Oblig 2: oppg. 4.4
AST1100 – Høsten 2011

Aleksander Hansen

Oppg. 4.4

Introduksjon

Vi skal finne bedre estimater på massene til planetene som går i bane rundt stjernene i datasettet vårt, enn det vi klarer ved øyemål. Massen til en planet som går i bane er gitt ved $m_p \sin i = \frac{m_s^{2/3} v_{sr} P^{1/3}}{(2\pi G)^{1/3}}$ (1), hvor m_p er planetens masse, m_s stjernens masse, v_{sr} er stjernens maksimale radielle hastighet sett fra massesenteret, P er perioden til omløpet om massesenteret, G er Newtons konstant, og i er inklinasjonsvinkelen mellom synslinjen og en linje som står normalt på planet som «utspennes» av banen til stjernen og planeten. m_s har vi fått oppgitt for stjernene. i vet man som regel ikke, men hvis vi velger stjerner der vi ser en «dip» i relativ flux indikerer det at planeten har skygget for stjernen mens den har beveget seg foran synslinjen og vi vet dermed at i~90° og (1) er ikke da bare en laveste grense, men et godt estimat på massen til planeten. P og v_{sr} er det vi har målt på øyemål til nå, vi skal istedet bruke minste kvadraters metode for å finne bedre verdier på disse.

Vi modellerer hastighetskurvene som cosinus kurver, $v_r^{model}(t) = v_{sr} \cos(\frac{2\pi}{P}(t - t_0))$ (2) hvor (2) er den teoretiske modellen av den radiale hastigheten relativt til massesenteret, og t_0 er tidspunktet hvor hastigheten er størst. Vi regner så ut en funksjon $\Delta(t_{0,} P, v_{sr}) = \sum_{t=t_0}^{t=t_0+P} (v_r^{data}(t) - v_r^{model}(t, t_{0,} P, v_{sr}))^2$ (3), for forskjellige verdier av t_0 , P og v_{sr} og vi vet dermed at for de verdiene som gjør (3) minst så passer modellen vår (2) best med dataene. Dermed har vi funnet verdier for P og v_{sr} som vi kan bruke til å estimere massen til planeten forhåpentligvis mer nøyaktig.

Siden mesteparten av koden hovedsaklig er beregnet for å løse oppg. 4.1-4.3, og man trenger noen av opplysningene man får ved å løse de i oppg. 4.4 så går jeg raskt gjennom denne delen av koden også. Hovedsaklig er det metoden *least_square* som er relevant for denne oppgaven.

Når man lager et objekt av klassen *Star* leser først metoden *read_file* dataene (tid, bølgelengden til Hα-spektrallinjen og relativ flux) fra fila og legger de i egne arrayer. Deretter regner metoden *velocities* ut den radiale hastigheten sett fra jorda ved å bruke dopplerformelen $v_r = \frac{\lambda - \lambda_0}{\lambda_0} c$ på bølgelengdedataene, hvor λ er bølgelengden som observert fra jorda, λ_0 er bølgelengden som observert fra kilden og c er lysets hastighet. Så regner den ut hastigheten til massesenteret ved å ta gjennomsnittet av den radiale hastigheten sett fra jorda. Og så regner *velocities* den radiale hastigheten sett fra massesenteret. Deretter plotter metoden *velocity_curve* og *light_curve* henholdsvis hastigheten mot tid og relativ flux mot tid. *least_square* tar som input en øvre og nedre grense for t_0 hvor innenfor grensen ligger faktisk t_0 , samme for v_{sr} og P , samt en verdi N som er antall verdier mellom den øvre og nedre grensen vi tester. Ideen er at vi skal prøve disse forskjellige verdier av t_0 , v_{sr} og P i modellen vår (2), for så å se hvilke som passer best til dataene ved å bruke (3). Derfor prøver vi alle kombinasjonene av t_0 , v_{sr} og P for så å tilslutt bare ta vare på de verdiene som gjør (3) minst.

