Fisher4Cast Version 2.2

User's Manual

Bruce A. Bassett[1,2,3], Yabebal Fantaye[1,2,4], Renée Hlozek[1,2,6], Jacques Kotze[2,5,6]

[1] *Department of Mathematics and Applied Mathematics, University of Cape Town*
Rondebosch, 7700, Cape Town, South Africa
[2] *South African Astronomical Observatory*
Observatory, Cape Town, South Africa
[3] *Astrophysics and Cosmology Group*
African Institute for Mathematical Sciences, Muizenberg, South Africa
[4] *Department of Astrophysics, SISSA*
Via Beirut 4, 34014 Trieste, Italy
[5] *Institute of Cosmology and Gravitation (ICG), University of Portsmouth*
Dennis Sciama Building, Burnaby Road, Portsmouth PO1 3FX, United Kingdom
[6] *Department of Astrophysics, Oxford University*
Denys Wilkinson Building, Keble Road, OX1 3RH, United Kingdom

Electronic address: **fisher4cast@gmail.com**

Chapter 1

The Code

Here we introduce Fisher4Cast, a code developed for general Fisher Matrix analysis. In addition to this manual there is also a release paper for Fisher4Cast which provides more background detail and cosmological applications as well as sample code for generating the plots in the release paper [4]. This Users' Manual explains how to install and run the code, both in the Command-line and the Graphical User Interface (GUI), which is coded for cosmology. The structure of the code is described in detail, as well as the various tests performed during the development of the code.

1.1   Introduction

The Fisher Matrix translates errors on observable quantities measured in a survey into constraints on parameters of interest in the underlying model. As such, it is the elegant way of doing propagation of errors to the case of multiple measurements and many parameters [26].

In contemporary cosmology, Fisher matrices are used to forecast parameter constraints from a proposed survey, and can be used to optimise future surveys (see [4] for a detailed discussion on the Fisher Matrix formalism). Fisher4Cast was developed with the aim of providing the community with a free, standard and tested tool for Fisher Matrix analysis, that is both easy to use through the Graphical User Interface, and yet also a robust general base-code for research. The underlying modular code of Fisher4Cast is completely general and is not specific to cosmology although the default setup for the GUI is intended for cosmology. It provides parameter error forecasts for cosmological surveys providing distance, Hubble expansion and growth measurements in a general, curved, FLRW background.

The code is written in Matlab and can be redistributed and modified under the Berkley Software Distribution (BSD) license* which is now the required license for the Matlab File Exchange†, the primary repository for Fisher4Cast. Both the command line and GUI versions of the code produce plots that are generated directly from the program and can be easily edited in the GUI and saved in a variety of standard formats ('.eps', '.jpg', '.pdf' etc...) for inclusion in research publications. There is also a report

---

*Read the BSD license available in the Fisher4Cast distribution, or go to http://www.opensource.org/licenses/bsd-license.php for more information.

†http://www.mathworks.com/matlabcentral/fileexchange/

extension facility which generates either a text or LaTeX report complete with syntax for incorporating the input and output in tables, matrices and figures. This means that both input and output data and results are easily portable from Fisher4Cast into a research publication.

The simple start-up procedure and ease of use of the Fisher4Cast suite make it well-suited to both teaching and research. The input to the Fisher4Cast code can easily be changed and adapted, hence is can be run in large loops to explore parameter spaces, and for visualisation of the Fisher Matrix. We now describe the Fisher Matrix framework, outline the start-up procedure of Fisher4Cast, and describe the various functions and routines. A shortened version of the start-up procedure of Fisher4Cast is found in the **Quickstart.pdf** guide – which is included both as an appendix in [4] and in the bundle of Fisher4Cast software.

1.2   Getting Started

Currently the code is available for download at two websites[‡]. The '.zip' file into the directory you want to run the Fisher4Cast suite from.

1.2.1   The Graphical User Interface

The GUI can be started from the Matlab editor. The file **FM_GUI.m** must be opened from the directory, and once the file is opened (click on the file icon from within the Command-line interface to open it with an editor) press 'F5' to run the code. This will open up the GUI screen.

You can also launch the GUI from the command line by typing:

```
>>FM_GUI
```

The output data will not be saved into the workspace, but the 'Saving Features' button allows one to save the input and output from any particular run in text or LaTeX code.

The Basic Layout Explained

We describe the basic layout of the GUI, and illustrate the various actions with screenshots taken of a working GUI.

The GUI has three main sections. The section on the top left controls the input to the GUI. The bottom left panel controls the things one might like to use in the analysis and the parameters you are interested in plotting. In Figure (1.1) we show the initial GUI screen, highlighting the observable about to be used (here the Growth function $G(z)$), and the cosmological parameters relevant to the analysis (which are the w_0 and w_a coefficients in the Chevallier-Polarski-Linder parameterisation of dark energy [7, 16] – see Eq. (1.3)). The specific cosmological example is described in detail in [4], which contains the set of analytical derivatives used in Fisher4Cast. The right-hand side of the GUI controls the plotting commands for the ellipse. The various actions used to control the output are described below.

Changing the Input Structure

In order to compute Fisher ellipses for different input structures, one can either choose from a drop down list of default example structures contained within the distribution (as shown in Figure (1.2)) or one can

---

[‡]http://www.cosmology.org.za and http://www.mathworks.com/matlabcentral/fileexchange/.
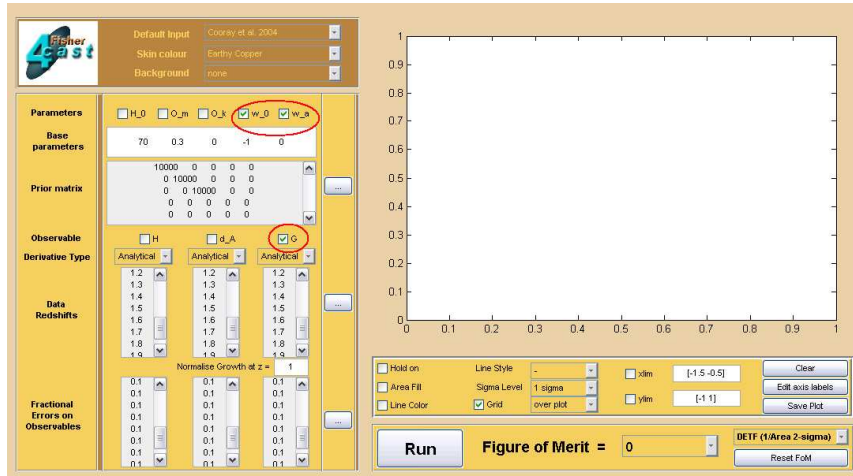
Figure 1.1: The start-up screen of the Fisher4Cast GUI.

generate a unique input structure. This file can then be loaded to the GUI, which must be given as '.m'
file. You can obviously also just edit the input parameters in the GUI after the default input has been
loaded or alternately you can edit the input file (eg **Cooray_et_al_2004.m**).
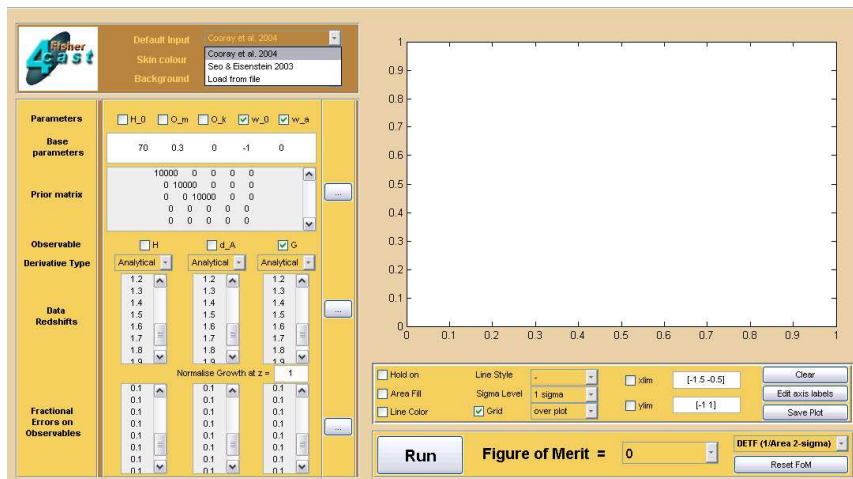
Figure 1.2: Changing the default input structure from the drop-down menu.

Floating Help

Floating help is provided with the Fisher4Cast GUI for most commands. The floating help is activated
by moving the mouse pointer over the button or parameter on the GUI and leaving it there for a few
seconds. This generates a screen prompt, which pops up and gives information about the function of the
button or parameter in question. Figure (1.3) shows this help prompt for the 'Run' button on the GUI.

