# 전산천문학 HW5 Solution

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from scipy import constants as c
import warnings
warnings.filterwarnings(action='ignore')
```

### 1.

#### (a)

문제에서 주어진 것처럼 $u(\xi)$를 power series ($u(\xi) = \sum_{m=0}^{\infty} a_m \xi^m$) 로 표현하면 (1)번 식을 다음과 같이 정리할 수 있다.

$$\begin{aligned}
\frac{1}{\xi^2}\frac{d}{d\xi}\left(\xi^2\frac{du}{d\xi}\right) &= \sum_{m=0}^{\infty} m(m+1)a_m\xi^{m-2} && (1)\\
&= \frac{2a_1}{\xi} + 6a_2 + 12a_3\xi + 20a_4\xi^2 + \cdots && (2)\\
e^{-u} &= \sum_{m=0}^{\infty}\frac{(-1)^m}{m!}u^m && (3)\\
&= \sum_{m=0}^{\infty}\frac{(-1)^m}{m!}\left[a_0^m + \xi\cdot\binom{m}{1}a_0^{m-1}a_1 + \xi^2\cdot\left\{\binom{m}{1}a_0^{m-1}a_2 + \binom{m}{2}a_0^{m-2}a_1^2\right\} + \cdots\right] && (4)\\
&= \sum_{m=0}^{\infty}\frac{(-1)^m}{m!}a_0^m - \xi\cdot a_1\sum_{m=1}^{\infty}\frac{(-1)^{m-1}}{(m-1)!}a_0^{m-1}\\
&\quad + \xi^2\cdot\left[\frac{a_1^2}{2}\sum_{m=2}^{\infty}\frac{(-1)^{m-2}}{(m-2)!}a_0^{m-2} - a_2\sum_{m=1}^{\infty}\frac{(-1)^{m-1}}{(m-1)!}a_0^{m-1}\right] + \cdots && (5)\\
&= e^{-a_0}\left[1 - a_1\xi + \left(\frac{a_1^2}{2} - a_2\right)\xi^2 + \cdots\right] && (6)
\end{aligned}$$

경계 조건에 의하면 $u(0) = 0, u'(0) = 0$이므로 $a_0 = a_1 = 0$이고 위에서 정리한 좌변과 우변의 계수를 서로 비교하면 다음과 같다.

$$6a_2 = e^{-a_0} \rightarrow a_2 = \frac{1}{6} \tag{7}$$

$$12a_3 = -a_1 e^{-a_0} \rightarrow a_3 = 0 \tag{8}$$

$$20a_4 = \left(\frac{a_1^2}{2} - a_2\right) e^{-a_0} \rightarrow a_4 = -\frac{1}{120} \tag{9}$$

**(b)**

문제에서 주어진 미분방정식을 정리하면 다음과 같다.

$$\frac{d^2u}{d\xi^2} = e^{-u} - \frac{2}{\xi}\frac{du}{d\xi} \tag{10}$$

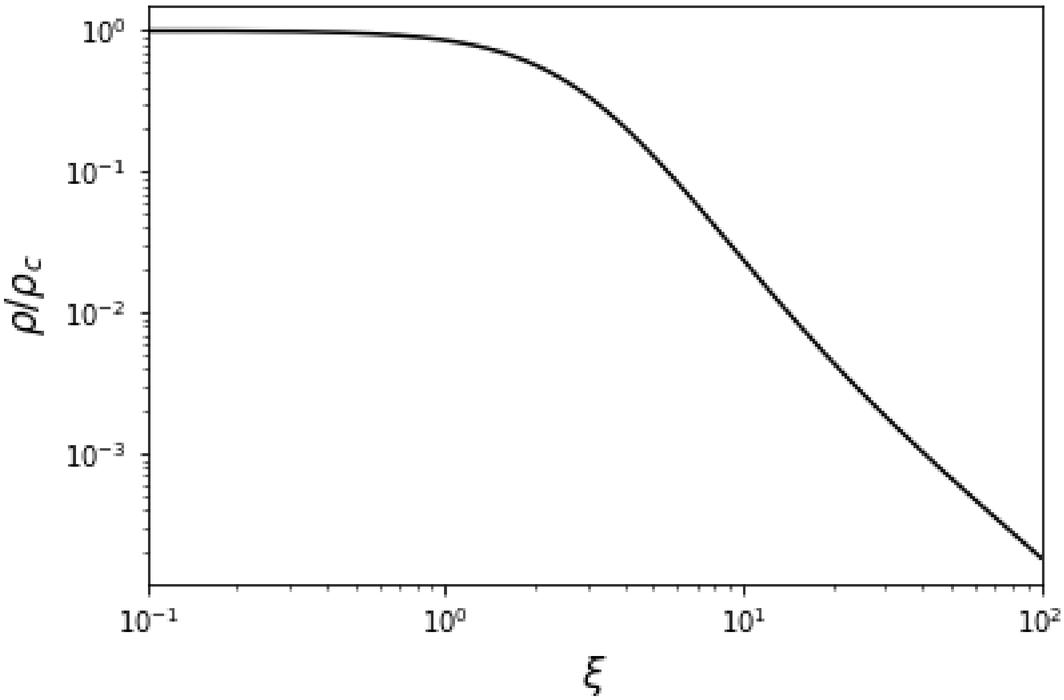
이 때, $\frac{du}{d\xi}$항에 분모에 $\xi$가 붙어있기 때문에 0에서 정의되지 않는다. 하지만 앞에서 살펴본 $u(\xi)$의 power series를 이용하면 $\lim_{\xi \to 0} \frac{1}{\xi}\frac{du}{d\xi} = 2a_2$이므로 다음과 같은 과정을 통해 미분방정식을 풀 수 있다.

[3]:
```
def BonnorEbert(u,xi):
    du_dxi0 = u[1]
    du_dxi1 = np.exp(-u[0])-2.*u[1]/xi
    return [du_dxi0,du_dxi1]
```

[4]:
```
dxi = 0.001
xi_domain = np.arange(dxi,100.+dxi,dxi)
u_ini = np.array([dxi**2./6.,dxi/3.])