Python kode

```
from scitools.all import *

class Star:

    # Constructor
    def __init__(self, filename):
        # Self-variables
        self.filename = filename
        self.time = []
        self.lambda1 = []
        self.flux = []
        self.rad_vel = []   # Radial velocity as measured from Earth
        self.pec_vel = []   # Peculiar velocity (velocity of center of mass)
        self.rel_vel = []   # Velocity relative to the center of mass

        # Calling functions
        self.read_file()
        self.velocities()
        self.velocity_curve()
        self.light_curve()


    # Reads data from file and makes arrays with the corresponding data
    def read_file(self):
        infile = open(self.filename, 'r')
        for line in infile:
            data = line.split()
            self.time.append(float(data[0]))
            self.lambda1.append(float(data[1]))
            self.flux.append(float(data[2]))
        self.time = array(self.time)
        self.lambda1 = array(self.lambda1)
        self.flux = array(self.flux)
        infile.close()


    # Calculation of radial, peculiar and relative velocites from wavelenght data
    def velocities(self):
        lambda0 = 656.3    # Wavelenght of the H_alpha spectral line seen from the restframe of
the source [nm]
        c = 299792458.     # Speed of light in vacuum [m/s]
        self.rad_vel = array([((self.lambda1[i] - lambda0)/lambda0)*c for i in
range(len(self.lambda1))])
        self.pec_vel = sum(self.rad_vel)/len(self.rad_vel)
        self.rel_vel = self.rad_vel - self.pec_vel

    # Plotting relative velocity vs. time
```

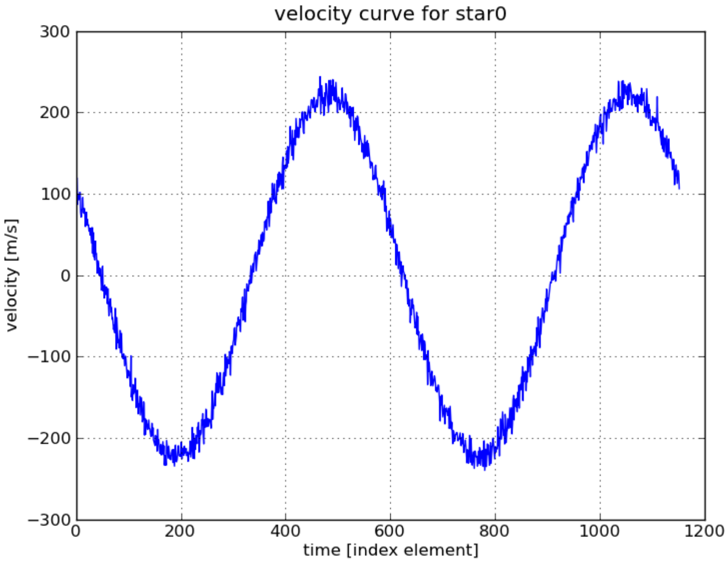
```
def velocity_curve(self):
    # rel_vel vs. time
    plot(self.time, self.rel_vel)
    xlabel('time [sec]')
    ylabel('velocity [m/s]')
    name = self.filename.split('.')
    title('velocity curve for %s' % name[0])
    hardcopy('velocity_curve_%s.png' % name[0])

    # rel_vel vs. time index
    index = array([i for i in range(len(self.time))])
    plot(index, self.rel_vel)
    xlabel('time [index element]')
    ylabel('velocity [m/s]')
    title('velocity curve for %s' % name[0])
    grid(True)
    hardcopy('velocity_curve_%s_index.png' % name[0])


# Plotting relative flux vs. time
def light_curve(self):
    plot(self.time, self.flux)
    xlabel('time [sec]')
    ylabel('relative flux')
    name = self.filename.split('.')
    title('light curve for %s' % name[0])
    hardcopy('light_curve_%s.png' % name[0])


# Finding better values for t0, vr and P based on the least square method
def least_square(self,t0_min,t0_max,v_min,v_max,P_min,P_max,N):
    t0 = linspace(self.time[t0_min],self.time[t0_max],N)
    vr = linspace(v_min,v_max,N)
    P = linspace(self.time[P_min],self.time[P_max],N)

    best_t0 = t0[0]
    best_vr = vr[0]
    best_P = P[0]
    best_delta = sum(self.rel_vel**2)   # Set to worst case senario

    for i in range(N):
        for j in range(N):
            for k in range(N):
                rel_vel_model = array([vr[j]*cos(2*pi/P[k]*(self.time - t0[i]))])
                delta = sum((self.rel_vel - rel_vel_model)**2)
                if delta < best_delta:
                    best_t0 = t0[i]
                    best_vr = vr[j]
                    best_P = P[k]
```

