AST1100 - 1D.6

Andreas Helland

14. november 2016

1 Introduksjon

Vi ser på observasjoner av en stjerne rundt bølgelengden $\lambda_0 = 656.3nm$ hvor vi forventer å finne en absorbasjonslinje. Det vil si vi forventer å finne en nedgang i fluksen ettersom fotoner med gitt bølgelengde blir absorbert og færre av dem rekker oss som observerer stjernen. Ved å se på forskjellen mellom der absorbasjonslinjen befinner seg og bølgelengden vi forventet å finne den, kan vi finne den radielle hastigheten til stjernen i forhold til oss fra doppler effekten.[1]

2 Metode & Fremgangsmåte

Vi har blitt gitt 10 tekstfiler fylt med observasjonsdata (med 88850, blir filene fra mappe 50 brukt). Disse filene inneholder to set med verdier. Fluks og bølgelengde. Vi begynner med å sortere dem i egne lister, slik at vi har en liste med bølgelengder, og en tilsvarende liste som gir oss hvor stor fluksen ved disse bølgelengdene er. Denne fluksen skal så plottes som en funksjon av bølgelengden. Disse to problemene løses i egne funksjoner.

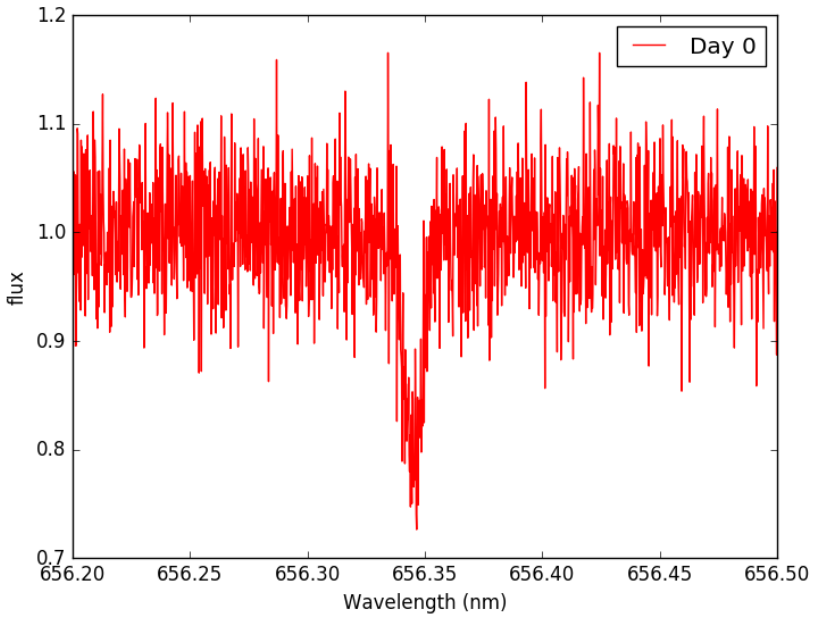
Programmet leter etter tekstfiler med forventet navn i samme mappen. Den sjekker 'spectrum_day' for hver dag og teller så høyt den klarer. Dersom den ikke har funnet en fil med dagen den har kommet til i filnavnet på over 2 uker slutter den å lese etter flere.[2] Når den finner en fil, leses verdiene inn i lister som sendes til plotte-funksjonen. Alle plot er lagret slik at vi kan gjøre øye-estimat av forflyttelsen til spektrallinjen og finne radiellhastigheten ved hjelp av dopplerligninga.

$$v_r = \frac{\Delta\lambda}{\lambda_0}c \tag{1}$$

Observasjonsverdiene vi har fått oppgitt inneholder nokså mye støy og store unøyaktigheter som gjør det vanskelig å lese av forflyttelsen av spektrallinjen med god presisjon. Vi skal derfor utvide programmet for å finne et mer nøyaktig estimat. Vi bruker minste kvadraters metode for å lage oss en mer presis modell av fluksen. Vi velger data fra dag 0 (se figur 1).

[1] Denne obligen ble fullført før 1C.4 i håp om å bli ferdig før den valgfrie inneveringsfristen for tilbakemelding. den er dermed, etter min mening, programmert dårligere og litt rushet. Som de fleste inneveringene skulle alt helst blitt gjort på nytt med et bedre utgangspunkt.

[2] Dette er bare en arbitrær måte å avslutte programmet på etter det har lest alle filene.



Figur 1: Fluksen som funksjon av bølgelengden den første dagen med observasjon (day 0).