BE_result = odeint(BonnorEbert,u_ini,xi_domain)
u_result = BE_result[:,0]
du_result = BE_result[:,1]
```

[5]:
```
dens = np.exp(-u_result) # 방정식을 통해 구한 u를 rho (Density)로 변환!

plt.loglog(xi_domain,dens,c='k')
plt.xlim(0.1,100.)
plt.xlabel(r'$\xi$',fontsize=15)
plt.ylabel(r'$\rho/\rho_c$',fontsize=15)
```

[5]:
```
Text(0, 0.5, '$\\rho/\\rho_c$')
```




**(c)**

(2)번식에 대해서 (1)번 식을 이용하여 정리하면 다음과 같다.

$$m = \frac{1}{\sqrt{4\pi}} \int_0^{\xi} \xi'^2 e^{-u} d\xi' \tag{11}$$

$$= \frac{1}{\sqrt{4\pi}} \int_0^{\xi} \frac{d}{d\xi'} \left( \xi'^2 \frac{du}{d\xi'} \right) d\xi' \tag{12}$$

$$= \frac{1}{\sqrt{4\pi}} \xi^2 \frac{du}{d\xi} \tag{13}$$

이를 통해 (b)에서 구한 $du/d\xi$ 값을 이용하여 $m$을 표현할 수 있다는 걸 알 수 있다. 이 점을 유의하여 $p \equiv m^2 e^{-u}$ 그리고 $r \equiv (\xi\, du/d\xi)^{-1}$을 구하면 다음과 같다.

[6]:
```
m = np.power(xi_domain,2.)*du_result/np.sqrt(4.*np.pi)
p = np.power(m,2)*np.exp(-u_result)
r = 1./(xi_domain*du_result)
```

이렇게 구한 $m, p, r$값에서 $p$값이 최대가 되는 $\xi$와 $r$의 위치는 다음과 같다.

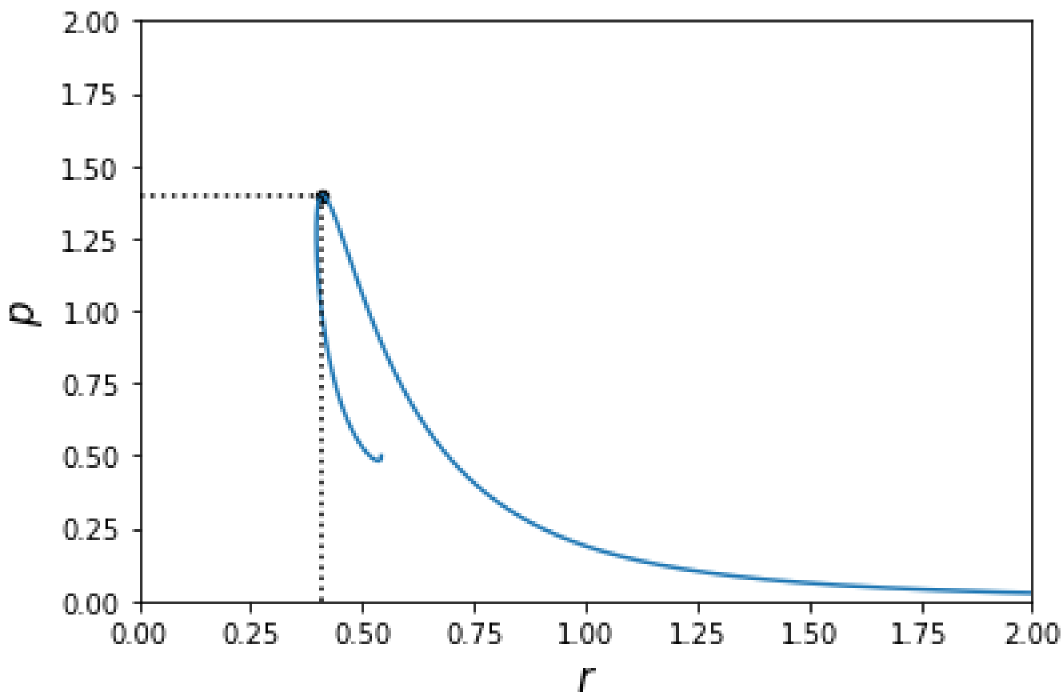
[12]:
```
p_max = np.max(p)
r_max = r[np.argmax(p)]
xi_max = xi_domain[np.argmax(p)]
```

```
print ('p_max = {:.5f} at r = {:.5f} (xi = {:.3f})'.format(p_max,r_max,xi_max))
```

