# *Sterrenstelsels* Computer Project: Colour-magnitude diagrams and star formation histories

Søren S. Larsen

November 11, 2019

# Introduction

This is the computer project for the course *Sterrenstelses* ("Galaxies"), a second-year course given at the Radboud University as part of the astrophysics Bachelor's curriculum. The original version of these notes was written by R. Scheepmaker, E. Helder and S. Larsen for a Master's course on Galaxies taught at Utrecht University during the years 2007–2011. The current version has been modified for this course to fit within a 3 ECT course load, and for use of the `Python` programming language (instead of `IDL`).

At the end of the project, it is mandatory to hand in a short report ($\leq$ 5 pages) which will be graded and forms part of your final grade for the course. You are allowed to work together with other students on the project, but each student *must* hand in a separate report, written by yourself. It is not allowed to copy the report (or parts of it) from others.

## Python

For these exercises you will use the `Python` programming language. `Python` is rapidly gaining popularity for astronomical data analysis and a large number of specialized libraries are already available. An attractive feature of `Python` is that it can be used *interactively*, i.e. you can type commands directly at the prompt and immediately see what happens before using the commands in your programmes. It is easy to make nice-looking plots and save them in a format that can be imported into documents.

`Python` has a built-in help system that provides information on how to call the various functions. This can be accessed from the `Python` prompt as in the following example:

```
>>> help(numpy.loadtxt)
```

`Python` will then display the help associated with the `numpy.loadtxt` function. A large amount of documentation is also available on-line, see e.g.
`http://docs.python.org/2/tutorial/index.html` for a general tutorial and
`http://matplotlib.org/users/pyplot_tutorial.html` for a tutorial describing the `matplotlib` plotting package. A good introduction to `Python` can also be found

on the Dutch Wikipedia page,
`http://nl.wikipedia.org/wiki/Python_(programmeertaal)`

For the purpose of this exercise you can use the standard `Python` installation on the workstations in the computer rooms of the FNWI. You will need to boot the workstation in Linux mode. Then `Python` can be called simply by typing `python` at the Unix prompt in a terminal window.

# Colour-Magnitude Diagram

In this exercise you are going to interpret observations of star clusters and dwarf galaxies by comparing them with theoretical stellar model calculations. The aim is to show how ages, distances and other properties can be derived from observational data.

You are going to simulate a Colour-Magnitude Diagram (CMD) using Padova isochrones and a Salpeter Initial Mass Function (IMF). Padova isochrones and stellar tracks can be found at `http://pleiadi.pd.astro.it/`, but you can also find all the necessary files for this exercise via links at
`http://www.astro.ru.nl/~slarsen/teaching/Galaxies/`

## 1.1 Tracks and Isochrones

The output of stellar model calculations is typically a number of *evolutionary tracks* which tabulate the evolution of various properties (effective temperature, luminosity, etc.) for stars of a given set of (initial) masses as a function of time. Such tracks are computed by starting out with a given chemical composition and following the star through its various stages of evolution as it uses up its nuclear fuel. However, for practical applications it is often more useful to interpolate in these tracks to get the stellar properties as a function of *mass* for a given *age*. When presented in this form we talk about stellar *isochrones*, from Greek *isos* (=same, equal) and *chronos* (=time).

An isochrone can be thought of as a snapshot of the properties of a population of coeval stars. It is important to realise that the isochrone itself does not contain information about the relative numbers of stars of different masses that are present in the population, but only tabulates the properties of stars as a function of mass.

The fundamental output from stellar model calculations is the effective temperature, luminosity and surface gravity of the stars. These properties are, however, not directly comparable to observations and most model calculations also include observables such as broad-band colours and absolute magnitudes (e.g. $M_V$, $B-V$, $U-B$, etc.). These are obtained by combining the stellar models with stellar *model atmospheres* that provide the detailed distribution of emergent flux as a function of wavelength. This can then be integrated over the transmission curves of any photometric filter. Alternatively, one might attempt to convert the observables back to the theoretical plane, but this greatly

complicates the error analysis since such transformations are often quite non-linear. In this exercise, as in most analyses, we will work in the observational plane and compare broad-band colours with those tabulated in the isochrone tables.

## 1.2 Padova isochrones

The isochrones we will use are computed by a group at the University of Padova. They assume a metallicity between $Z = 0.0001$ and $Z = 0.030$ (where the Sun has $Z_\odot = 0.019$) and present absolute magnitudes in the Johnson-Cousins filter system (UBVRIJHK). One advantage of the Padova isochrones is that they are available for a wide range of ages and metallicities.