Figure 1.3: The floating help for the 'Run' button.

Running Fisher4Cast

Once satisfied with the observables considered and the parameters of interest, pressing the 'Run' button will execute the code. A box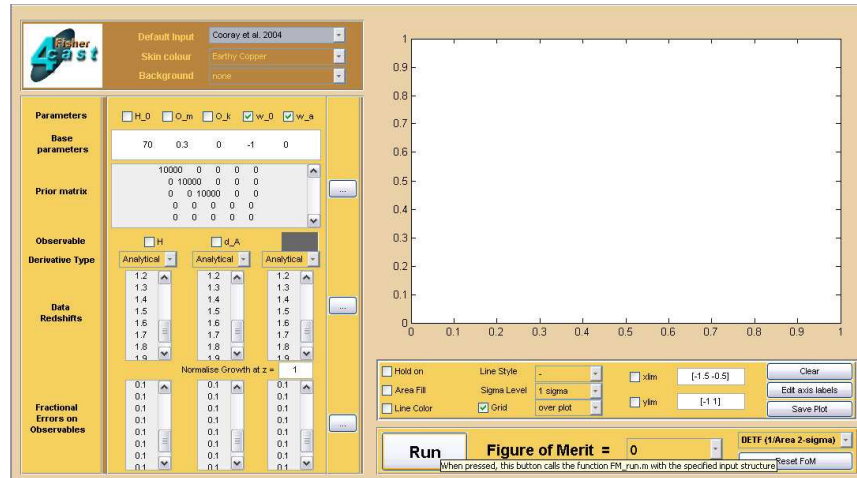 will pop up that will state that the code is running, and an error ellipse will appear when the code has finished running. This is shown in Figure (1.4).
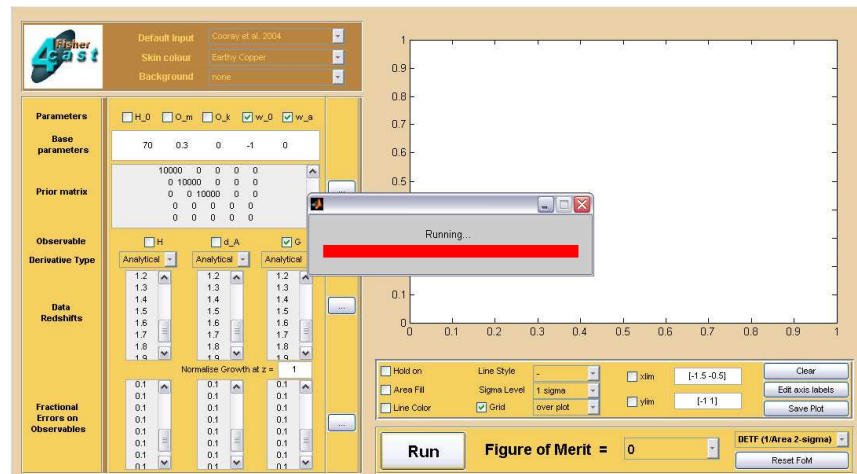
Figure 1.4: Running the Fisher4Cast from the GUI.

Errors in the Input

When the 'Run' button is pushed, the GUI first calls the **FM_errorchecker.m** function with the input supplied. This checks for the input files, checks that the data vector (e.g. redshifts at which one has measurements of the Hubble parameter) and error vectors (e.g. the fractional errors on the Hubble parameter, σ_H/H, at the redshifts above) are the same length and performs other consistency checks. Should any of these tests fail, an error box will appear explaining which errors to fix before calling the GUI again. A log file of these errors is created in the same directory the GUI is being run in, and is

called 'log.mat'. Loading and reading this log file is described in Section 1.2.3. Figure (1.5) shows the error dialogue box indicating that a single error has been found.
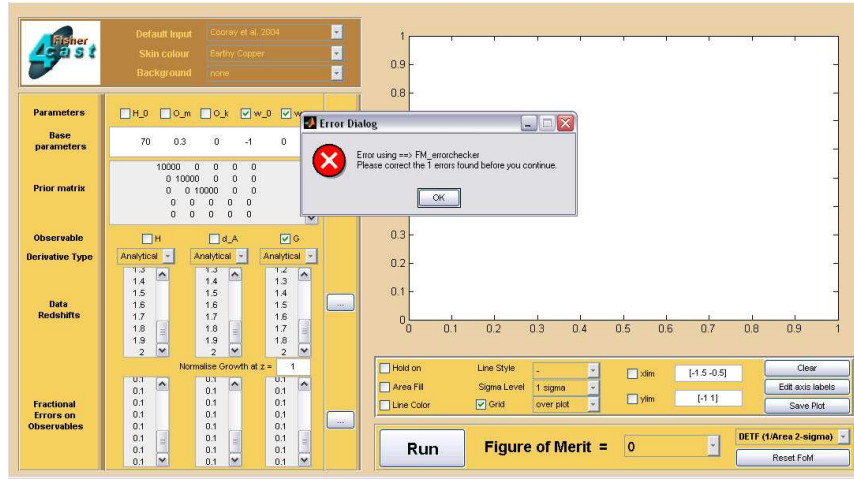
Figure 1.5: The GUI error message box. The errors are detailed in 'log.mat'.

The Fisher Ellipse

Once the code is running smoothly, the resulting Fisher error ellipse is plotted. This is shown in Figure (1.6).
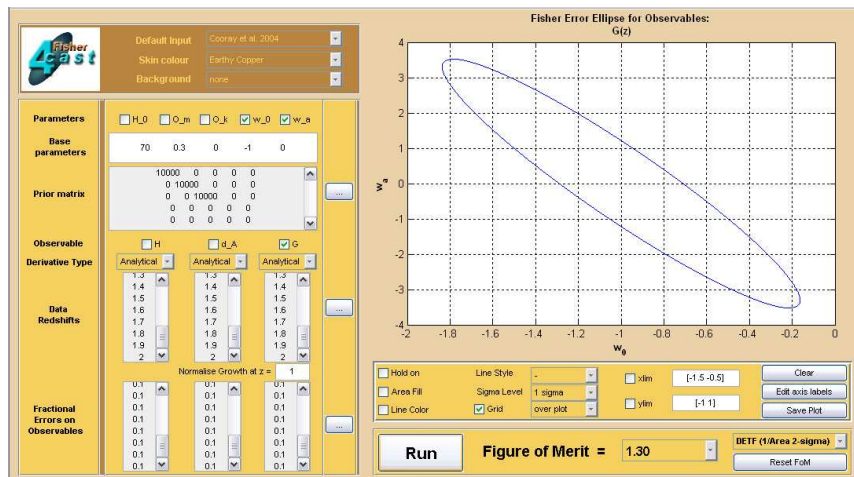
Figure 1.6: The Fisher ellipse resulting from a run of Fisher4Cast.

Plotting more than one Ellipse

Should one want to superimpose more than one ellipse, click the 'Hold on' button. This works both for the line and the area (although the same line and area fill properties will be used for both ellipses – see the below item for discussion of changing the colour of the area fill). Figure (1.7) shows the resulting ellipse for two observables, $G(z), d_A(z)$.
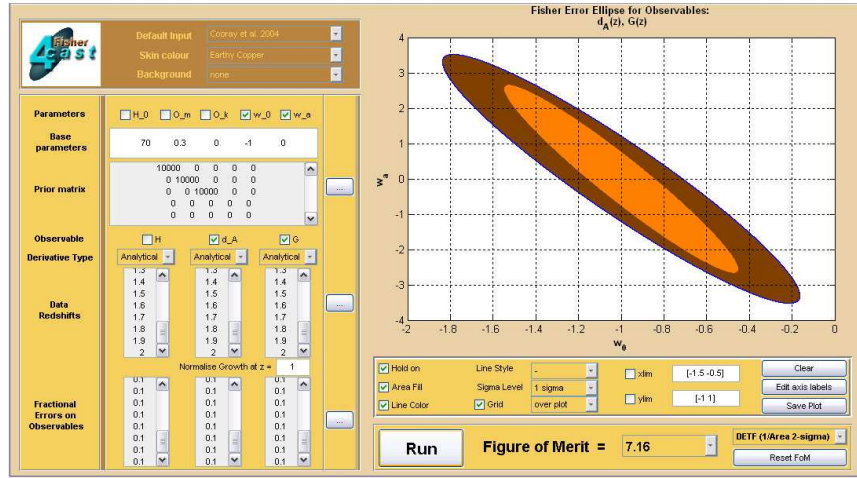
Figure 1.7: Multiple ellipses are plotted in one figure.