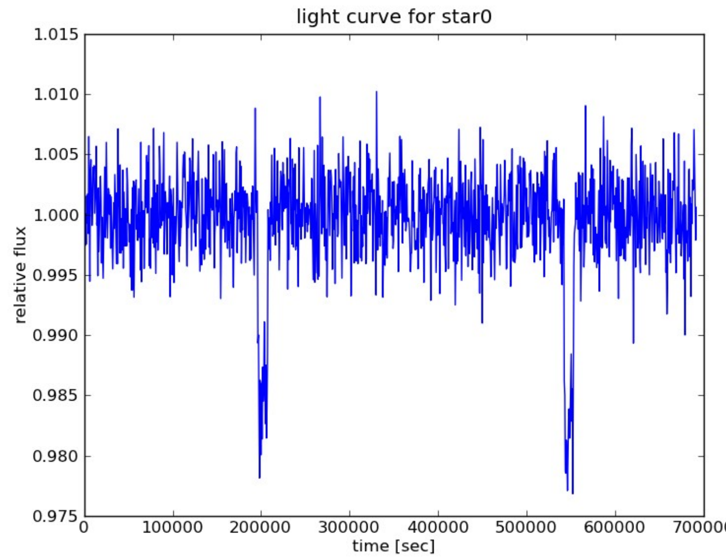
```
                    best_delta = delta

        name = self.filename.split('.')
        print '%s:' % name[0]
        print 'best_t0:%d' % best_t0
        print 'best_vr:%d' % best_vr
        print 'best_P:%d' % best_P


# Main
star = [Star('star%s.txt' % i) for i in range(10)]

star[0].least_square(450,550,200,250,550,650,20)
star[3].least_square(200,250,700,800,280,320,20)
star[4].least_square(700,900,280,380,1700,1900,20)
```

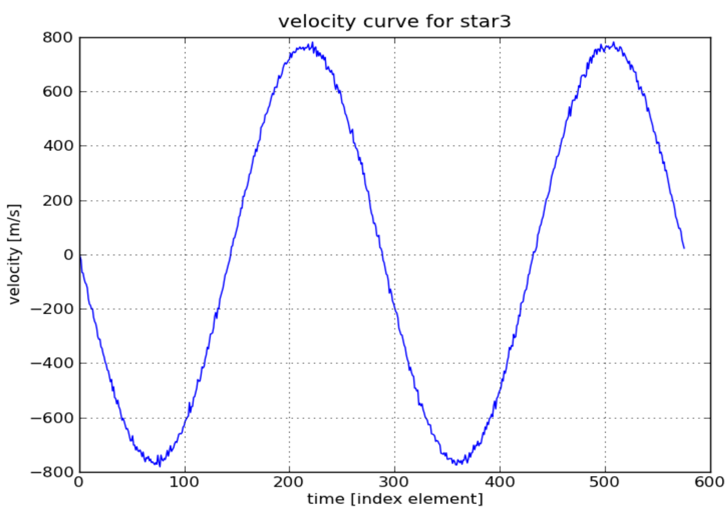
Resultat

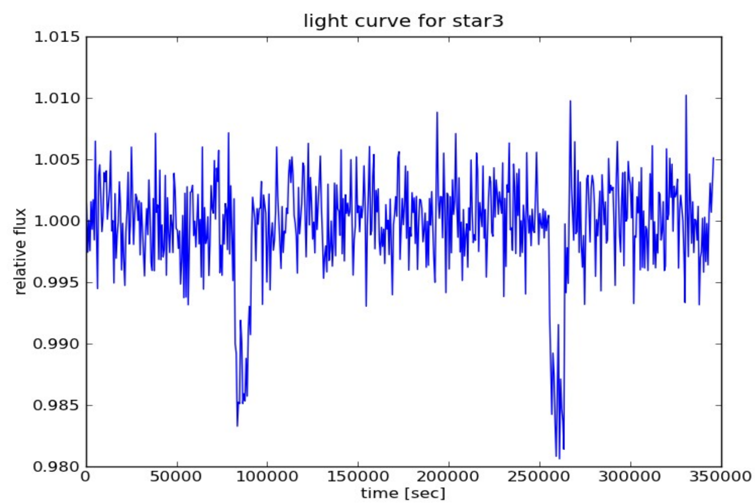
Jeg valgte å presentere resultatet for stjernene 0, 3 og 4 fordi inklinasjonsvinkelen for de er tilnærmet 90° fordi planeten formørker stjerna. Vi bruker (1) til å regne ut massen til planetene basert på verdier fra både øyemål og minste kvadraters metode. Resultatet er summert i tabellen under.