1.1 Copy the isochrone files `isoc_z0???.dat` to your working directory and have a look at one with your favourite text editor. You can find the files at the link: `http://www.astro.ru.nl/~slarsen/teaching/Galaxies/data`
What is listed in the various columns? What stepsize in log(age) do they use? Why is there a difference between actual and initial masses? What is the range in initial masses? What, according to you, is the reason for not using equal mass intervals?

## 1.3 Plotting with Python

Let us now have a look at the isochrones in a Hertzsprung-Russell diagram, i.e. a diagram of luminosity versus effective temperature. Below is a small `Python` programme that will produce a plot of two isochrones, label the $x$- and $y$-axes and add a legend to the figure.

---

```
import numpy
import matplotlib.pyplot as plt

loga, logl, logte = numpy.loadtxt('isoc_z008.dat',
  usecols=(0, 3, 4), unpack=True)

w7 = numpy.where(loga == 7.0)
w8 = numpy.where(loga == 8.0)

plt.plot(logte[w7], logl[w7],label='10 Myr')
plt.plot(logte[w8], logl[w8],label='100 Myr')
```

```
plt.xlabel('Log(Teff)')
plt.ylabel('Log L')
plt.axis([5.0, 3.5, 0, 6])

plt.legend(loc='lower left')

plt.show()
```

---

Let us take a look at this small piece of code, step by step, and try to understand what happens:

We first load two *libraries*, `numpy` and `matplotlib.pyplot`. `numpy` contains functions for manipulating data arrays, while `matplotlib.pyplot` contains functions for making plots. Note that we load the two with slightly different syntaxes:

```
import numpy
```

means that we can from now on simply use functions in the `numpy` library by specifying the name of the library, followed by the relevant function, as in `numpy.loadtxt()`. We could have loaded the `matplotlib.pyplot` library in the same way, but by using the syntax

```
import matplotlib.pyplot as plt
```

we specify that we will refer to this library by the shorter name `plt`, rather than the full name (as in `plt.plot`, instead of `matplotlib.pyplot.plot`).

Next, we use the `loadtxt` function in the `numpy` library to read columns from the isochrone data file. The `usecols` option allows us to specify which columns to read (*note: starting with 0*). Looking at one of the isochrone files (for example `isoc_z008.dat`), we see that the logarithm of the age is in the first column, and the luminosity and effective temperature are in the 4th and 5th columns. We therefore specify `usecols=(0,3,4)`.

Note that, unlike many other programming languages, `Python` functions can return multiple variables. In this case, `loadtxt()` returns as many variables as the number of columns specified by the `usecols` parameter. Each variable is, in this case, itself an *array* of numbers.

The isochrone file contains isochrones for a large number of ages. To pick out the entries with a given age (that we read in the first column) we use the `numpy.where()` command:

```
w7 = numpy.where(loga == 7.0)
```

The array `w7` now contains the indices of the array `loga` that have the value 7.0. We repeat this for log(age)=8.0. Note that we can use variables directly without declaring them – in `Python`, variables are allocated dynamically as needed.

We are now ready to plot the isochrones for the ages we just selected. This is done with the `plot()` command in the `matplotlib.pyplot` library. We call this as

```
plt.plot(logte[w7], logl[w7], label='10 Myr')
```

This produces a plot with `logte` along the *x*-axis and `logl` along the *y*-axis for the chosen age. We have also assigned a *label* to this curve; this will be used later when we add a legend to the figure.

The next commands add labels to the *x*-axis and *y*-axis and change the range on both axes. We plot the *x*-axis with temperature *decreasing* from the left to the right, following the standard convention for H-R diagrams.

```
plt.xlabel('Log(Teff)')
plt.ylabel('Log L')
plt.axis([5.0, 3.5, 0., 6])
```

At the end, we add a legend to the figure:

```
plt.legend(loc='lower left')
```

The `legend` command automatically keeps track of the curves in your plot and associates the proper line colour/style with each label.

Finally, we need to actually display the plot. For some graphics environments, the plot will already have appeared on the screen by now. More generally, however, it is necessary to call the

```
plt.show()
```

command.

If you need to include the plot in a report, you can save the plot with the `savefig()` function:

```
plt.savefig('hrd.pdf')
```

`Python` will try to recognize automatically what format you want to save the figure in. In this case, it will be a PDF file because of the `.pdf` extension to the file name. This command has to be placed *before* the final `plt.show()` command. You can also save the plot from the interactive window that appears on the display.

1.2 Copy all the commands above to a text file, e.g. `plot.py`. Then execute the commands in the file by typing

Figure 1.1: The Pleiades open cluster (left) and the globular cluster 47 Tuc (right).

```
python plot.py
```

at the Unix prompt. Python will bring up a window where you can interactively inspect the plot, zoom in, and save to a file.