```
p_max = 1.39766 at r = 0.41076 (xi = 6.451)
```

[13]:
```
plt.plot(r,p)
plt.scatter(r_max,p_max,c='k',s=20)
plt.axhline(p_max,xmax=r_max/2.,c='k',ls=':')
plt.axvline(r_max,ymax=p_max/2.,c='k',ls=':')
plt.xlim(0.,2.)
plt.ylim(0.,2.)
plt.xlabel('$r$',fontsize=15)
plt.ylabel('$p$',fontsize=15)
```

[13]:
```
Text(0, 0.5, '$p$')
```




**2.**

**(a)**

(5)번 식을 정리하면 다음과 같다.

$$\frac{da}{dt} = aH_0 \left[ \frac{\Omega_R}{a^4} + \frac{\Omega_M}{a^3} + \frac{\Omega_k}{a^2} + \Omega_\Lambda \right]^{1/2} \tag{14}$$

이 문제를 푸는 데 있어서 사용할 수 있는 경계조건은 $a(0) = 0$이다. 하지만 위의 식은 $a = 0$일때 $da/dt$가 무한대가 되는 문제가 있다.

하지만 이는 이전 과제에서와 같이 역함수를 이용하면 해결 가능하다.

$$\frac{dt}{da} = \frac{1}{aH_0}\left[\frac{\Omega_R}{a^4} + \frac{\Omega_M}{a^3} + \frac{\Omega_k}{a^2} + \Omega_\Lambda\right]^{-1/2} \tag{15}$$
$$= \frac{a}{H_0}(\Omega_R + \Omega_M a + \Omega_k a^2 + \Omega_\Lambda a^4)^{-1/2} \tag{16}$$

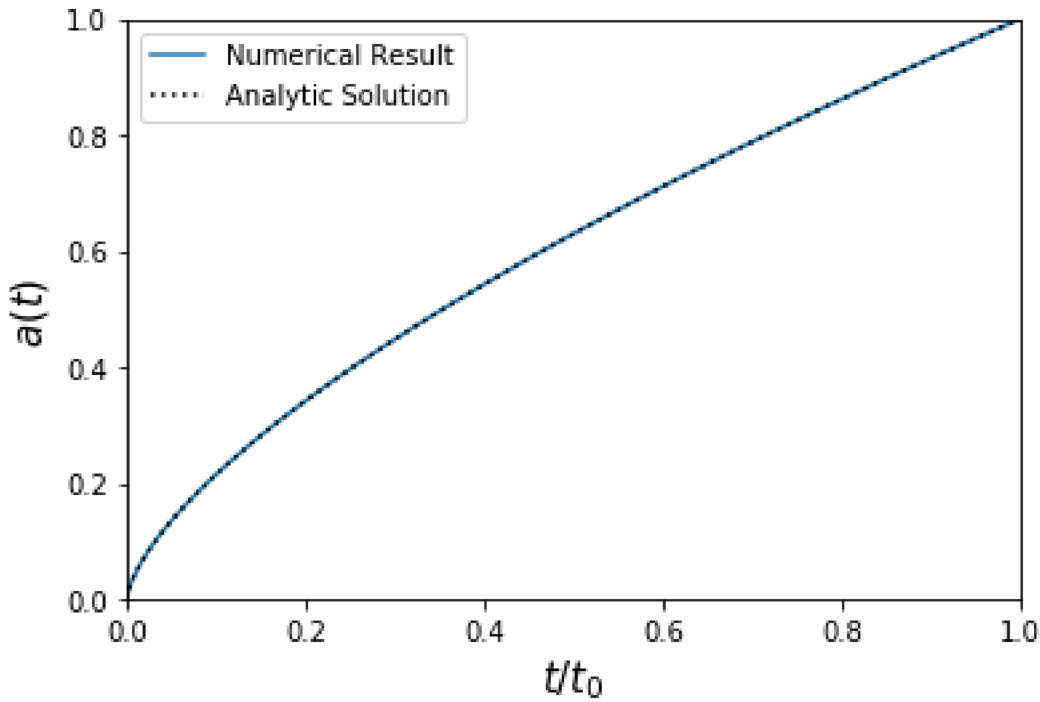
[14]:
```
H0 = 71.
def Friedmann_inv(t,a,Omega):
    if a!=0:
        dt_da = (a/H0)/np.
↪sqrt(Omega[0]+Omega[1]*a+Omega[2]*(a**2)+Omega[3]*(a**4))
    else:
        dt_da = 0.
    return dt_da
```

문제에서 주어진 조건인 $\Omega_R = \Omega_k = \Omega_\Lambda = 0$, $\Omega_M = 1$을 대입하면 다음과 같이 미분방정식을 풀 수 있다.

[15]:
```
t0 = 2./(3.*H0)
a_domain = np.linspace(0., 1., 10000)
t_ini = [0.0]

t_MassDom = odeint(Friedmann_inv,t_ini,a_domain,args=([0.,1.,0.,0.],))
```

[16]:
```
plt.plot(t_MassDom/t0,a_domain,label='Numerical Result')
plt.plot(t_MassDom/t0,np.power(t_MassDom/t0,2./3.),c='k',ls=':',label='Analytic␣
↪Solution')
plt.xlim(0.,1.)
plt.ylim(0.,1.)
plt.xlabel(r'$t/t_0$',fontsize=15)
plt.ylabel(r'$a(t)$',fontsize=15)
plt.legend()
```

[16]: <matplotlib.legend.Legend at 0x28db66a35c0>




**(b)**

[17]:
```
a_domain = np.linspace(0., 1., 10000)
t_ini = [0.0]

t_Current = odeint(Friedmann_inv,t_ini,a_domain,args=([3.0e-5,0.27,0.,0.73],))
```

$H_0$의 단위인 km $s^{-1}$ $Mpc^{-1}$는 $\approx 3.24 \times 10^{-20} s^{-1}$에 해당하므로 $1/H_0 \approx 13.78$Gyr에 해당한다.

[18]:
```
t_Hubble = 1./(H0*1.0e3/(1.0e6*c.parsec))/(1.0e9*c.year)
print ('1/H_0 = %.2f Gyr' % (t_Hubble))
```

```
1/H_0 = 13.78 Gyr
```

[19]:
```
print ('Current age of the universe is {:.2f} Gyr'.
 ↪format(t_Current[-1,0]*H0*t_Hubble))
```

