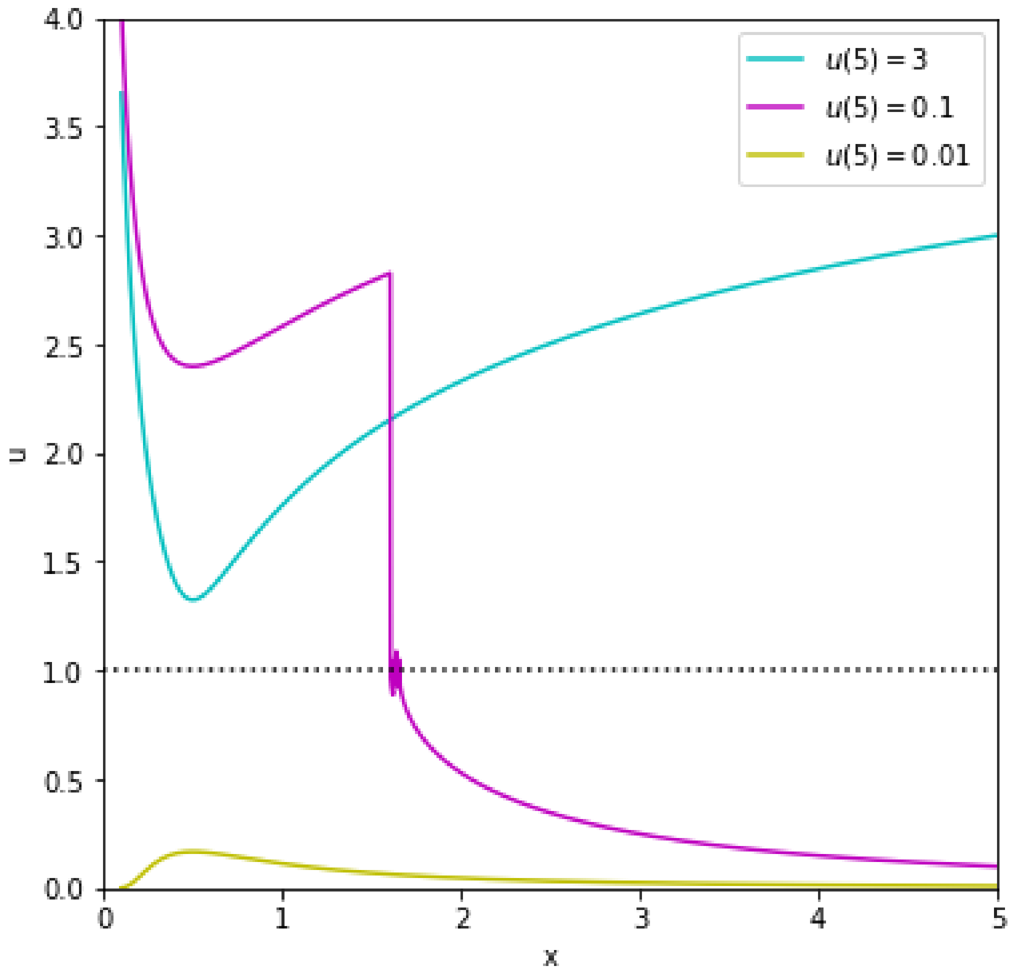
0.0.1 (a)

[15]:
```
def dudx(x,u):
    return (2./x-1./pow(x,2.))/(u-1./u)

def Bondi(x_init,x_stop,u_init,N): #x_init,u_init:초기조건,
                                   #x_stop :적분 범위, N: Step 갯수
    dx=np.abs(x_init-x_stop)/N
    u_result=np.array([u_init])
    if x_init>x_stop: # 적분 방향이 -인 경우
        i,i_f=N,0
        while i>i_f:
            x_domain=np.linspace(x_stop,x_init,N+1)
            x_i,u_i=x_domain[i],u_result[0]
            u_new=RK4(dudx,x_i,u_i,-dx)
            u_result=np.append(u_new,u_result)
            i-=1
    elif x_init<x_stop: # 적분 방향이 +인 경우
        i,i_f=0,N
        while i<i_f:
            x_domain=np.linspace(x_init,x_stop,N+1)
            x_i,u_i=x_domain[i],u_result[-1]
            u_new=RK4(dudx,x_i,u_i,dx)
            u_result=np.append(u_result,u_new)
            i+=1
    return x_domain,u_result #x와 u를 동시에 출력
```