1.3 Plot a single Hertzsprung-Russell diagram in which you show the isochrones for log(age) = [7.0, 7.5, 8.0, 8.5, 9.0] for metallicity $Z = 0.008$.

The plots above refer to the *theoretical* plane of effective temperature and luminosity. These are the fundamental physical properties of stars that are computed by stellar models. However, we cannot observe them directly, and observers therefore typically refer to the *observational* plane of colours and magnitudes. In this case, we talk about the *colour-magnitude* diagram, which is the observational counterpart of the H-R diagram.

1.4 Plot $V$ versus $B - V$ for a log(age) = 8.5 isochrone. Remember that numerically smaller magnitudes correspond to brighter stars. What is the colour of the main sequence turnoff for $Z = 0.008$? What is the effect of metallicity on the colour of the turnoff?

## 1.4 47 Tuc and the Pleiades

We will now look at some real data of two star clusters, and compare these to the theoretical Padova isochrones. We have Strömgren photometry from the Danish 1.5 m telescope of the globular cluster 47 Tucanae, and UBV photometry of the "Pleiades" open cluster.

Copy the files `47tuc.dat` and `m45.dat` to your working directory.

We suggest that you make a copy of the `Python` script above and modify it for this part of the exercise.

Read the columns of data into `Python` (for `m45.dat` only read the first 4 columns). Use these approximations to transform the Strömgren photometry to the Johnson-Cousins filter system:

$$V = y \tag{1.1}$$

$$B - V = -0.055 + 1.707(b - y). \tag{1.2}$$

In `Python` it is quite easy to perform this type of operations on data arrays. If X and Y are two `numpy` arrays (such as the output from the `loadtxt` function), you can use the usual operators:

```
>>> XY = X*Y
>>> Z = X + Y
>>> ZZ = 2*X + Y
```

etc..

1.5 Make a $V$ versus $B - V$ diagram of the Pleiades and overplot the isochrone that resembles the CMD best. Use this best-fitting isochrone to estimate the age, distance and metallicity of the Pleiades. You will need to shift the model isochrone in the vertical direction to match the observed CMD.
*Hint:* it may be helpful to leave out the `plt.axis()` command initially. `Python` will then auto-scale the plot, and you can subsequently adjust the axes to get a nice-looking plot.

How many red giants do you see in this dataset?

1.6 Repeat this for 47 Tuc: estimate its age, distance and metallicity by overplotting the best-fitting isochrone. It may help here to only plot data points with small errors. For example, try including only data points with an error $< 0.05$ mag on the $b-y$ colour (column 8 in the data file). *Hint:* use the `numpy.where` command.

1.7 Can you think of an effect which we have ignored, but which might be important? What would the effect be and how would it bias your result?

## 1.5 The Initial Mass Function (IMF)

A disadvantage of using isochrones alone, is that they contain no information on how a number of real stars are actually distributed along the isochrone. Since the position along an isochrone is determined by the mass of the star, we need to include information

about the *Initial Mass Function* (IMF) of the stars. This describes the number of stars formed per mass interval, as a function of mass. We will assume a "Salpeter IMF", which is a power-law with an index of −2.35:

$$N(M)\mathrm{d}M \propto M^{-2.35}\mathrm{d}M. \tag{1.3}$$

To sample $n$ random masses from a Salpeter IMF between $M_{\rm min}$ and $M_{\rm max}$, we suggest to use the following function in `Python`:

```
import numpy.random as ran

def plaw(n, xmin, xmax, a):
# n = number of sources you are simulating (e.g. stars)
# xmin = lower limit of the distribution
# xmax = upper limit of the distribution
# a = power-law index (including the correct sign!)

    norm = (xmax/xmin)**(1.+a) - 1.
    mi = xmin * (norm * ran.random(n) + 1.)**(1./(1.+a))
    return mi


# Call the function:
masses = plaw(10, 1, 10, -2.35)
```

You can program a function like this yourself, or you can copy the routine `plaw.py` from the website and save it in your working directory. You can then include it at the beginning of your `Python` code and call it as follows:

```
masses = plaw(10, 1, 10, -2.35)
```

in this case generating an array, `masses`, with 10 random numbers in the range from 1 to 10, drawn from a power-law with exponent −2.35.

Notice that `Python` cares a lot about *indentation* of the text. In the `plaw` function above, the main statements of the function are shifted to the right by 4 characters. The number of characters is not important (though 4 is the recommended number) but must be consistent for all statements in a block. There is no specific statement that signals the end of a function (or other blocks of text, such as `if` or `while` statements). `Python` will simply assume the end of the block has been reached when it finds statements at the previous level of indentation.