```
Current age of the universe is 13.68 Gyr
```

**(c)**

(b)번 문제에서 구한 $a(t)$를 이용하여 $d^2a/dt^2$를 구하면 다음과 같다.

$$\frac{da}{dt} = \frac{H_0}{a}(\Omega_R + \Omega_M a + \Omega_k a^2 + \Omega_\Lambda a^4)^{1/2} \tag{17}$$

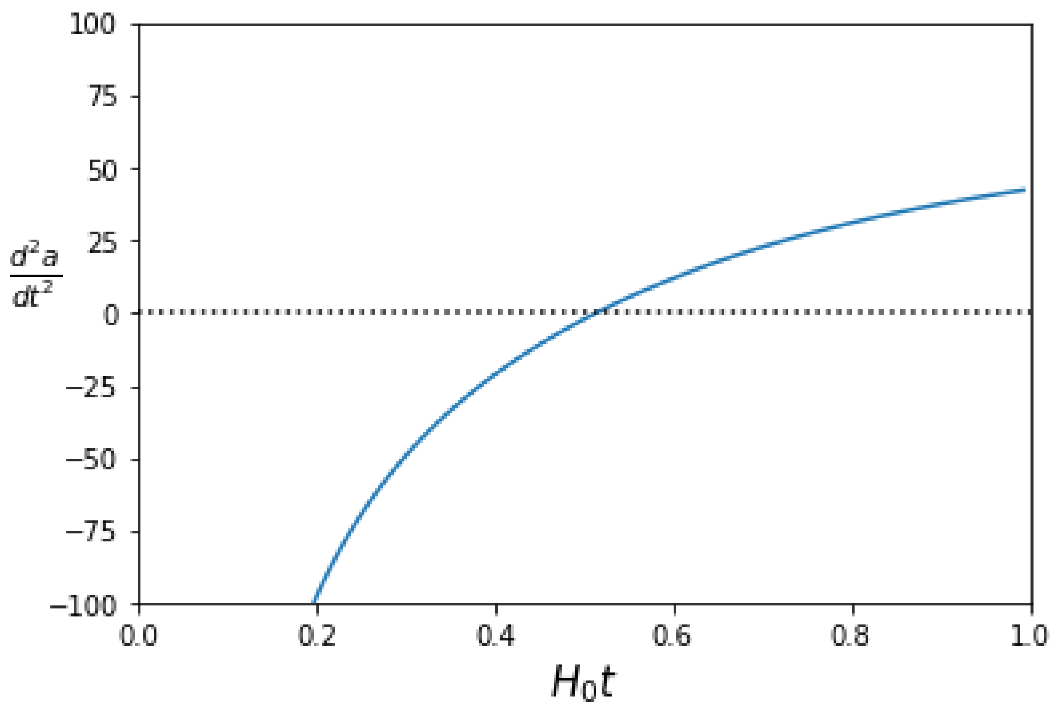
$$\frac{d^2a}{dt^2} = -\frac{H_0}{a^2}(\Omega_R + \Omega_M a + \Omega_k a^2 + \Omega_\Lambda a^4)^{1/2} + \frac{H_0}{2a}(\Omega_R + \Omega_M a + \Omega_k a^2 + \Omega_\Lambda a^4)^{-1/2}(\Omega_M + 2\Omega_k a + 4\Omega_\Lambda a^3) \tag{18}$$

$$= -\frac{1}{a}\frac{da}{dt} + \frac{H_0^2}{2a^2}\left(\frac{da}{dt}\right)^{-1}(\Omega_M + 2\Omega_k a + 4\Omega_\Lambda a^3) \tag{19}$$

[20]:
```
da_dt = (H0/a_domain)*np.sqrt(3.0e-5+0.27*a_domain+0.73*np.power(a_domain,4))
d2a_dt2 = -da_dt/a_domain+0.5*pow(H0/a_domain,2.)*(0.27+4.*0.73*np.
 ↪power(a_domain,3.))/da_dt
```

[21]:
```
plt.plot(t_Current*H0,d2a_dt2)
plt.axhline(0.0,c='k',ls=':')
plt.xlim(0.,1.)
plt.ylim(-1.0e2,1.0e2)
plt.xlabel(r'$H_0t$',fontsize=15)
plt.ylabel(r'$\frac{d^2a}{dt^2}$',fontsize=15,rotation=0)
```

[21]: `Text(0, 0.5, '$\\frac{d^2a}{dt^2}$')`




$d^2a/dt^2$가 0을 지나는 위치는 이 두가지 Index 사이에 존재한다.

[22]:
```
idx_in = np.where(d2a_dt2<0.)[0][-1]
idx_out = idx_in+1
print (idx_in,idx_out)
```

```
5697 5698
```

이 두 위치에서의 시간 $t$를 구하고, Linear Interpolation을 이용해 Expansion이 바뀌는 지점의 시간 $t_{switch}$를 계산할 수 있다.

[23]:
```
t_in,t_out = t_Current[idx_in],t_Current[idx_out]
d2a_in,d2a_out = d2a_dt2[idx_in],d2a_dt2[idx_out]

t_switch = t_in-d2a_in*(t_out-t_in)/(d2a_out-d2a_in)
print ('The universe starts acclerating expansion at t = %.2f Gyr ' %␣
 ↪(t_switch*H0*t_Hubble))
```

```
The universe starts acclerating expansion at t = 7.08 Gyr
```

이렇게 구한 $t_{switch}$에 대해 $a(t_{switch})$, 그리고 $da/dt|_{t=t_{switch}}$를 구하고 이를 이용하여 Redshift와 Hubble Parameter를 계산하면 다음과 같다.

[26]:
```
a_in,a_out = a_domain[idx_in],a_domain[idx_out]
a_switch = a_in+(a_out-a_in)*(t_switch-t_in)/(t_out-t_in)

da_in,da_out = da_dt[idx_in],da_dt[idx_out]
da_switch = da_in+(da_out-da_in)*(t_switch-t_in)/(t_out-t_in)

z_switch = 1./a_switch-1.
H_switch = da_switch/a_switch
print ('z(t_switch) = %.3f, H(t_switch) = %.2f km s-1 Mpc-1' %␣
 ↪(z_switch,H_switch))
```

