

SMc20374: Programming for Prototyping**Assignment****Scenario**

You are approached by a research institute to design and prototype a desktop application prototype for the statistical analysis of forthcoming data sets, using the Fisher matrix formalism. The user would specify the path of a text file containing a covariance matrix, and another file specifying the model parameters along with their corresponding fiducial values. The system would return error ellipses of two specified parameters as a figure within the desktop application, and a statistical report is saved to a text file. You are to build a selection of prototypes that eventually lead to a software artefact.

Section A

In this section, you will be creating the preliminary prototypes that would facilitate the programming of prototypes in Section B.

1. Prepare an initial **acceptance document** (two pages maximum) where you present your understanding of the problem in the form of user stories, constraints, and the list of prototypes that will be delivered.

[5 marks]

2. Create **two** variations of how the problem can be solved by illustrating each variation with a **hand-drawn storyboard** that has at least **eight frames** and a maximum of **sixteen frames**.

[5 marks]

3. Create **two hand-drawn sketches** of the desktop interface, and for each create **one office-stationary based prototype** by making use of post-it notes, blank A4 papers and medium markers. For each prototype, create variations by moving components around and photographic changes/variations.

[5 marks]

4. After choosing the best option, proceed to building a **PowerPoint mock-up** of the interface and its logical flow.

[5 marks]

Section B

In this section, you will be creating a high-fidelity prototype using Python together with a selection of scripts. You will also be developing a code to obtain the confidence ellipses and apply marginalization on a number of parameters describing a cosmological model of our Universe.

1. Create a **plan** for the development of the prototyped application. This plan should include a brief overview of libraries that can be possibly used together with their pros and cons. This overview should consider both visual and technical attributes.

[5 marks]

2. Design a **flow-chart** of the final application.

[10 marks]

3. Use a standard Python GUI framework (**Tkinter**) to create an **interface** for this program based on your work in Section A.

[15 marks]

4. Implement the following tasks in a well-documented Python script `Fisher_Matrix.py`:

- (a) The inverse of the Fisher matrix is the covariance matrix, which for two model parameters x and y , is given by the square matrix:

$$[F]^{-1} = [C] = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix},$$

where σ_x and σ_y denote the $1-\sigma$ uncertainties in the parameters x and y , respectively. $\sigma_{xy} = \rho\sigma_x\sigma_y$, with ρ being the correlation coefficient which varies from 0 to 1. **Load** the covariance matrix provided in the file

`base_w_plikHM.TTTEEE_low1_lowE_BAO_Riess18_Pantheon.covmat` from Moodle, and save it as a 28×28 matrix.

[5 marks]

- (b) To calculate a new Fisher matrix marginalized over any variable, simply remove that variable's row and column from the covariance matrix, and take the inverse of that to yield the new Fisher matrix. Thus, if you are interested in the first two parameters of the covariance matrix (refer to the first line of the covariance matrix file), which are `omegab2` ($\Omega_b h^2$) and `omegach2` ($\Omega_c h^2$), you would remove all the columns and rows of the other parameters from the original covariance matrix, and you end up with a 2×2 covariance matrix. Generate **four** marginalized covariance matrices and corresponding Fisher matrices for the following parameter combinations:

$$\Omega_b h^2 \text{ vs } \Omega_c h^2,$$

$$\Omega_c h^2 \text{ vs } w,$$

$$\ln(A) \text{ vs } n_s,$$

$$\tau \text{ vs } w.$$

Write these matrices to the console in **matrix form**.

[15 marks]

- (c) The confidence ellipse parameters are calculated as follows:

$$\begin{aligned}
 a^2 &= \frac{\sigma_x^2 + \sigma_y^2}{2} + \sqrt{\frac{(\sigma_x^2 - \sigma_y^2)^2}{4} + \sigma_{xy}^2}, \\
 b^2 &= \frac{\sigma_x^2 + \sigma_y^2}{2} - \sqrt{\frac{(\sigma_x^2 - \sigma_y^2)^2}{4} + \sigma_{xy}^2}, \\
 \tan(2\theta) &= \frac{2\sigma_{xy}}{\sigma_x^2 - \sigma_y^2},
 \end{aligned}$$

where θ is the anticlockwise angle of the semi-major axis, and the ellipse axis lengths a and b are multiplied by a coefficient α depending on the confidence level we are interested in. For instance, for a 68.3% confidence level we have $\alpha = 1.52$.