Area fill

Clicking on this button yields a filled error ellipse. Once it is clicked a colour must be selected from the menu on the left pop-up box. Note that should more than one error ellipse be plotted later, this area fill box must be ticked and un-ticked again to change the colour, otherwise the same colour will be used for all filled ellipses. This box is shown in Figure (1.8).
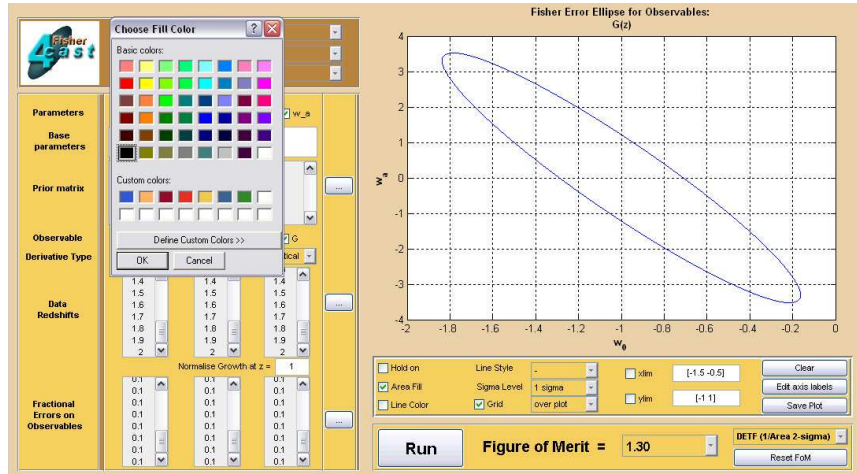
Figure 1.8: The Area Fill option with colour selection.

Importing Data

The input data can also be imported from a file - either as the redshift vector (the data), the error vector or the matrix of prior information on the cosmological parameters. This can be done by clicking the relevant 'Browse' buttons on the GUI. This brings up a screen in which one can either load the data from file, or from the clipboard, in which case the data is cut and paste into the GUI fields. Figure (1.9) shows the screens for the loading of data from a file in the directory. In addition there is a check-box which specifies whether or not to use the prior matrix.
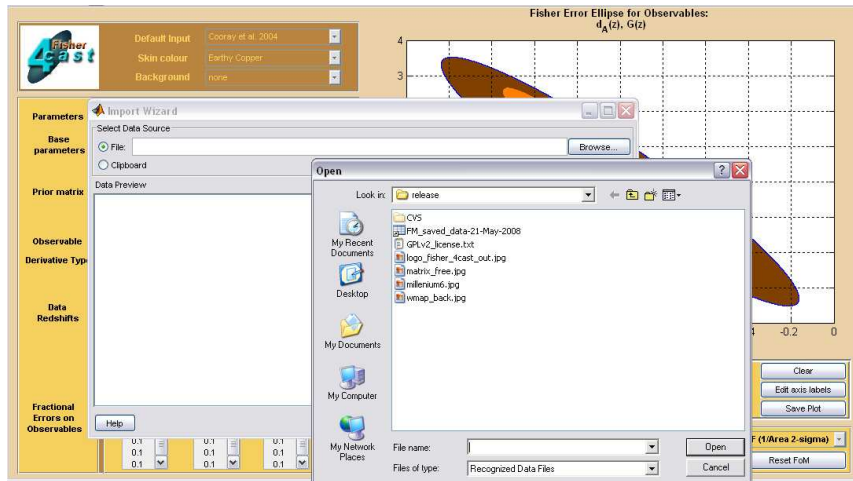
Figure 1.9: Loading data into the GUI.

Multiple σ

It is possible to plot the ellipses for multiple confidence levels (i.e. $67\%, 95\%, 99\%$ specified by $1-, 2-,$ and $3-$ σ respectively). This is done via a drop-down menu on the right-hand side of the GUI, and is illustrated in Figure (1.10). It is worth noting that the 'Hold on' was used to generate this plot.
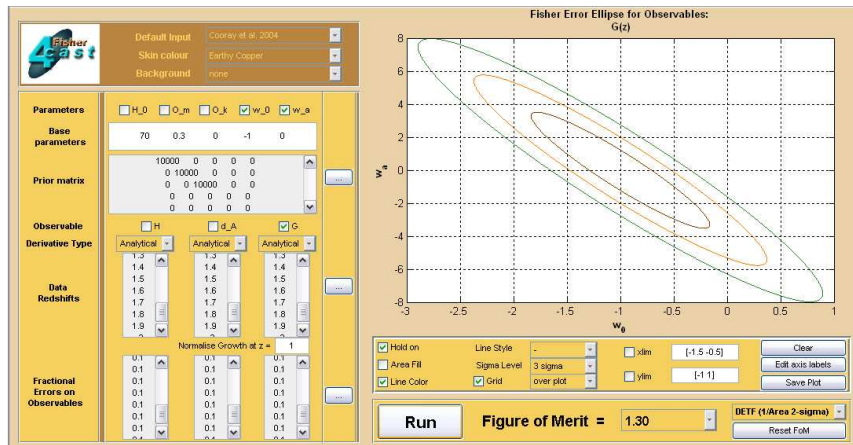
Figure 1.10: Ellipses with many σ.

Different Definitions of the Figure of Merit

Fisher4Cast provides various Figures of Merit in order to compare different surveys. These are defined in detail in Section 1.4.4. Figure (1.15) illustrates how these various FoMs are accessible in the GUI.

Controlling Output

The buttons on the right-hand side of the GUI all control the output specifications of the ellipse, such as the limits of the x and y axis, the line style and colour of the ellipse, and whether or not to have a
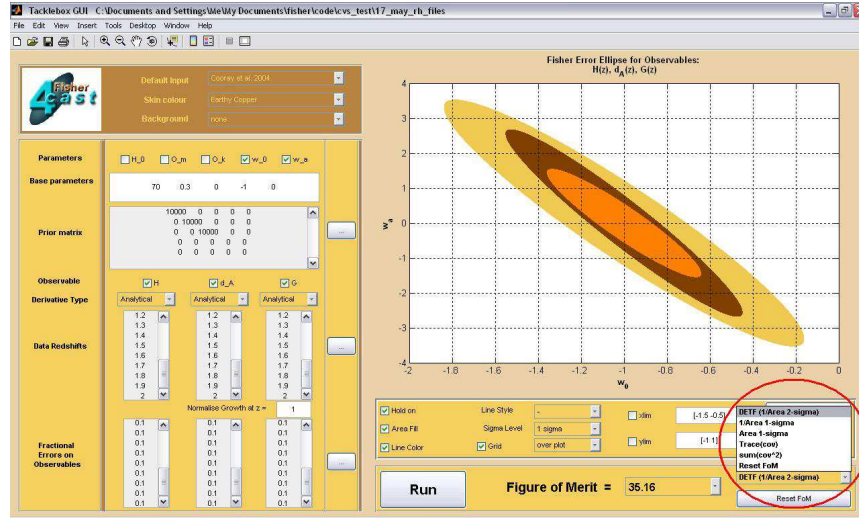
Figure 1.11: Various Figures of Merit σ are calculated in Fisher4Cast.

grid on (over or under) the data. This is designed for maximum flexibility in representing the ellipses in a unique and distinguishable way. The axis labels can also be modified using the 'Edit Axis' button.