```
z(t_switch) = 0.755, H(t_switch) = 105.06 km s-1 Mpc-1
```

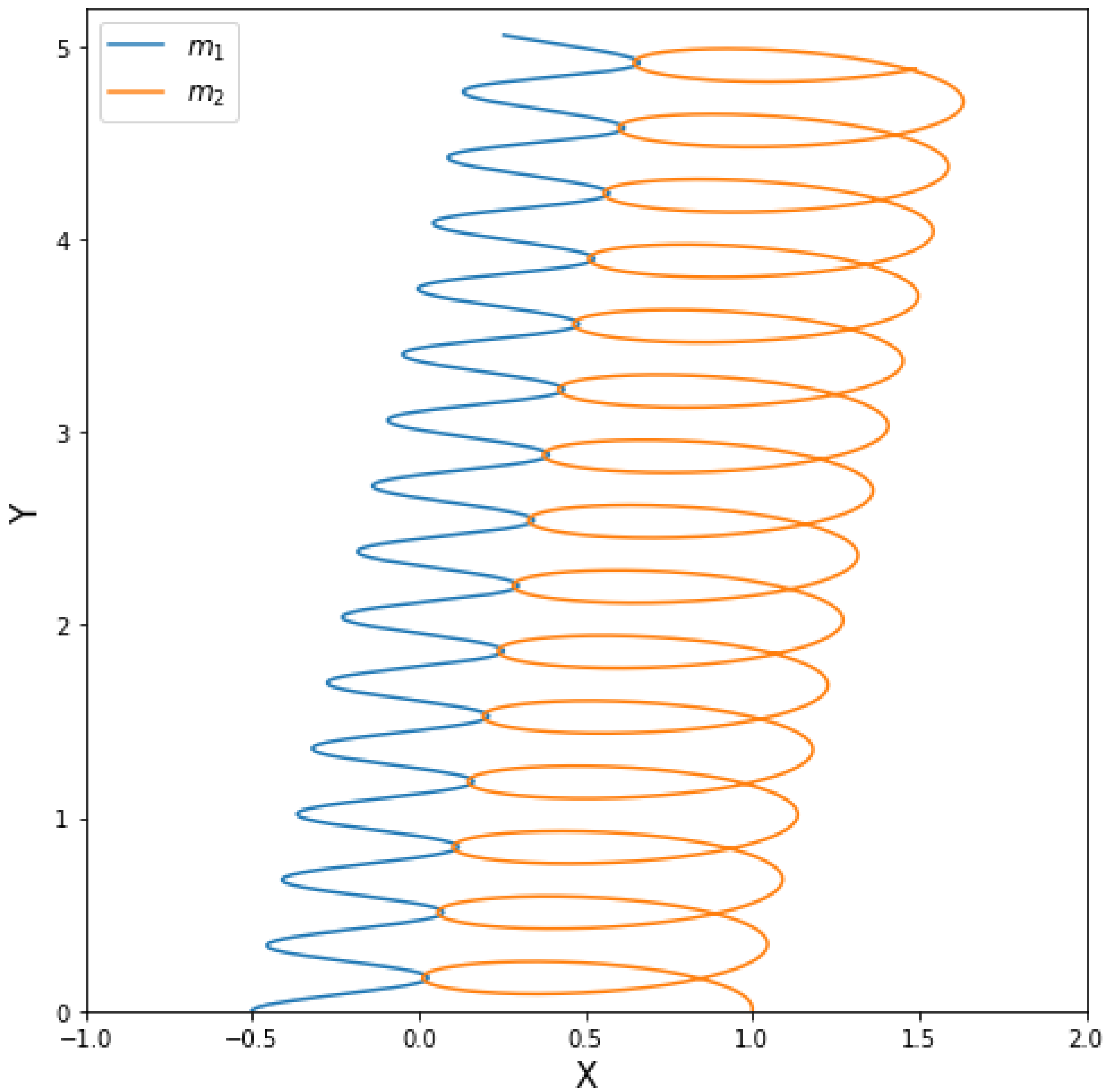
**3.**

**(a)**

[27]:
```
m=[1.,0.5]
xini, yini = [-0.5,1.], [0.,0.]
vxini, vyini = [0.01,0.02], [0.05,0.2]
tini,tmax = 0.,50.

def accel(x,y,t):
    dist = np.sqrt((x[0]-x[1])**2.+(y[0]-y[1])**2.)

    d2x_1 = -m[1]*(x[0]-x[1])/pow(dist,3.)
```

```
    d2y_1 = -m[1]*(y[0]-y[1])/pow(dist,3.)
    d2x_2 = -m[0]*(x[1]-x[0])/pow(dist,3.)
    d2y_2 = -m[0]*(y[1]-y[0])/pow(dist,3.)

    return np.array([d2x_1,d2x_2]),np.array([d2y_1,d2y_2])

x, y = xini, yini
vx, vy = vxini, vyini
t, dt = tini, 2.0e-4

xhist,yhist = np.array(xini),np.array(yini)
vxhist,vyhist = np.array(vxini),np.array(vyini)

while (t<=tmax):
    fx,fy = accel(x,y,t) #opening kick
    vhx,vhy = vx + fx*dt/2., vy + fy*dt/2.
    t+=dt/2

    x,y = x + vhx*dt, y + vhy*dt #drift

    fx,fy = accel(x,y,t)
    vx,vy = vhx + fx*dt/2., vhy + fy*dt/2. #resync

    xhist,yhist = np.vstack((xhist,x)),np.vstack((yhist,y))
    vxhist,vyhist = np.vstack((vxhist,vhx)),np.vstack((vyhist,vhy))