Define a function `ellipse_params` which returns the confidence ellipse's width, height and inclination. This function should allow the user to input σ_x , σ_y , σ_{xy} and the confidence level parameter α . Call the function `ellipse_params` and **output** to the console the width, height and inclination of the ellipse with $\alpha = 1.52$ for all the above parameter combinations.

[15 marks]

- (d) **Plot and label** the 1- σ ($\alpha = 1.52$) and 2- σ ($\alpha = 2.48$) confidence ellipses for **all** the above parameter combinations in **separate** figures. Assume that the center of each ellipse lies at the following fiducial parameter mean values of $\Omega_b h^2 = 0.022$, $\Omega_c h^2 = 0.12$, $w = -1$, $\ln(A) = -19.94$, $n_s = 0.96$ and $\tau = 0.09$. Mark the center of the ellipses with a **red dot**. For each figure plot 500 sample points generated from a random Gaussian realization using `numpy`'s random sampling module. For this multivariate Gaussian realization you should assume the center of the ellipses as the mean value of the Gaussian distribution, and the respective covariance matrix. A sample figure is illustrated in Fig. 1. **Save** each figure separately.

[15 marks]

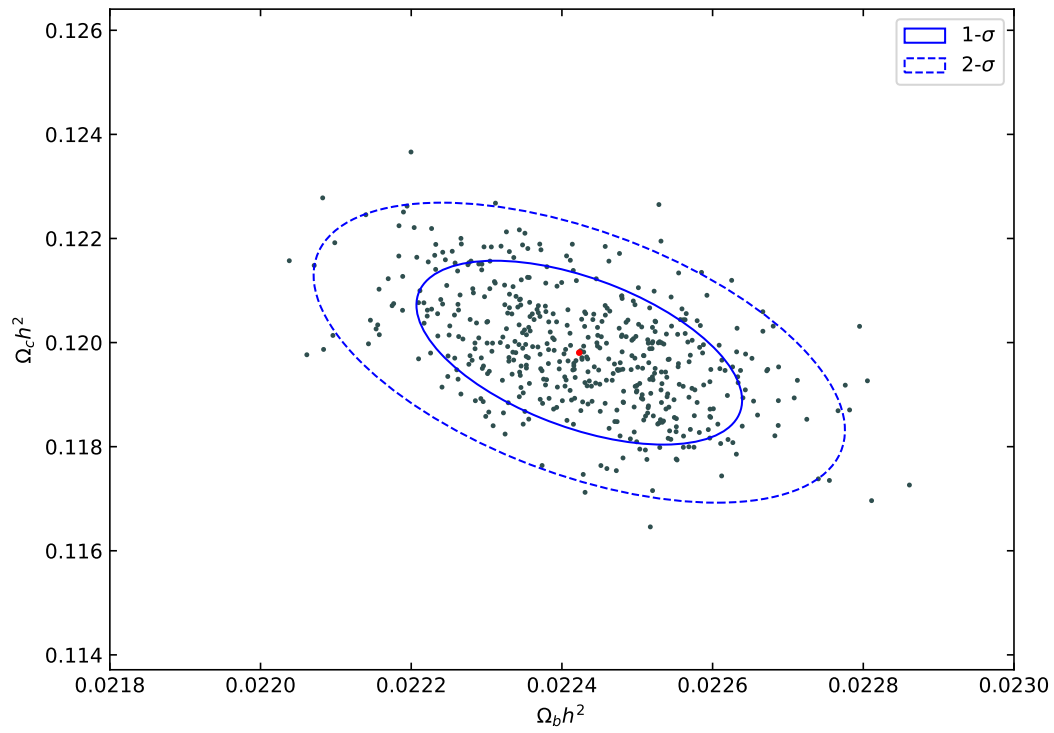


Figure 1: Confidence ellipses