Saving the Plot

Once satisfied with the ellipses plotted, figures are saved by clicking on the 'Saving Features' menu and selecting the 'Save Plot' option. This will bring up a window to save the figure to a particular directory, in a selected file format ('.eps', '.fig', '.png', '.pdf' etc.), as illustrated in Figure (1.12). No legend information (for example which colours correspond to which ellipses) will be saved. If legend entries are required, the figure must be saved as a '.fig' file, opened with Matlab, with the required legend entries added later. In
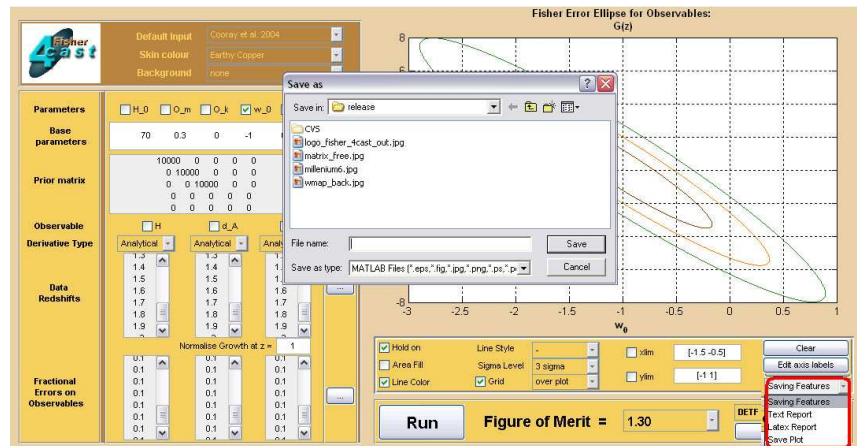
Figure 1.12: Saving the figure of your Fisher4Cast ellipse.

addition to the simple saving of the figure, the reporting features of the Fisher4Cast code are accessed through this 'Saving Features' menu. In short, they provide the opportunity to save a report of the inputs and the resulting outputs and Fisher matrices from the survey, in either ASCII text or LaTeX format. When clicking the 'Text report' feature the user is prompted to save the resulting '.txt' file. Similarly if

one chooses the LaTeX report, both a '.tex' report and an Encapsulated Postscript File will be generated, and the user is prompted to save both the figure and the LaTeX document. These reporting features are discussed in more detail in Section 1.6.2.

Skins

The GUI is available in a variety of skins and backgrounds. These can be chosen from a drop-down list (consisting of both colour schemes and background images [§]); additional background images can be loaded by the user. This is shown in Figure (1.13).
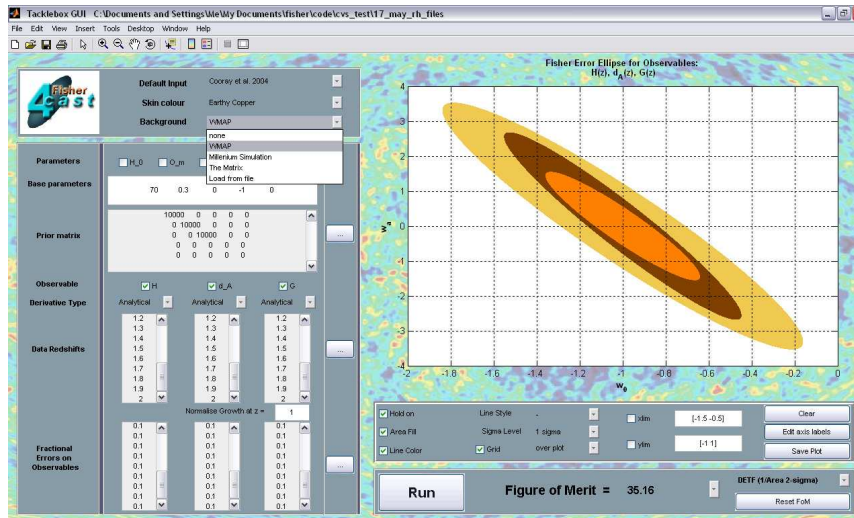
Figure 1.13: Changing the skins and background images.

Fisher4Cast Menu

A Fisher4Cast menu is defined in the top left-hand corner of the Fisher4Cast GUI. From this drop-down menu one can access the **Readme** file of the code suite, the **Users' manual** and **Quickstart Guide** for easy reference, and the version history of the code. The BSD licence[¶] for the Fisher4Cast suite is also available from the drop-down list. This list is illustrated in Figure (1.15). In addition to the Fisher4Cast menu, there is a menu with information on the extensions available for Fisher4Cast. In future releases of the code this menu will select the extensions themselves, at present it provides the **Readme** for the modules.

Interactive Plotting

Interactive 'point-and-click' plotting is available in Fisher4Cast Version 2.2, available by selecting the 'Activate Interactive Plotting' option from the 'Fisher4Cast Extension' menu. Once selected the user interactively sets the values for parameters being plotted by clicking on the plotting area of the GUI. The arrow is activated for the first click on the plot area, the next click will run the code and produce the

---

[§]see the images http://commons.wikimedia.org/,http://map.gsfc.nasa.gov/ and http://www.mpa-garching.mpg.de/galform/millennium/.

[¶]http://www.opensource.org/licenses/bsd-license.php.
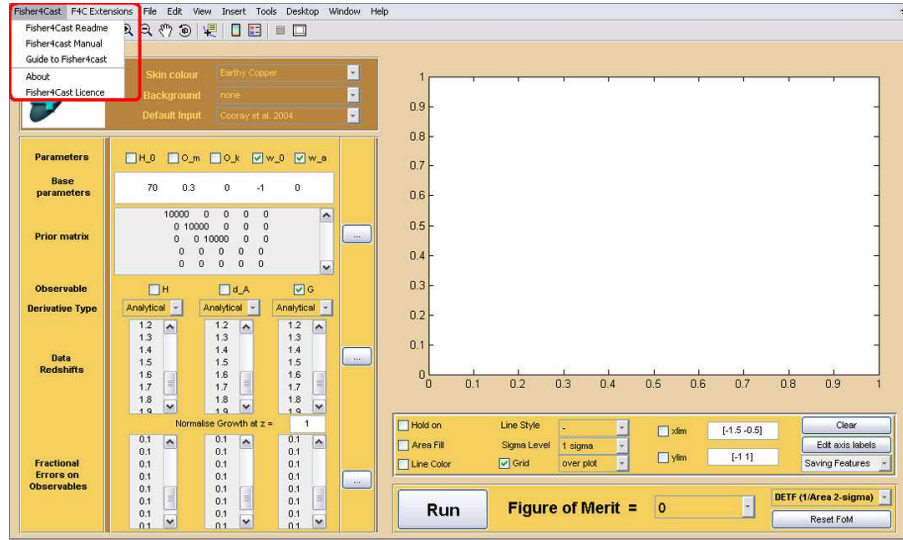
Figure 1.14: The Fisher4Cast drop-down menu with information on the code and version.

appropriate ellipse. The values selected will be displayed in the parameter input section of the GUI, the same as they would have been if manually entered. Care should be taken not to step to very unphysical values of the parameters, e.g. very positive values of w_0.
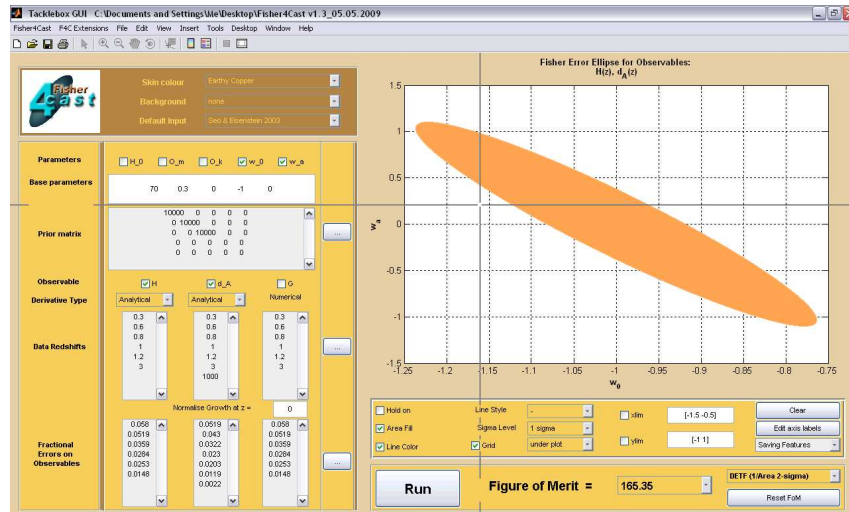
Figure 1.15: The Interactive Plotting Feature in the Fisher4Cast code. This allows you to click on the parameter plane to change the assumed fiducial model.

1.2.2   The Command Line

Running the Code

Open your version of Matlab and change the working directory to be the same as where you saved Fisher4Cast in. To run the code from the command line with one of the standard test input structures

supplied, type:

```
>>output = FM_run(Cooray_et_al_2004)
```

This will call the code using the pre-supplied test input data (**Cooray_et_al_2004**) and then generate an error ellipse plot for the parameters and observables supplied in the chosen input. All the relevant generated output is written to the output structure. You can see the range of outputs to access by typing:

```
>>output
```

and then examine each output individually by specifying it exactly. For example:

```
>>output.marginalised_matrix
```

will access the marginalised_matrix field in the output structure. It is worth noting that each '.' denotes another sub-level in the input structure.

Example input files are supplied as a template for generating new input files with your own customised parameters and values. All fields specified in the example inputs must be specified in any user-defined example input. These are outlined in Section 1.4.2.