    t += dt/2.
```

[28]:
```
plt.figure(figsize=(7,7))
plt.plot(xhist[:,0],yhist[:,0],label='$m_1$')
plt.plot(xhist[:,1],yhist[:,1],label='$m_2$')
plt.xlim(-1,2.0)
plt.ylim(0.,5.2)
plt.xlabel('X',fontsize=15)
plt.ylabel('Y',fontsize=15)
plt.legend(loc=2,fontsize=12)
plt.tight_layout()
```

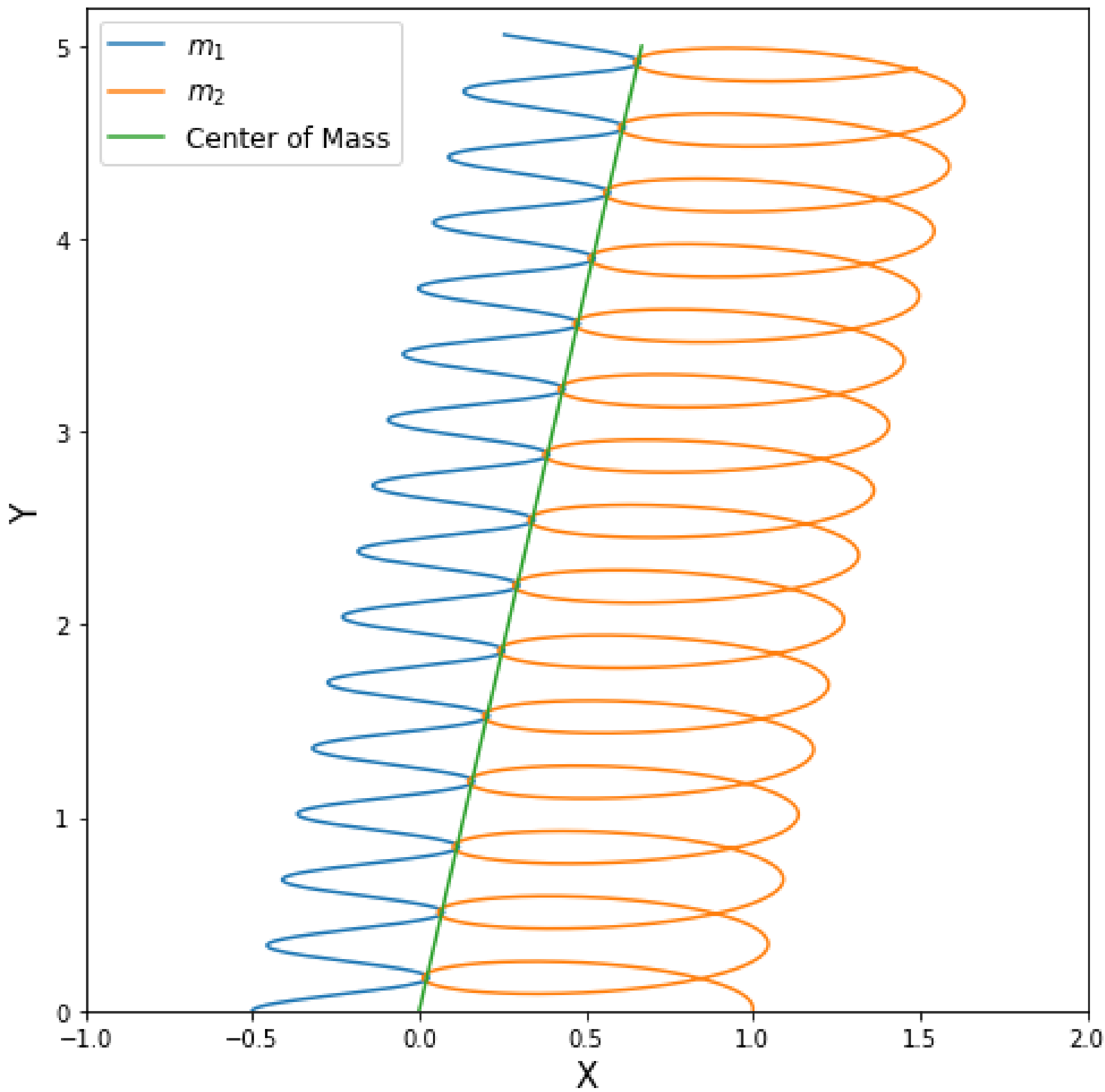



**(b)**

[29]:
```
CoM_x = (m[0]*xhist[:,0]+m[1]*xhist[:,1])/(m[0]+m[1])
CoM_y = (m[0]*yhist[:,0]+m[1]*yhist[:,1])/(m[0]+m[1])
```

[30]:
```
plt.figure(figsize=(7,7))
plt.plot(xhist[:,0],yhist[:,0],label='$m_1$')
plt.plot(xhist[:,1],yhist[:,1],label='$m_2$')
plt.plot(CoM_x,CoM_y,label='Center of Mass')
plt.xlim(-1,2.0)
plt.ylim(0.,5.2)
plt.xlabel('X',fontsize=15)
plt.ylabel('Y',fontsize=15)
```

```
plt.legend(loc=2,fontsize=12)
plt.tight_layout()
```



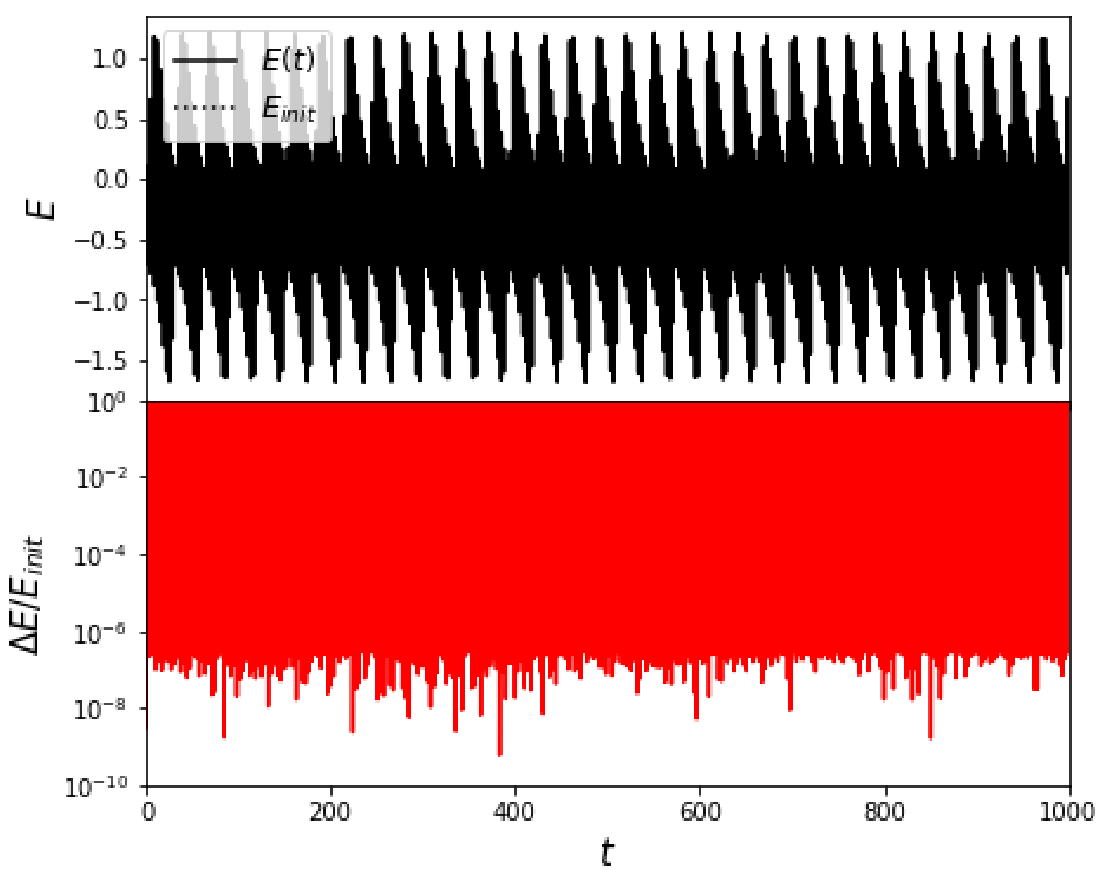

**(c)**

[ ]:
```
m=[1.,0.5]
xini, yini = [-0.5,1.], [0.,0.]
vxini, vyini = [0.01,0.02], [0.05,0.2]
tini,tmax = 0.,1000.