I model funksjonen (se figur 1) definerer vi først verdiene vi trenger for å gjøre disse estimatene. Ut ifra øyeobservasjoner defineres et intervall i bølgelengden hvor spektrallinjen befinner seg mellom (λ_{center}), standardavvik (σ) og fluksen i bunnpunktet på spektrallinjen. Estimatene av disse verdiene ble først gjort ved øye-estimat, for så å bli endret her og der for å forbedre modellen.

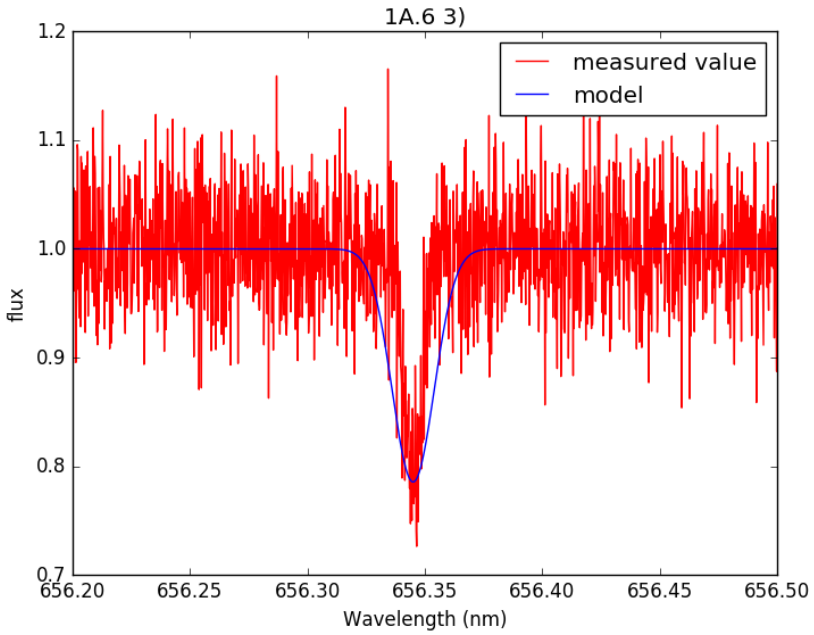
Med disse verdiene finner vi dermed den minste mulige $\Delta(F_{min}, \sigma, \lambda_{center})$ ved å sjekke alle kombinasjonene av $n = 25$ verdier i de innskrevne intervallene, finner forskjellen fra det observerte (ligning 2) og lagrer Δ som gir oss minst avvik.[3]

$$\sum_{\lambda} \quad = F^{obs}(\lambda) - F^{model}(\lambda, F_{min}, \sigma, \lambda_{center}))^2 \tag{2}$$

Modellen opprettes dermed med verdiene vi har funnet og den gaussiske funksjonen fra oppgavesettet (3) og plottes sammen med observasjonsdataen (se figur 2)

[3]Intervallene F_{min}, σ, og λ_{center} er hardkodet og avhenger ikke av input fra bruker for å spare tid ved testing av nye verdier. Dette kan lett endres hvis det ønskes.

$$F^{model}(\lambda) \quad = F_{max} + (F_{min} - F_{max})e^{-(\lambda-\lambda_{center})^2/(2\sigma^2)} \tag{3}$$



Figur 2: Fluksen som funksjon av bølgelengden den første dagen med observasjon (day 0) sammen med en model opprettet med minste kvadraters metode.

3 Resultater

Plot fra dag 3, dag 9 (4), dag 12 (5) er vedlagt, og fra dem kan vi se at spektrallinjen forflytter seg over tid. Følgende $\Delta\lambda$ i tabell 1 har blitt funnet ved øye-estimat. Fra dopplerformelen (1).

I den mer nøyaktige modellen vår får vi $v_r = 20.55\frac{km}{s}$ med $f_{min} = 0.785625$, $\sigma = 0.008958$ og $\lambda_{center} = 656.345$