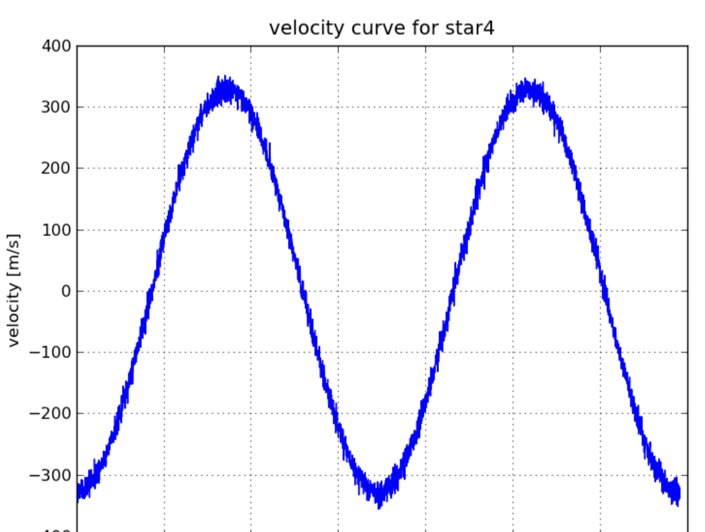
Stjerne #	0	3	4	
m_s [solmasser]	0.8	0.5	1.8	
v_{sr} [m/s]	230	750	320	
P [s]	$3.5*10^5$	$1.75*10^5$	$1.1*10^6$	Øyemål
m_p [kg]	$2.95*10^{27}$	$5.58*10^{27}$	$1.03*10^{28}$	
v_{sr} [m/s]	221	763	327	
P [s]	346089	173353	1032930	Least square fit
m_p [kg]	$2.83*10^{27}$	$5.66*10^{27}$	$1.03*10^{28}$	
$\frac{m_p[øyemål]}{m_p[lsf]}$	~1.04	~0.98	~1.00	

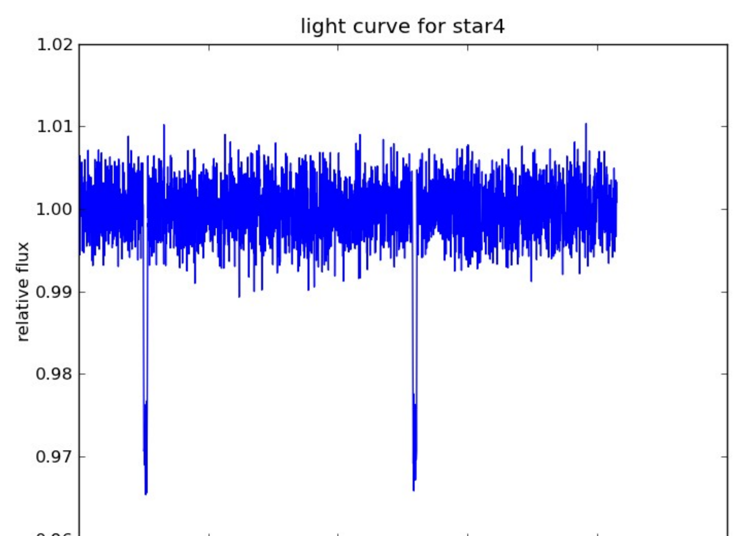

velocity curve for star0
velocity [m/s]
300
200
100
0
−100
−200
−300
time [index element]
0
200
400
600
800
1000
1200


light curve for star0
relative flux
1.015
1.010
1.005
1.000
0.995
0.990
0.985
0.980
0.975
time [sec]
0
100000
200000
300000
400000
500000
600000
700000


velocity curve for star3
velocity [m/s]
800
600
400
200
0
−200
−400
−600
−800
time [index element]
0
100
200
300
400
500
600


light curve for star3
relative flux
1.015
1.010
1.005
1.000
0.995
0.990
0.985
0.980
time [sec]
0
50000
100000
150000
200000
250000
300000
350000


velocity curve for star4
velocity [m/s]
400
300
200
100
0
−100
−200
−300
−400
time [index element]
0
500
1000
1500
2000
2500
3000
3500


light curve for star4
relative flux
1.02
1.01
1.00
0.99
0.98
0.97
0.96
time [sec]
0
500000
1000000
1500000
2000000
2500000