The code can also be run from the Matlab editor. Once the code is opened (open it from inside the Matlab window), pressing 'F5' will run the code. Note that if the code is run from the Editor it will call the default input structure, which is the **Cooray_et_al_2004.m** file. This is an example file containing input data from the paper by Cooray *et al.* [8]. This output can be directly compared to that of Figure 1 of that paper. If your output compares correctly, you have a working installation of the code. Another input available is **Seo_Eisenstein_2003.m** [24].

1.2.3   FM_errorchecker

The error-checker function acts 'behind-the-scenes' to check that the input structure and all the required variables are correct before executing the code. It can be run directly by using the command:

```
>>FM_errorcheck(FM_initialise)
```

where **FM_initialise** is the specific function to initialise the input structure. The error checker validates, among other things, that all the derivative functions (whether analytical or numerical derivatives are going to be implemented) do in fact exist and that the data and corresponding variances vectors are the same length. This error checker is continually being updated to facilitate ease of use of the code. All error and checking messages are displayed to the screen and are also saved in the Matlab file 'log.mat' which can be loaded and examined at a later stage by invoking the following command:

```
>>load(log.mat)
```

1.3   Flowchart

Throughout the code structures are used to allow different sub-parts of the general code access to the data. These structures are defined as global variables. The structures containing information on the input for the code are either defined at the beginning or loaded from file. All output structures can be
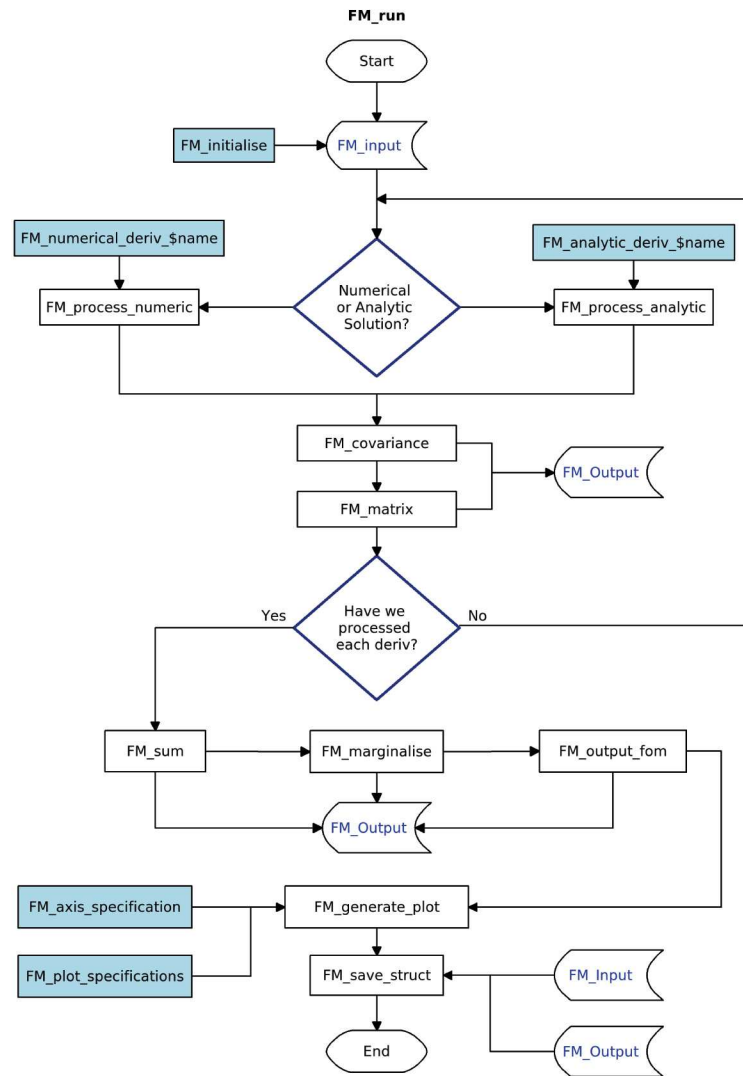
Figure 1.16: The flowchart of the code showing the processes, decisions and storage of data. For a key to the symbols see Figure (1.17).
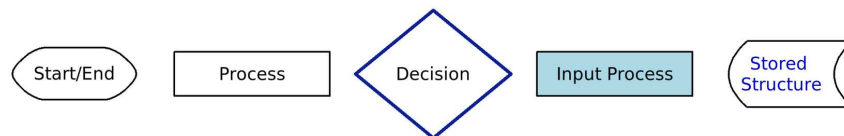
Figure 1.17: This key shows the symbols contained in the flowchart, Figure (1.16). They are from, left to right, the begin and terminate indicator; a simple processing function which would generally return an output; an if statement or for loop; an input process function designed to be edited and changed as per the user specifications and lastly a stored structure for either input or output and passed globally for use throughout the code.

saved for later use. We now outline the general framework of the code and describe the cosmological example specifically coded in Fisher4Cast. The flowchart shown in Figure (1.16) summarises the layout and structure of the code, and the symbols are summarised in Figure (1.17).

1.4 Components of the Code

We now discuss the various components of the code in detail. With each section we give a subsection of the flowchart to highlight the position in the flow of information through the code structure.

1.4.1 FM_run

FM_run

Start

FM_run.m is the general wrapper of the code. In order to make the code clear and easily editable, all main processes are called from this general function, and it is where all data storage occurs. Links to separate functions for specific calculations are documented in the code. From the command line this code is called with one argument, namely the specific function that initialises the input structure. This is unique to each example. As outlined in the section on implementing the code, if no argument is given the code is run with a pre-defined function, given by **Cooray_et_al_2004.m**, which gives the parameters for a redshift survey as outlined by Cooray *et al.* [8].

1.4.2 FM_initialise

FM_initialise → FM_input

This function initialises the input used throughout the processing of the Fisher4Cast code. The values, names and areas of interest are specified here. It is called by **FM_run.m** to set the initial values for the **input** structure, which is then passed globally to all other parts of the code. Examples of the default initialising functions provided are **Cooray_et_al_2004.m, Seo_Eisenstein_2003.m** [8, 24]. It is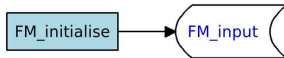 important to note that the format of the initialising function must be kept constant - in other words the same input must be specified in each initialising function - the code expects values for certain entries in the **input** structure. The entries are as follows:

- **input.function_names** - A cell of strings containing the specific filenames of the *analytical derivatives*. Note in the coded cosmological example that no analytical derivative function is specified for the growth function, derivatives are only taken numerically.

- **input.observable_names** - A cell of strings with the names of the observables.

- **input.observable_index** - A vector of the indices corresponding to the observable_names you are interested in, eg [2 3] would imply that your considering the second and third observables listed in observable_names.

- **input.data{i}** - These are the row vectors of the data for each of the respective observables (indexed again from beginning to end by i). In the cosmological example **input.data{1}** would be the redshifts at which you have measurements of the Hubble parameter, for example.

- **input.parameter_names** - A cell of strings containing the names of the parameters you can include for consideration to generate Fisher Ellipses.

- **input.base_parameters** - A row vector of the parameter values (they must be specified with the same order as the parameter_names vector). This is the model assumed to be true in the analysis, the Fisher Matrix is taken around this fiducial model.

- **input.prior_matrix** - The prior matrix for the parameters taken from previous surveys etc. The order of the matrix columns and rows correspond to the respective parameters listed as your parameter_names.

- **input.parameters_to_plot** - A row vector of the indices of the specific parameters you want to plot. If one index is given then a likelihood function for that parameter is plotted and if two are specified then an error ellipses is plotted. Selecting more than two parameters will produce an error message, as Fisher4Cast is only coded for up to 2-dimensional error contours.

- **input.num_parameters** - This is a derived value and is given by the length of the number of parameters you are considering in **parameters_to_plot**.

- **input.num_observables** - This is a derived value and is generated from the number of observables under consideration in observable_index.

- **input.error{i}** - The fractional error on the data from the observables ($\sigma_{\mathbf{X}^\alpha}/\mathbf{X}^\alpha$). It is key that there are as many error entries as there are observables you are considering (i.e. **input.error{1}** gives the error on your measurements of the Hubble parameter, measured at **input.data{1}**). The entry can either be a row vector, in the case of uncorrelated observables (this vector is converted to a diagonal matrix in the code) or a covariance matrix.

- **input.numderiv.flag** - A logical entry is expected here for each observable, should you wish to use numerical derivatives.

- **input.numderiv.f** - Single string entries which are combined into a struct later. These entries are only required if you have specified that you would like numerical derivatives for your observables. They give the function name of the function (say **g.m**) of which you are taking derivatives.