[16]:
```
xdom_1,u_1=Bondi(5.0,0.1,3.,10000)
xdom_2,u_2=Bondi(5.0,0.1,0.1,10000)
xdom_3,u_3=Bondi(5.0,0.1,0.01,10000)

plt.figure(figsize=(6,6))
plt.plot(xdom_1,u_1,label=r'$u(5)=3$',c='c')
plt.plot(xdom_2,u_2,label=r'$u(5)=0.1$',c='m')
plt.plot(xdom_3,u_3,label=r'$u(5)=0.01$',c='y')
plt.axhline(1.0,c='k',ls=':')
plt.xlim(0.0,5.0)
```

```
plt.ylim(0.0,4.0)
plt.xlabel('x')
plt.ylabel('u')
plt.legend()
```

[16]: <matplotlib.legend.Legend at 0x2412052c8d0>



u(5)=0.1일때 ODE를 푸는 과정에서 $u = 1$인 지점을 지나가게 되는데, 이 경우에 $\frac{du}{dx} = \frac{2/x-1/x^2}{u-1/u}$를 계산하면 분모가 0이 되므로 문제가 발생하게 된다.

이는 $\frac{du}{dx}$가 아닌 $\frac{dx}{du}$를 품으로써 해결가능하다.

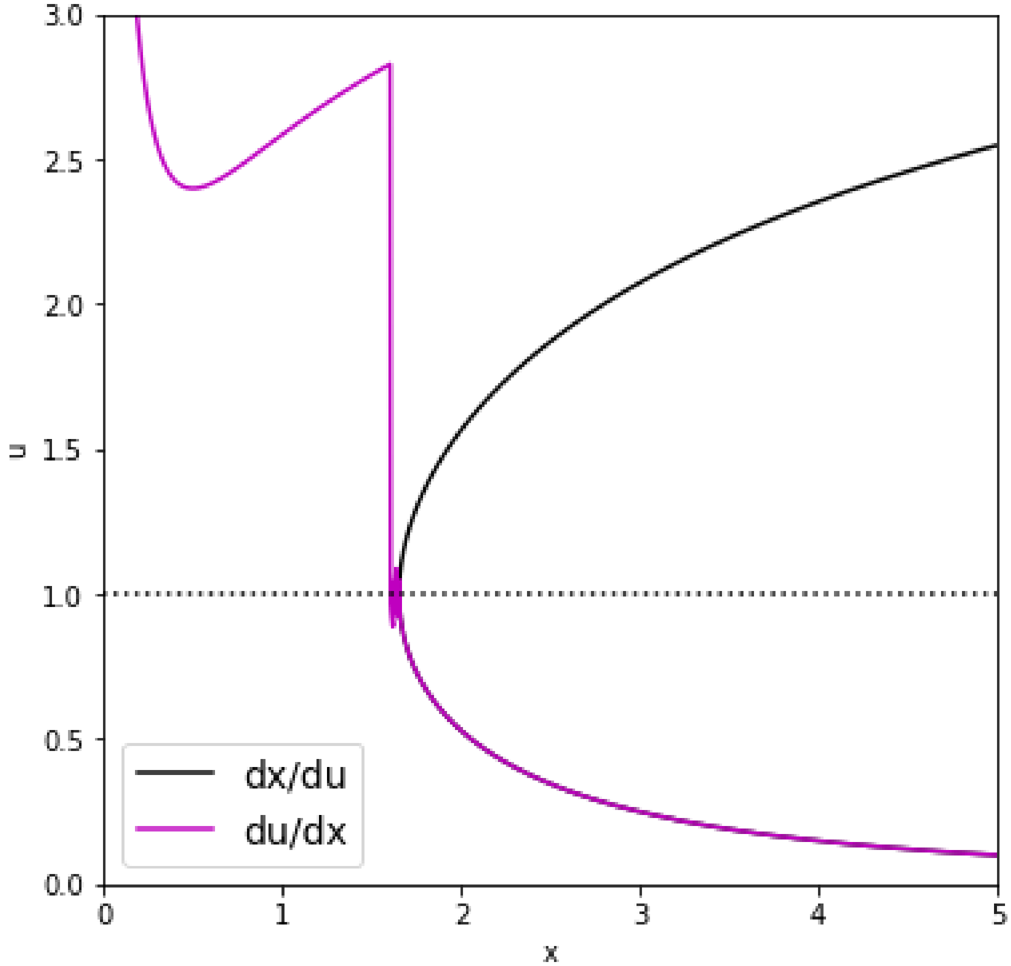
$$\frac{dx}{du} = \frac{u - 1/u}{2/x - 1/x^2} \tag{1}$$