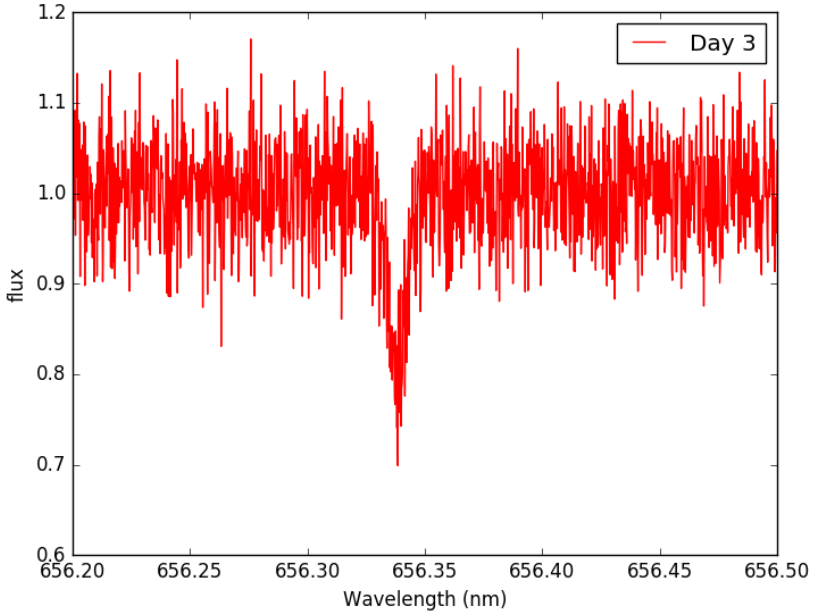
Dag	$\Delta\lambda$	v_r
Day 0	$0.046nm$	$21.01\frac{km}{s}$
Day 1	$0.042nm$	$19.64\frac{km}{s}$
Day 3	$0.040nm$	$18.27\frac{km}{s}$
Day 4	$0.041nm$	$18.73\frac{km}{s}$
Day 6	$0.043nm$	$19.64\frac{km}{s}$
Day 7	$0.044nm$	$20.1\frac{km}{s}$
Day 9	$0.045nm$	$20.55\frac{km}{s}$
Day 10	$0.043nm$	$19.64\frac{km}{s}$
Day 12	$0.041nm$	$18.73\frac{km}{s}$
Day 13	$0.040nm$	$18.27\frac{km}{s}$

Tabell 1: Øye-estimat av $\Delta\lambda$ fra observasjonene gitt og den radielle hastigheten til stjernen ut ifra estimert verdi.

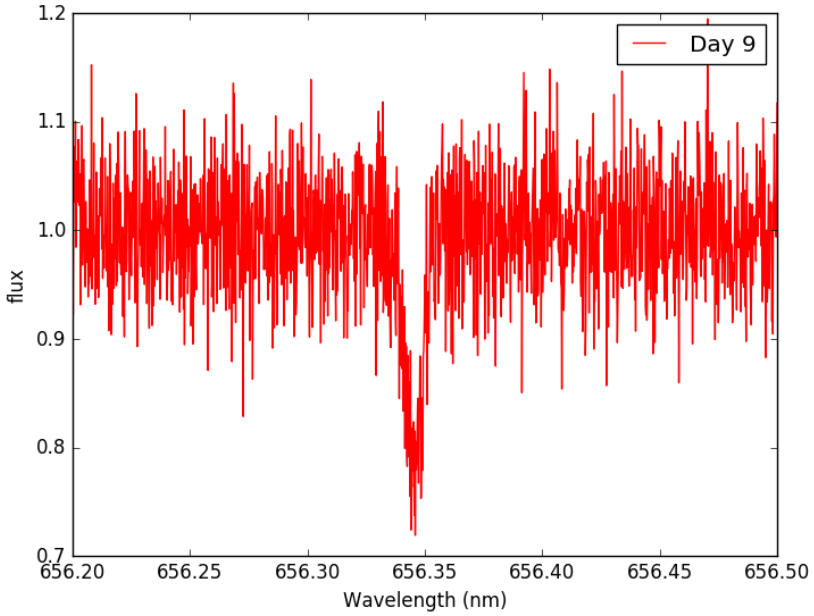
4 Diskusjon og konklusjon

Minste kvadraters metode ser ut til å være en god nok måte å lage en modell for fluksen fra stjerner. Hvis man vil ha et godt estimat gjennom støyet og få en mer nøyaktig radiell hastighet for stjernen i forhold til deg (observatøren).