1.4.3   The Derivative Loop
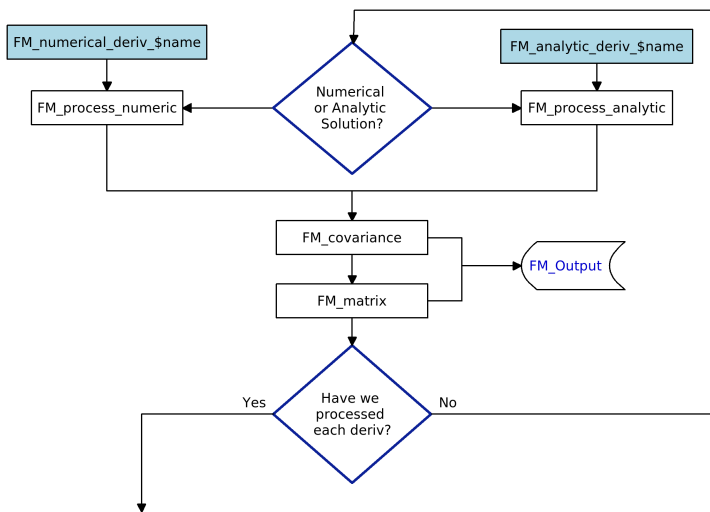
The code now runs various operations in a loop over the specific observables. For each observable it checks whether numerical or analytical derivatives are to be used by checking the numerical flag as specified in the input structure (see the above discussion on the numderiv.flag for the input structure). Both the analytical and numerical derivatives return a matrix of derivatives for all the parameters and observables

as well as a vector of the function evaluated at the data points specified. The specific details of the numerical and analytical derivative codes are outlined in the proceeding discussion. Once the selected derivative process is completed, the relevant output is stored in the **output** structure.

Numerical Derivatives

```
FM_numerical_deriv_$name
        │
        ▼
 FM_process_numeric
```
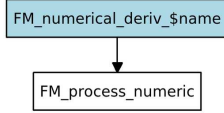
The numerical derivative code will calculate the numerical derivatives of any function (say **g.m**) provided that the function is specified as a function of the input parameters (i.e. $g = g(\mathbf{d}, \theta_A, \theta_B, ...)$), by calling on **FM_process_numeric.m** which in turn calls **FM_num_deriv.m** and passes it the name of the function you wish to take derivatives of.

The standard numerical derivative algorithm used is known as the complex-step method [21]:

$$\frac{\partial g}{\partial \theta_A} = \mathrm{Im}\left[\frac{g(\mathbf{d}, \theta_A + ih, \theta_B, ...)}{h}\right],$$

where Im represents the imaginary part of the argument, and $i^2 = -1$ as usual. This method is a second order accurate formula and is not subject to subtractive cancellation. Unlike the finite-difference method an arbitrarily small step-size can be chosen and therefore the complex-step method can achieve near analytical accuracy.

In addition, the simple double-sided central derivative is coded in the **FM_num_deriv.m** function. In order to use this algorithm the user must change the method field inside the derivative function from 'complex' to 'central'. In this case the gradient is then calculated as

$$\frac{\partial g}{\partial \theta_A} = \frac{g(\mathbf{d}, \theta_A + h, \theta_B, ...) - g(\mathbf{d}, \theta_A - h, \theta_B, ...)}{2h}.$$

This is then iterated until the gradient converges for the parameter. Note that the convergence criterion is quite stringent - and an error message will result if there are possible convergence issues. However this criterion can be relaxed by changing the settings in the **FM_num_deriv.m** code.

Once the derivatives are saved the Fisher Matrix must be calculated for this observable. This is done by first calculating the data covariance matrix for the observable. This is done in **FM_covariance.m** which is passed the function value and the index of the observable. The code checks if the error entry is a covariance matrix (in the case of correlated observables) or a vector in the uncorrelated case. It then calculates the covariance matrix by multiplying the variance with the function value at the data points considered.

FM_matrix.m then produces a Fisher Matrix (F) from the covariance matrix (C) and the derivative matrix (V) using matrix multiplication as $F = V^{\mathrm{T}} C^{-1} V$.

Analytical Derivatives

```
FM_analytic_deriv_$name
        │
        ▼
 FM_process_analytic
```
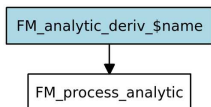
The analytical derivatives are specific to each user and example. If one knows the analytical form of both the function and of the Fisher derivatives, one can include these functions explicitly. The only conditions on these functions are that they must be of the form $g = d(\mathbf{d}, \boldsymbol{\theta})$ and must return as output a matrix of Fisher derivatives $\partial g / \partial \boldsymbol{\theta}$ and a vector of the function itself evaluated at the data points $\mathbf{d}$ given in the **input.data{i}**. These derivative functions are supplied for the Hubble parameter and angular diameter distance as **FM_analytic_deriv_1.m** and **FM_analytic_deriv_2.m** respectively. The Fisher derivatives of the angular diameter distance with respect to the cosmological parameter Ω_k must be taken as Taylor series expansion when $\Omega_k \to 0$ (see [4] for the full set of derivatives in Fisher4Cast).

As in the numerical derivative case, once the derivatives are saved the Fisher Matrix must be calculated for this observable. This is done by calculating the data covariance matrix for the observable in **FM_covariance.m** which is passed the function value and the index α of the observable. The code checks if the errors are specified a covariance matrix (in the case of correlated observables) or a vector in the uncorrelated case. It then calculates the covariance matrix by multiplying the variance with the function value at the data points considered.

FM_matrix.m then produces a Fisher Matrix (F) from the covariance matrix (C) and the derivative matrix (V) using matrix multiplication as $F = V^{\mathrm{T}} C^{-1} V$.

1.4.4 Final processing
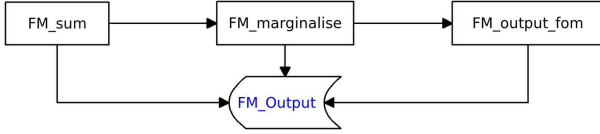
FM_sum.m collates all the derivative matrices from the previous steps and sums them together to form a full Fisher Matrix. The individual Fisher matrices for each observable are added to the prior matrix (as specified in the **input** structure). This complete Fisher Matrix is general and is assigned to the **output** structure for future reference.

FM_marginalise.m produces a marginalised Fisher Matrix (say $\widetilde{F}$). It takes the parameters you are interested in (specified as **parameters_to_plot** in the **input** structure) and shuffles the Fisher Matrix into a block form. It then performs matrix multiplication on the blocks to produced the marginalised Fisher Matrix, which is also assigned to the output structure.

FM_output_fom then produces the appropriate error for the likelihood case and a range of FoMs, listed below, for the case of an ellipse (e.g. if the length of **parameters_to_plot** in the **input** structure is two then an ellipse will be plotted). These Figures of Merit are then stored in the **output** structure. Fisher4Cast includes the standard FoMs as well as some new ones available through the GUI and command line. Those using the volume are calculated using Eq (10) in [4], with $n = 2$. Although some of the Figures of Merit are only defined for the error ellipse in the $w_0 - w_a$ plane, where w_0, w_a are the coefficients in the CPL [7, 16] parameterisation of the equation of state of dark energy (see for e.g. [2]), the FoMs in Fisher4Cast are calculated by the code for any pair of cosmological parameters being considered rather than the full 5-D matrix. We briefly outline the FoMs used in Fisher4Cast:

- DETF
  This Figure of Merit in the Report of the Dark Energy Task Force [2] is defined as $\det(F^{1/2})$, which is the inverse of the $1 - \sigma$ ellipse in the $w_0 - w_a$ plane of the CPL dark energy parameterisation [7, 16], in units of the area of the unit circle. Given that one requires the ellipse to be as small as possible, this FoM increases monotonically for survey configurations that best constrain the dark energy parameters w_0, w_a.
- $\mathrm{Area}_{1-\sigma}^{-1}$

This Figure of Merit is the reciprocal of the $1 - \sigma$ error ellipse area in the parameter plane currently plotted, i.e. $\det(F^{1/2})/(\pi\sqrt{2.31})$. While this FoM has a similar definition as the DETF FoM (and differs by a constant factor for the $w_0 - w_a$ example) it is however defined for all parameter combinations, while the DETF FoM is defined specifically for the dark energy parameters.

- $\text{Area}_{1-\sigma}$
  Simply the inverse of the previous FoM and hence is smaller for surveys which constrain the specific 2-parameter combination more tightly.