def accel(x,y,t):
    dist = np.sqrt((x[0]-x[1])**2.+(y[0]-y[1])**2.)
```

```python
    d2x_1 = -m[1]*(x[0]-x[1])/pow(dist,3.)
    d2y_1 = -m[1]*(y[0]-y[1])/pow(dist,3.)
    d2x_2 = -m[0]*(x[1]-x[0])/pow(dist,3.)
    d2y_2 = -m[0]*(y[1]-y[0])/pow(dist,3.)

    return np.array([d2x_1,d2x_2]),np.array([d2y_1,d2y_2])

x, y = xini, yini
vx, vy = vxini, vyini
t, dt = tini, 5.0e-4

xhist,yhist = np.array(xini),np.array(yini)
vxhist,vyhist = np.array(vxini),np.array(vyini)

while (t<=tmax):
    fx,fy = accel(x,y,t) #opening kick
    vhx,vhy = vx + fx*dt/2., vy + fy*dt/2.
    t+=dt/2

    x,y = x + vhx*dt, y + vhy*dt #drift

    fx,fy = accel(x,y,t)
    vx,vy = vhx + fx*dt/2., vhy + fy*dt/2. #resync

    xhist,yhist = np.vstack((xhist,x)),np.vstack((yhist,y))
    vxhist,vyhist = np.vstack((vxhist,vhx)),np.vstack((vyhist,vhy))

    t += dt/2.

# np.savetxt('xhist.txt',xhist)
# np.savetxt('yhist.txt',yhist)
# np.savetxt('vxhist.txt',vxhist)
# np.savetxt('vyhist.txt',vyhist)
```

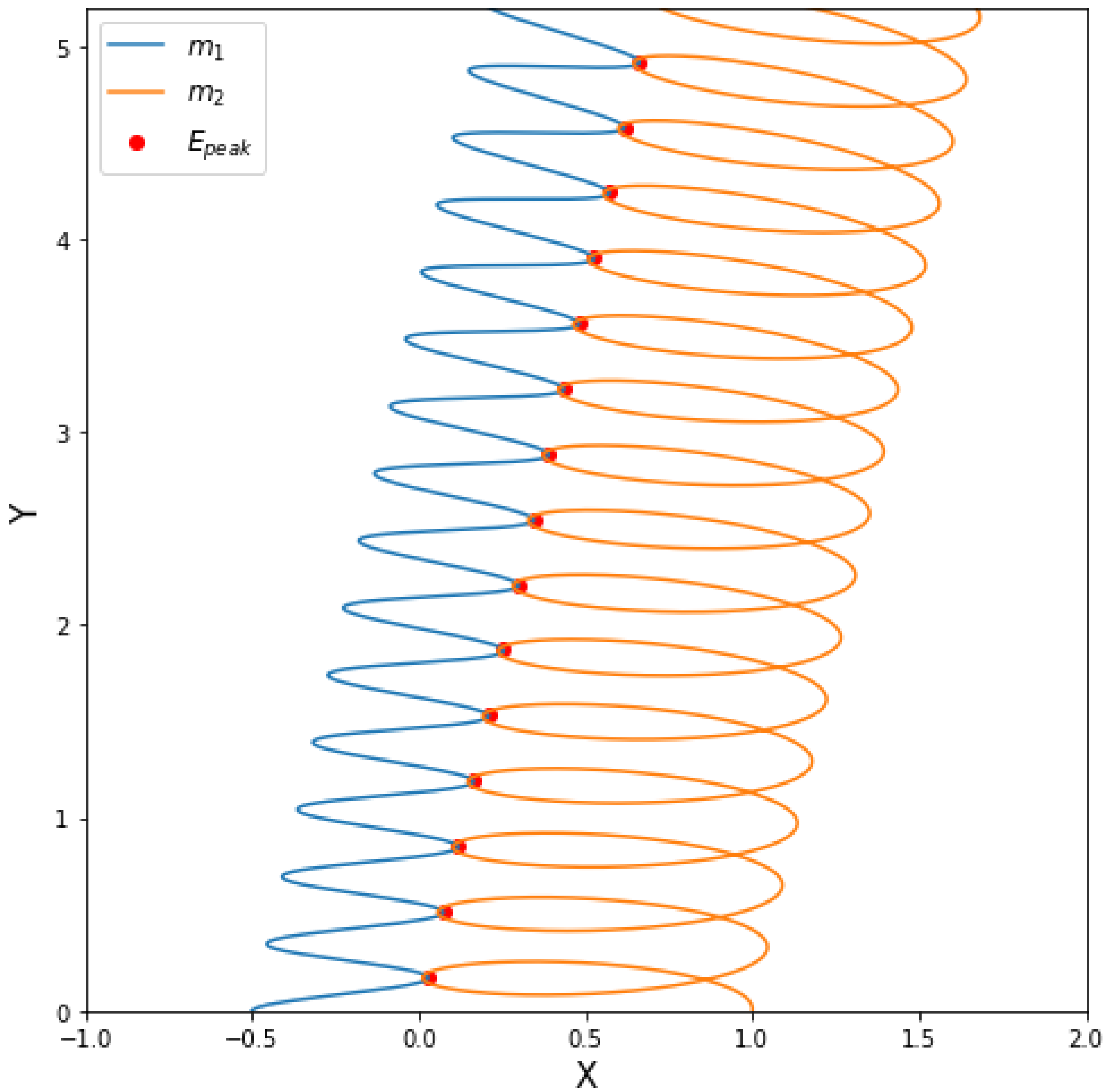
[50]:
```python
xhist = np.loadtxt('xhist.txt')
yhist = np.loadtxt('yhist.txt')
vxhist = np.loadtxt('vxhist.txt')
vyhist = np.loadtxt('vyhist.txt')