[17]:
```
def dxdu(u,x): #역함수에 대한 ODE
    return (u-1./u)/(2./x-1./pow(x,2.))
```

```
def Bondi2(u_init,u_stop,x_init,N):
    du=np.abs(u_init-u_stop)/N
    x_result=np.array([x_init])
    if u_init>u_stop:
        i,i_f=N,0
        while i>i_f:
            u_domain=np.linspace(u_stop,u_init,N+1)
            u_i,x_i=u_domain[i],x_result[0]
            x_new=RK4(dxdu,u_i,x_i,-du)
            x_result=np.append(x_new,x_result)
            i-=1
    elif u_init<u_stop:
        i,i_f=0,N
        while i<i_f:
            u_domain=np.linspace(u_init,u_stop,N+1)
            u_i,x_i=u_domain[i],x_result[-1]
            x_new=RK4(dxdu,u_i,x_i,du)
            x_result=np.append(x_result,x_new)
            i+=1
    return x_result,u_domain
```

[18]:
```
xdom_rev,u_rev=Bondi2(0.1,4.0,5.0,10000)

plt.figure(figsize=(6,6))
plt.plot(xdom_rev,u_rev,label='dx/du',c='k')
plt.plot(xdom_2,u_2,label='du/dx',c='m')
plt.axhline(1.0,c='k',ls=':')
plt.xlim(0.0,5.0)
plt.ylim(0.0e-3,3.0)
plt.xlabel('x')
plt.ylabel('u')
plt.legend(fontsize=14)
```

[18]:
```
<matplotlib.legend.Legend at 0x241206459e8>
```



(b)

$\frac{du}{dx} = \frac{2/x-1/x^2}{u-1/u} = \frac{u}{x^2(u+1)}\frac{2x-1}{u-1}$이므로, $x = 1/2, u = 1$ 근처에서는 다음과 같이 표현된다.

$$\frac{\Delta u}{\Delta x} \approx \frac{2\Delta x}{\Delta u}\frac{1}{1/2^2 \times 2} \tag{2}$$
$$\approx 4\frac{\Delta x}{\Delta u} \tag{3}$$

그러므로 $\Delta u/\Delta x \approx \pm 2$가 됨을 예측할 수 있다.

[19]:
```
def Bondi_tr(x_start,x_stop,u_start,N,pm): # Transonic Solution을 구하기 위한 함수
    dx=np.abs(x_start-x_stop)/N
    u_result=np.array([u_start])
    if x_start>x_stop:
        i,i_f=N,0
```

```python
        while i>i_f:
            x_domain=np.linspace(x_stop,x_start,N+1)
            x_i,u_i=x_domain[i],u_result[0]
            if x_i==0.5 and u_i==1.: # x=1/2, u=1 점에서의 du/dx 값 지정
                if pm=='+':
                    u_new=u_i-2.*dx #du/dx=+2인 경우
                elif pm=='-':
                    u_new=u_i+2.*dx #du/dx=+2인 경우
            else:
                u_new=RK4(dudx,x_i,u_i,-dx)
            u_result=np.append(u_new,u_result)
            i-=1
    elif x_start<x_stop:
        i,i_f=0,N
        while i<i_f:
            x_domain=np.linspace(x_start,x_stop,N+1)
            x_i,u_i=x_domain[i],u_result[-1]
            if x_i==0.5 and u_i==1.:
                if pm=='+':
                    u_new=u_i+2.*dx #du/dx=+2인 경우
                elif pm=='-':
                    u_new=u_i-2.*dx #du/dx=+2인 경우
            else:
                u_new=RK4(dudx,x_i,u_i,dx)
            u_result=np.append(u_result,u_new)
            i+=1
    return x_domain,u_result
```

[20]:

```python
# du/dx=+2인 해
xdom_1i,trson_1i=Bondi_tr(0.5,0.1,1.,10000,'+') #범위 0.1~0.5
xdom_1o,trson_1o=Bondi_tr(0.5,5.0,1.,10000,'+') #범위 0.5~5.0
xdom_1,trson_1=np.append(xdom_1i,xdom_1o),np.append(trson_1i,trson_1o)
# du/dx=-2인 해
xdom_2i,trson_2i=Bondi_tr(0.5,0.1,1.,10000,'-') #범위 0.1~0.5
xdom_2o,trson_2o=Bondi_tr(0.5,5.0,1.,10000,'-') #범위 0.5~5.0
xdom_2,trson_2=np.append(xdom_2i,xdom_2o),np.append(trson_2i,trson_2o)
```

[21]:

```python
plt.figure(figsize=(6,6))

plt.plot(xdom_1,trson_1,c='r',label=r'$du/dx|_{(x,u)=(1/2,1)}=+2$')
plt.plot(xdom_2,trson_2,c='b',label=r'$du/dx|_{(x,u)=(1/2,1)}=-2$')
plt.axhline(1.0,xmax=0.10,c='k',ls=':')
plt.axvline(0.5,ymax=0.25,c='k',ls=':')
plt.xlim(0.0,5.0)
plt.ylim(0.0,4.0)
plt.xlabel(r'$x$',fontsize=14)
plt.ylabel(r'$u$',fontsize=14)
plt.legend(fontsize=14)
```

```
plt.tight_layout()
```