- $\text{Tr}\mathbf{C}$
  This FoM is defined as the trace of the covariance matrix of the parameters, $\mathbf{C} = \mathbf{F}^{-1}$, estimated as the inverse of the marginalised Fisher Matrix. This FoM is simply the sum of the squares of the marginalised errors on each parameter. As the errors decrease on the parameters (for more optimal surveys), the FoM decreases in magnitude.

- $\Sigma_{AB}C_{AB}^2$
  This FoM is defined as the sum of the squares of the entries of the whole covariance matrix, $\mathbf{C} = \mathbf{F}^{-1}$. Unlike the previous definition this FoM is sensitive to the off-diagonal components of the covariance matrix as well as the diagonal components. This FoM also decreases in magnitude for surveys that reduce the error on (and the correlation between) the parameters

Designing a FoM which is both robust and easy to use, while not favouring a particular model space or paradigm has been the topic of recent work [1]. Other suggestions have been proposed, such as using principal component analysis [13], as the basis of an effective FoM to compare survey efficiency. We include a module which plots the principal components from standard survey configurations in the latest release, which is discussed in Section **??**.

1.4.5 Generating plots
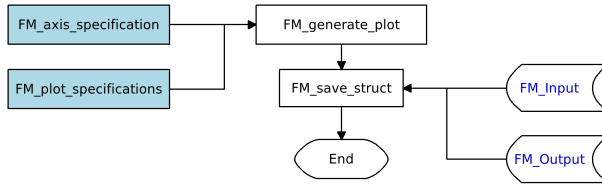
FM_generate_plot calls the either **FM_plot_ellipse.m** or **FM_plot_likelihood.m** depending on whether a $1-$D likelihood or an ellipse is required (whether one or two parameters are specified in the **parameters_of_interest** field in the **input** structure. The style of the plot is controlled by the **FM_plot_specifications.m** file, which controls variables such as the line style, the colour of the lines, the resolution of the grid and the contour level (for example $1 - \sigma, 2 - \sigma$). Similarly the file **FM_axis_specifications.m** controls the x and y labels and the range of the plot that will be generated.

Lastly **FM_save_struct** is called to save the **input** and **output** structures with a user specified filename. One could invoke this function from the command line:

```
>>FM_save_struct('saved_filename',input,output)
```

where **input** and **output** correspond to the structures that are being saved as **saved_filename-01-Nov-2008.mat**. The date on which the structure is saved is appended to the end of the filename. If a filename is not specified then the default name of **FM_saved_data** is used. It is important to note that the function overwrites existing files with the same name and date and no warning is given. Care should thus be taken to ensure different names are specified when saving important data on the same day. The structures can be loaded once again by issuing the following command:

```
>>load('saved_filename-01-Nov-2008')
```

this will make the previously used **input** and **output** structures available in the current session.

To end with we provide a global view of the structure of the code in Figure (1.18).
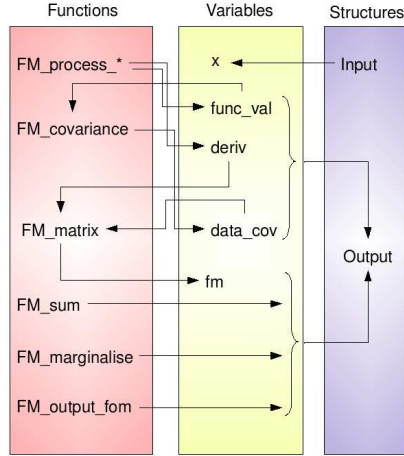
Figure 1.18: Schematic of the relationships between the various functions, variables and structures of the code. The vertical bars indicate the functions, variables and structures respectively while the arrows illustrate how a function may produce a variable and how that variable is in turn stored in a structure (left to right). Alternately the arrows can also show how a variable is retrieved from a structure and (possibly) used in a function. The order from top to bottom shows the chronological order in which the functions, variables and structures are called **FM_run.m**.

1.5 Cosmological Application

1.5.1 The Background Observables

– *Hubble parameter*
The expansion history of the Universe is described by the Hubble parameter, which is defined as

$$H^2(z) = H_0^2 E^2(z) = H_0^2 \left(\Omega_m (1+z)^3 + (1 - \Omega_m - \Omega_k) f(z, w_0, w_a) + \Omega_k (1+z)^2 \right), \quad (1.1)$$

with the evolution of the dark energy density, $f(z)$ given by

$$f(z) = \exp \left(3 \int_0^z \frac{1 + w(z')}{1 + z'} dz' \right). \quad (1.2)$$

Assuming the Chevallier-Polarski-Linder (CPL) expansion of the dark energy equation of state [7, 16]:

$$w(z) = w_0 + w_a \frac{z}{1+z} = w_0 + w_a \left(1 - \frac{a}{a_0}\right),$$ (1.3)

where $a_0 = c/(H_0 \sqrt{|\Omega_k|})$ is the curvature radius of the cosmos, $f(z)$ becomes:

$$f(z) = (1+z)^{3(1+w_0+w_a)} \exp\left\{-3w_a \frac{z}{1+z}\right\}$$ (1.4)

– *Angular diameter distance*
Measurements of 'standard rulers' of known intrinsic length is widely used as a probe of the cosmology of the Universe. The angular diameter distance relates the angular size of an object to its known length to obtain a measure of the distance to the object, and given by

$$d_A(z) = \frac{1}{1+z} \frac{c}{H_0} \frac{1}{\sqrt{\Omega_k}} \cdot \sinh\left(\sqrt{\Omega_k} \chi(z)\right),$$ (1.5)

where

$$\chi(z) = \int_0^z \frac{1}{E(z')} dz',$$ (1.6)

and $E(z)$ is as defined in Eq. (1.1).

– *The Growth of Structure*
The growth of structure is a potentially powerful probe of dark energy [2, 3, 9, 20, 29, 17, 18]. Consider the differential equation for the evolution of perturbations in the matter density δ (assuming the pressure and pressure perturbations of the matter are zero - $p = \delta p = 0$) [23, 27, 19]:

$$\ddot{\delta} + 2H\dot{\delta} = 4\pi G \rho_m \delta$$ (1.7)

The growth function provides the temporal evolution of these density perturbations, i.e. $\delta(\mathbf{x}, z) \propto G(z)$. In Fisher4Cast his is solved in a general FLRW universe, and hence there is in general no analytical solution to Eq. (1.7). Under the assumption of a *flat universe* and a cosmological constant (or pure curvature) however, the growing mode satisfies the following integral form [10, 11]:

$$G(z) = \frac{5\Omega_m E(z)}{2} \int_z^\infty \frac{(1+z')dz'}{E(z')^3},$$ (1.8)

where the 5/2 coefficient is chosen to ensure that $G(z) \to 1/(1+z)$ as $z \to \infty$ This expression should not be used however, to compute the Fisher derivatives $\partial G/\partial \Omega_k, \partial G/\partial w_0$ or $\partial G/\partial w_a$ since all of these derivatives violate the validity of the equation. Instead, the growth derivatives should be computed numerically from the solution of the full differential equation for $\delta(x)$. Rewriting the Raychaudhuri equation in terms of the Friedmann equation and the curvature density allows one to find an equation explicitly showing the curvature and dynamical dark energy contributions to the friction term:

$$G'' + \frac{3}{2}\left(1 + \frac{\Omega_k(x)}{3} - w(x)\Omega_{\text{DE}}(x)\right)\frac{G}{x} - \frac{3}{2}\left(\Omega_m(x)\right)\frac{G}{x^2} = 0,$$ (1.9)

where the new independent variable is $x \equiv a/a_0 = 1/(1+z)$, a_0 is the radius of curvature and $\Omega_k(x) = -k/(a_0^2 x^2 H(x)^2)$ & $\Omega_{\text{DE}}(x) = \rho_{\text{DE}}(x)/\rho_{crit}(x)$ are the fractions of the critical density in curvature and dark energy respectively. Alternatively, this can be written as a differential equation in terms of $\ln(x)$:

$$\frac{d^2 G}{d\ln^2 x} + \frac{3}{2}\left(\frac{1}{3} + \frac{\Omega_k(x)}{2} - w(x)\Omega_{\text{DE}}(x)\right)\frac{dG}{d\ln x} - \frac{3}{2}\Omega_m(x)G = 0,$$ (1.10)

which is the equation actually solved in Fisher4Cast since it is typically more stable numerically. Appropriate initial conditions for this differential equation are set deep in the matter dominated era $G(z_i) = 1, dG/d\ln x(z_i) = G(z_i)$ for $z_i \geq 100$.[∥] Note that as a result, the growth solutions will be unreliable if $w(z \rightarrow \infty) = w_0 + w_a \geq 0$ (or even if it is close to zero from below) since then there will be significant or even dominant early dark energy. Fisher4Cast allows the user to choose the redshift where the growth is normalised to unity. The Fisher derivatives all satisfy $\partial G/\partial \theta_i = 0$ at the normalisation redshift.