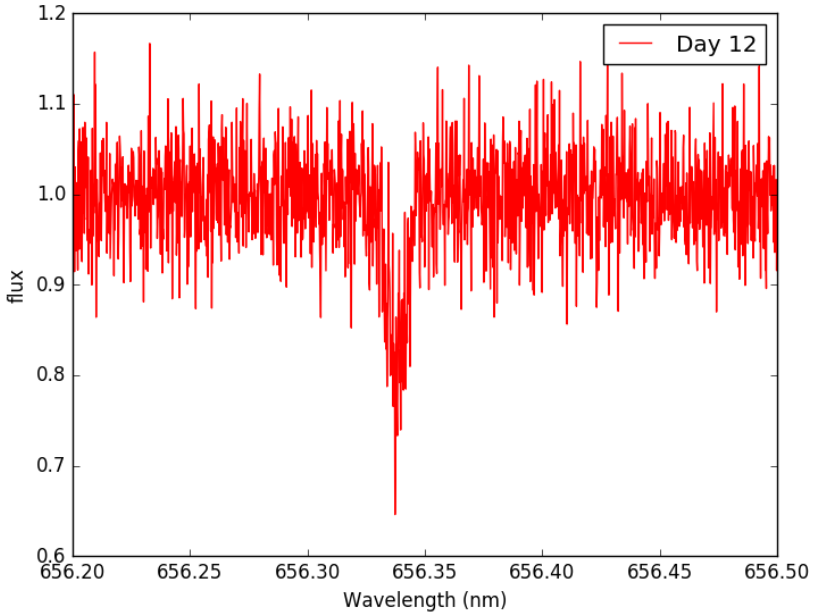
A Vedlegg



Figur 3: Fluksen som funksjon av bølgelengden fra dag 3 med observasjon.



Figur 4: Fluksen som funksjon av bølgelengden fra dag 9 med observasjon.



Figur 5: Fluksen som funksjon av bølgelengden fra dag 12 med observasjon.

```
def plotter(filename, flux, wavelength):
    name = filename.strip('.txt').strip('spectrum_day')
    fig = plt.figure()
    ax = fig.add_subplot(111)
    plt.plot(wavelength, flux, c='r' )
    plt.legend(['Day %s' % name])
    plt.title('1A.6 1)')
    plt.xlabel('Wavelength (nm)')
    x_formatter = matplotlib.ticker.ScalarFormatter(useOffset=False)
    ax.xaxis.set_major_formatter(x_formatter)
    plt.ylabel('flux')
    plt.show()
    figureName = 'Day'
    figureName +=str(name)
    figureName +='plot.png'
    fig.savefig(figureName)

# read data from file
def read(filename):
    i = 0
    file = open(filename, 'r')
    for data in [x.split() for x in file]:
        # separate flux & wavelength into lists
        flux.append(float(data[1]))
        wavelength.append(float(data[0]))
        i += 1

    return flux, wavelength

f = 0           #days gone without a new observation data
day = 0         #current day
while f <= 14:  #as long as there's been data the past f days, contin
    flux = []
    wavelength = []
    filename = 'spectrum_day'
    filename +=str(day)
    filename +='.txt'

    if os.path.isfile(filename): #if data file exist
        flux, wavelength = read(filename)   #read data file
        plotter(filename, flux, wavelength) #plot data

        f = 0   #reset day count
    else:
        f +=1   #count day without data
    day +=1
```

Figur 6: Plot som viser satellitbanene inn mot planeten med lange tidsintervall $dt = 0.15s$. $v_{0,y} = -2837\frac{m}{s}$, $v_{0,x} = -810\frac{m}{s}$ $A = 2.5m^2$.

```
def model(flux, wavelength, index):
    model, fmax = 0, 1.0

    #Rough, by-eye estimates, followed by fine tuning through small changes
    n = 25  #number of values
    fmin = np.linspace(0.86, 0.775, n)
    sigma = np.linspace(0.005, 0.1, n)
    lambdacenter = np.linspace(656.343, 656.347, n)

    delta = [0,0,0]
    best_delta = 10000000000
    observed = (sum(flux))/(len(flux)) #F_obs

    # Calculate minimal delta by finding the ideal combination of fmin,sigma and lambdacenter
    for i in fmin:
        for j in sigma:
            for k in lambdacenter:
                for l in wavelength[500:800]: #only check certain interval to save time

                    # Calculate model
                    testModel = fmax + (i - fmax)*(exp(-(l - k)**2.0/(2.0*(j**2.0))))

                    # Calculate change
                    deltaCurrent = (observed - testModel)**2

                    # if current delta is smaller than previous smallest delta
                    if deltaCurrent < best_delta:
                        best_delta = deltaCurrent #set currentDelta as best estimate
                        delta[0] = i
                        delta[1] = j
                        delta[2] = k
    c, f0 = 299792.458, 656.3
    print 'fmin = ',delta[0],', sigma = ',delta[1],' lambdacenter = ',delta[2],'\n'
    print 'relative velocity = ',((delta[2]-f0)/f0)*c,'km/s '

    # Create model array
    model = np.zeros(len(index))
    for i in index:
        model[i] = fmax + (delta[0] - fmax)*(exp(-((wavelength[i] - delta[2])**2.0)/(2.0*(delta[1]**2.0))))

    return model
```

Figur 7: funksjonskoden for å finne modellen gjennom minste kvadraters metode.