Once we have sampled a number of stellar masses, we need to interpolate in the isochrone data in order to assign a $B$ and $V$ magnitude to each stellar mass. This can be done with the `interp1d()` function in the `scipy.interpolate` package. This works as in the following example:

```
import scipy.interpolate as ip
...
(read the m and mv arrays from the isochrone file,
 and select stars of a given age, as before)
...
mvfunc = ip.interp1d(m[w9], mv[w9])
masses = plaw(n, mmin, mmax, -2.35)
imv = mvfunc(masses)
...
```

We first import the `scipy.interpolate` library under the shorter name `ip`. We then call `interp1d()` with two parameters, the "$x$-values" and "$y$-values". The function `interp1d()` returns a new function (here called `mvfunc`) which will return new interpolated $y$-values for an array of $x$-values passed as parameter to this function. Hence, the array `imv` will contain interpolated $V$ magnitudes corresponding to the array `masses`.

1.8 Plot a simulated $V$ versus $B-V$ diagram for 1000 stars in a star cluster with log(age) = 9.0 and $Z = 0.008$. First sample the masses of the stars randomly according to a Salpeter IMF. Make sure to pick $M_{\rm min}$ and $M_{\rm max}$ from the correct isochrone.
*Hint:* you can use the built-in `min()` and `max()` functions in `Python`.
You can then use `interp1d()` to interpolate the isochrone for every mass.
You can change the plotting style of `plot()` from the default (lines) to '+' markers by giving an extra parameter:

```
plt.plot(x, y, '+')
```

Many different plotting symbols, colours, etc., are possible, see the on-line documentation or the built-in help.
About how many red giant stars do you see?

## 1.6 The Carina dwarf spheroidal galaxy

In Fig. 1.2 you see a CMD of the Carina dwarf spheroidal galaxy (Monelli et al. 2004). This galaxy clearly shows evidence for multiple stellar populations.

1.9 How many populations do you recognize? What could be the cause of the "noise" for $B - V > 0.5$?

We are going to simulate this CMD by expanding the `Python` routines we wrote so far. Since we want to simulate different CMDs, and then combine them in a single figure, it is a good idea to rewrite the parts of `Python` code you have so far into 1 function, which returns $B - V$ and $V$ as a 2-dimensional array. For example:

```
def cmdfunc(logage,isofile,nstars,dmodulus):

# Read the isochrone...

# Take the correct age from the isochrones with
# a "where" statement:
    wloga = numpy.where(...)

# Sample masses from a Salpeter IMF:
    mass = plaw(...)

# Interpolate the isochrone to get the magnitudes of the sampled masses...
# Add the distance modulus...
# Add observational errors...

    return mag1-mag2, mag2
```

If you call this function with

```
bv, v = cmdfunc(8.0,'isoc_z008',1e5, 20.24)
```

`bv` should contain $B - V$ for a population of 100 000 stars with log(age) = 8.0 and $Z = 0.008$ and a distance modulus $m - M = 20.24$, and `v` will contain their apparent $V$ magnitudes.

1.10 Copy/paste parts of your current code into a new code and rewrite it as a function according to the above structure. It should return $B - V$ and $V$ for a specified age, isochrone file, number of stars and distance modulus.

To make the simulated CMD look more similar to the observations, you can add Gaussian errors to the photometry. For this, you may use the `numpy.random.normal()` Gaussian random number generator. When called with a `size=n` option, this function will return an array of $n$ random numbers, drawn from a normal distribution with mean=0 and sigma=1. You may assume that the actual photometric errors vary with *apparent* magnitude as

$$\sigma = 0.02 \times 10^{0.2\times(m-22)}$$

in both the $B$ and $V$ filters. Hence, the arrays output by `numpy.random.normal()` should be scaled by this factor before the errors are added to the simulated magnitudes.

1.11 Write a separate routine to plot a simulated $B-V$ versus $V$ diagram for the Carina dwarf galaxy, using the function you just wrote. The distance modulus of this galaxy is $m - M = 20.24$. Assume a metallicity of $Z = 0.004$. Experiment with the number of populations, the number of stars in each population and their ages, until you get a CMD that resembles Fig. 1.2 as closely as possible. How many populations do you need? What are their ages?