E_k1 = 0.5*m[0]*(vxhist[:,0]**2.+vyhist[:,0]**2.)
E_k2 = 0.5*m[1]*(vxhist[:,1]**2.+vyhist[:,1]**2.)
E_p  = -m[0]*m[1]/np.sqrt((xhist[:,0]-xhist[:,1])**2.+(yhist[:,0]-yhist[:,1])**2.
 ↪)
E_tot = (E_k1+E_k2)+E_p
dE_tot = np.abs(1.-E_tot[0]/E_tot[1:])

f=plt.figure(figsize=(7,6))
```

```
ax1=f.add_subplot(211)
ax1.plot(np.arange(dt,tmax+2.*dt,dt)[::10],E_tot[::10],color='k',label=r'$E(t)$')
ax1.axhline(E_tot[0],c='k',ls=':',label=r'$E_{init}$')
ax1.set_xlim(0,tmax)
# ax1.set_ylim(-0.3225,-0.3215)
ax1.set_ylabel(r'$E$',fontsize=15)
ax1.set_xticklabels([])
plt.legend(fontsize=13)

ax2=f.add_subplot(212)
ax2.semilogy(np.arange(dt,tmax+2.*dt,dt)[1:][::10],dE_tot[::10],color='r')
ax2.set_xlim(0,tmax)
ax2.set_ylim(1.0e-10,1.0e-0)
ax2.set_xlabel(r'$t$',fontsize=15)
ax2.set_ylabel(r'$\Delta E/E_{init}$',fontsize=15)

plt.subplots_adjust(hspace=0.)
# plt.tight_layout()
```



시간에 따른 에너지의 변화를 살펴보면 에너지가 급격하게 변화하는 부분이 있다는 사실을 알 수 있다. 이 지점들을 직접 살펴보면 두 물체가 서로 매우 가까워지는 지점에 해당한다. 결국 택한 timestep 의 크기가 적절하지 않았기에 오차가 생겼음을 알 수 있다.

[49]:
```
from scipy.signal import argrelextrema

Epeak=argrelextrema(E_tot,np.greater)[0]

plt.figure(figsize=(7,7))
plt.plot(xhist[:,0],yhist[:,0],label='$m_1$')
plt.plot(xhist[:,1],yhist[:,1],label='$m_2$')
plt.scatter(xhist[Epeak[1:],0],yhist[Epeak[1:
  ↪],0],color='red',label=r'$E_{peak}$')
plt.xlim(-1,2.0)
plt.ylim(0.,5.2)
plt.xlabel('X',fontsize=15)
plt.ylabel('Y',fontsize=15)
plt.legend(loc=2,fontsize=12)
plt.tight_layout()
```




하지만 이럼에도 불구하고 이를 제외한 지점들에서는 에너지의 보존이 잘 일어나고 있음을 확인할 수 있다.

**4.**

[51]:
```
def eigen(y,x,w):
    dy = y[1]
    d2y = (np.abs(x)-w)*y[0]
    return np.array([dy,d2y])
```

[52]:
```
tol = 1.0e-6

xs, xb, xe = -5.,0., 5.
ys, dys, yb=0., 0.1, 0.
```

```
x_domain = np.linspace(xs, xb, 10000)

wplist = []
i,di=0.,2

while len(wplist) < 5:
    w1, w2 = i, i+di # two trial values for eigenvalue
    y1, y2 = ys, yb
    yini = np.array([ys,dys])
    err = 100.
    loop = -1

    while np.abs(err) > tol and loop<1000:
        loop += 1
        if loop == 0:
            wp = w1
        elif loop == 1:
            wp = w2
        y_result = odeint(eigen, yini, x_domain, args=(wp,))
        y1 = y2
        y2 = y_result[-1,0]
        if loop>0:
            err = yb - y2
            wp = w2 + (w2-w1)/(y2-y1) * err
            w1 = w2
            w2 = wp
    if np.abs(err) < tol:
        if len(wplist)==0:
            print (int(i),wp)
            wplist.append(wp)
        elif np.min(np.abs(wplist-wp))>0.1:
            print (int(i),wp)
            wplist = np.append(wplist,wp)
    wplist = np.sort(wplist)
    i+=di
```

```
0 2.339049297545528
2 4.162125085976274
4 6.166596089699012
8 8.890796375171442
10 12.419212316673699
```

[53]:
```
x_domain = np.linspace(xs, xe, 10000)
yini = np.array([ys,dys])
ylist,dylist = np.zeros((5,len(x_domain))),np.zeros((5,len(x_domain)))

for i in range(5):
```

```
    yini = np.array([ys,dys])
    y_result = odeint(eigen,yini,x_domain,args=(wplist[i],))
    ylist[i] = y_result[:,0].T
    dylist[i] = y_result[:,1].T
```

[54]:
```
f = plt.figure(figsize=(12,6))
ax1 = f.add_subplot(121)
ax2 = f.add_subplot(122)
for i in range(5):
    ax1.plot(x_domain,ylist[i],label=r'$\lambda=%.3f$' % (wplist[i]) )
    ax2.plot(x_domain,dylist[i])
ax1.axvline(0.0,c='k',ls=':')
ax1.axhline(0.0,c='k',ls=':')
ax1.set_xlim(-5.,5.)
# ax1.set_ylim(-1.5,1.5)
ax1.set_xlabel('$x$',fontsize=15)
ax1.set_ylabel('$y$',fontsize=15)
ax1.set_xticks(np.arange(-5.,5.1,1.))
ax1.set_yticks(np.arange(-1.5,1.51,0.1),minor=True)
ax1.legend(fontsize=12)

ax2.axhline(0.1,c='k',ls=':')
ax2.set_xlim(-5.,5.)
ax2.set_ylim(-2.,1.)
ax2.set_xlabel('$x$',fontsize=15)
ax2.set_ylabel("$y'$",fontsize=15)
ax2.set_xticks(np.arange(-5.,5.1,1.))
ax2.set_yticks(np.arange(-2.0,1.01,0.1),minor=True)

plt.tight_layout()
```