## 1.7 Report

Write a short ($\approx$ 5 pages) report, in which you summarize this exercise. Include the following points:

- Give some general background information about the properties of star clusters, dwarf galaxies and larger galaxies. What, in general, do we know about the star formation histories of these systems?
- Discuss the general properties of Hertzsprung-Russell diagrams and colour-magnitude diagrams, and how these depend on age, metallicity and any other important parameters you can think of.
- A plot of $V$ versus $B - V$ for 47 Tuc, with your best-fitting isochrone overplotted. Explain how you estimated the age, distance and metallicity of this globular cluster.
- A plot of a simulated CMD of the Carina dwarf spheroidal galaxy. Discuss the parameters of the different stellar populations you included in your simulation. What can you say about the star formation history of this galaxy?

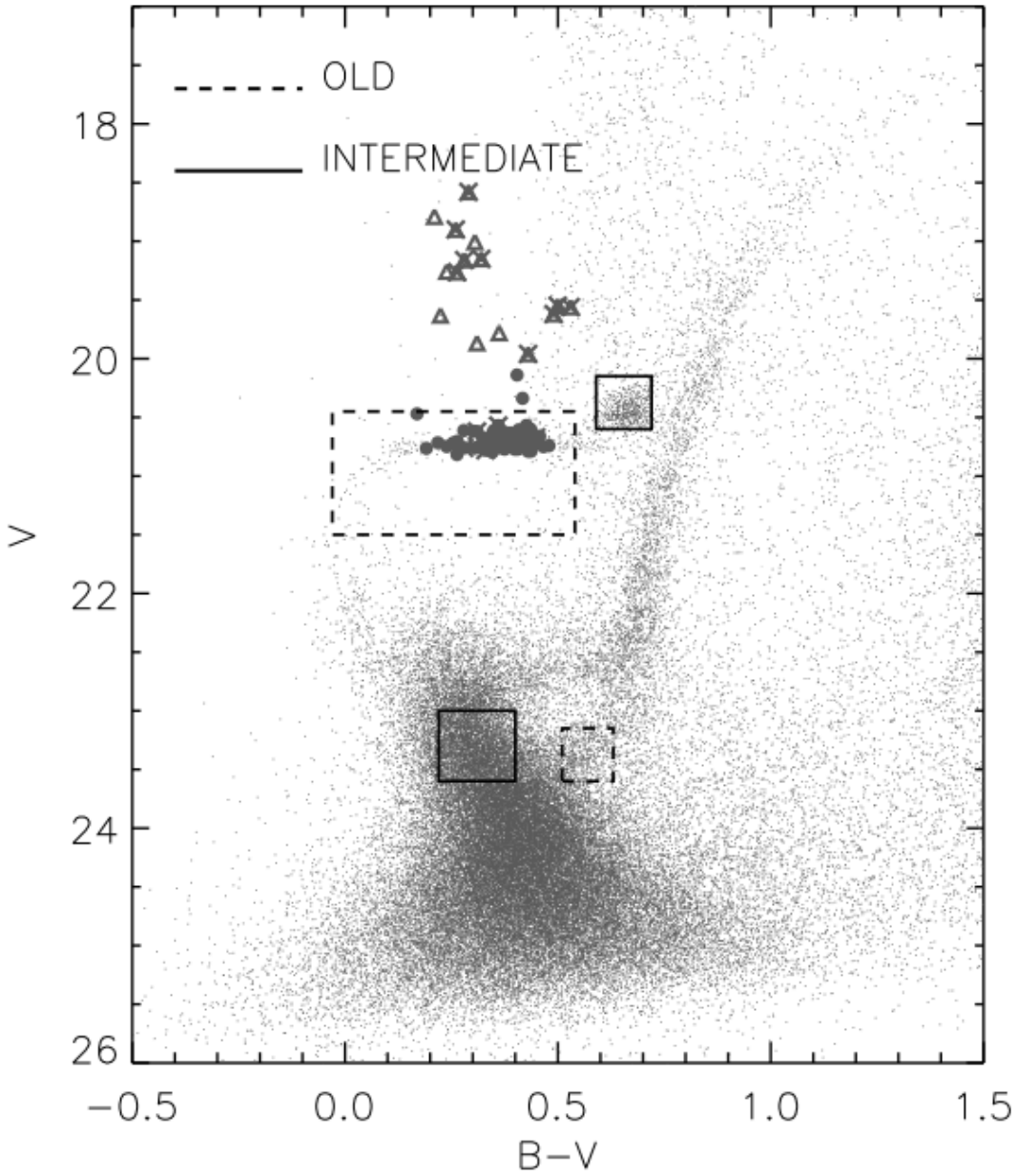


Figure 1.2: Colour-magnitude diagram of the Carina dwarf spheroidal galaxy (dSph) (Monelli et al. 2004).