1.5.2 Alternative Dark Energy Parametrisations

The Fisher4Cast GUI is hard-coded for three cosmological observables (H, d_A, and G), assuming the Chevallier-Polarski-Linder (CPL) parameterisation [7, 16] with parameters (w_0, w_a) – see Eq. (1.3). This is true of both the functions themselves, and the analytical derivatives included in the Fisher4Cast suite. The general framework of Fisher4Cast, however, means that one is not restricted to this parametrisation. As can be seen from Eq. (1.1), (1.5) and (1.10), the dark energy equation of state enters in the cosmological observables through the evolution of the dark energy, via $f(z)$, defined in Eq. (1.4). Hence for any given $w(z)$ one only needs to specify (in the **input** structure) the names of the functions that will replace the current versions of **FM_function_1.m** (H), **FM_function_2.m** (DA) and **FM_function_3.m** (G). The same is true for the derivatives - either they can be coded analytically for the particular parametrisation of dark energy, or the derivatives will be evaluated numerically from the functions specified in the **input** structure.

As a caveat, the GUI can only be used if the new parametrisation of dark energy still contains only two coefficients. If this is not the case, Fisher4Cast must be run from the command line version.

1.6 Extensions

The general philosophy of Fisher4Cast was to make it as easy as possible to mould and extend to the needs of a general user. In line with this philosophy we have introduced extensions as a means to add functionality and customisation to the existing Fisher4Cast suite. As a design philosophy for future extensions we envisage that extensions do not alter the core functions of the code but rather access the input, output or other modular core-functions. This will enable a large community of contributors to add and make available their own specific extensions while ensuring that the robust design features of the core code remain intact.

An important element in ensuring the success of shared extensions is that all contributors have a good appreciation of the structures used in Fisher4Cast while also documenting and commenting the details of their code thoroughly, including the purpose of the extension, the required input and output produced, which files or structures the extension interacts with from Fisher4Cast and whether it is run from the command line or GUI.

The latest extensions included in this release are listed below.

1.6.1 Obtaining Baryon Acoustic Oscillation Errors from Survey Parameters

Two modules are included that calculate errors on the Hubble parameter and angular diameter distance in BAO surveys characterised in input structures of survey parameters. The provided codes are extensions to Fisher4Cast, and should be placed in the same folder as the main code suite so that they can access the required elements of the Fisher4Cast suite.

[∥]In particular one can compare the results with the growth code available at
http://gyudon.as.utexas.edu/~komatsu/CRL/.

The first of these extensions, **EXT_FF_Blake_etal2005** uses the fitting formulae of Blake *et al.* [5] to calculate the errors on the Hubble parameter and angular diameter distance given certain survey specifications, such as the survey area and the redshifts used for the measurements of H and d_A. These are either given as central redshift bins or as the edges of redshift bins. The galaxy number density is also required, and is expected in units of $10^{-3}\mathrm{Mpc}^{-3}h^3$.

All files associated with this module have the same prefix to identify them as external modules for the fitting formula of Blake *et. al* [5]. The fitting formulae contain coefficients specific to either photometric or spectroscopic surveys, and hence it must be specified which survey one is considering. The input parameters to this module are supplied in an input structure, which is explained in the **EXT_FF_Blake_etal2005_Readme.txt** file. They are:

- Input_survey.base_parameters: The fiducial values of the cosmological parameters are specified here, as $(H_0, \Omega_m, \Omega_K, w_0, w_a)$, in the same way as the input parameters are defined in the main Fisher4Cast code.

- Input_survey.surv_type: This specifies either a photometric or spectroscopic survey by either setting it to 'spec' or 'phot'. This defines which of the sets of coefficients to use in the fitting formulae.

- Input_survey.z_type: This field indicates that the redshift data being used are either given at the 'edge' of the bins or as the 'central' redshifts. If Input_survey.z_type='central' then an additional input for width of the redshift bins, dz, is required.

- Input_survey.area: This field specifies the area of the survey in units of 1000 square degrees.

- Input_survey.n: This gives the number density of galaxies and is measured in units of $10^{-3}\mathrm{h}^3\mathrm{Mpc}^{-3}$

- Input_survey.vecH and Input_survey.vecDA: These entries give the redshift vectors of the survey for the Hubble parameter $H(z)$ and angular diameter distance. $d_A(z)$

- Input_survey.biasH and Input_survey.biasDA : These entries specify the bias for the Hubble parameter and angular diameter distance. These two vectors should have the same length as that of 'Input_survey.vecH' and 'Input_survey.vecDA' respectively.

An example of a default input structure supplied in the file **EXT_FF_Blake_etal2005_Input.m**. The command used to run the Blake *et al.* [5] fitting formula is given as:

```
>>[z_H_central,z_DA_central,vol_H, vol_DA,sigH, sigDA] = ...
>>EXT_FF_Blake_etal2005_Main(Input_survey)
```

Should no input structure be specified when calling the code, the default input (**EXT_FF_Blake_etal2005_Input.m**) is assumed. The module comprises of various smaller modules to compute the oscillation scales in the radial and transverse direction, and hence the errors on the angular diameter distance and Hubble parameter. Figure (1.19) outlines the procedures in the module. The module outputs the following parameters:

- z_H_central: A vector of the central redshifts of the $H(z)$ data bins, related to the Input_survey.vecH fields specified in the input structure.

- z_DA_central: A vector of the central redshifts of the $d_A(z)$ data bins, related to the Input_survey.vecH fields specified in the input structure.

- vol_H: A vector of the volume of the redshift bins for the Hubble parameter.

- vol_DA: A vector of the volume of the redshift bins for the angular diameter distance.

- sigH: A vector of the *fractional* error on the Hubble parameter: σ_H/H. For percentage errors, multiply this by 100.
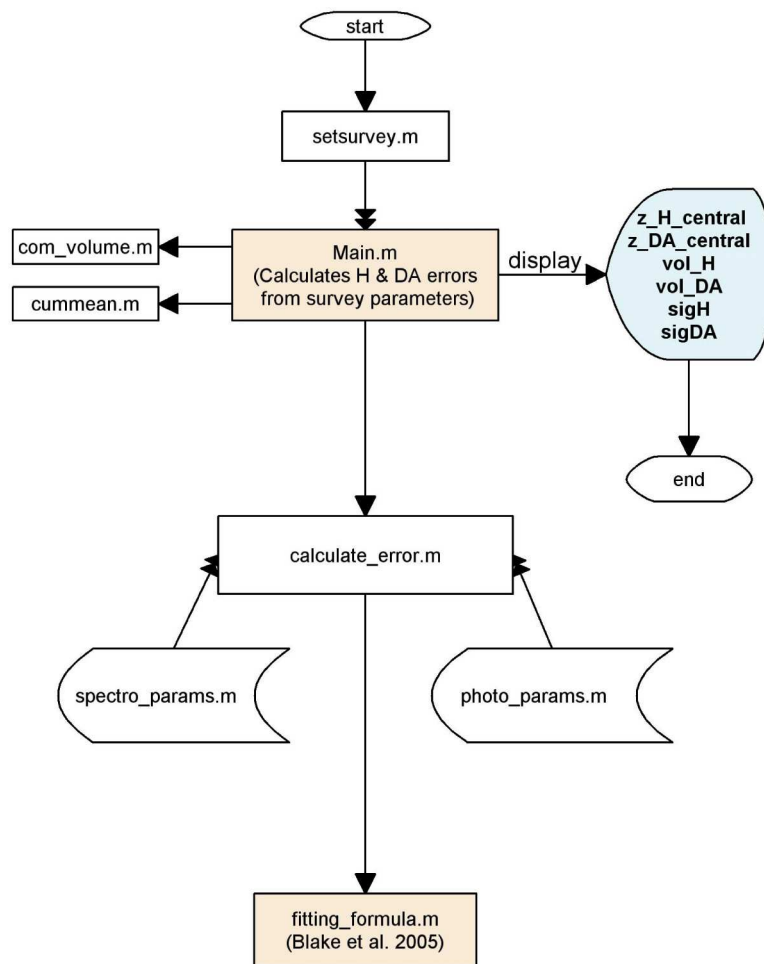
Figure 1.19: The flow of inputs and functions within the Blake *et al.* [5] extension module to Fisher4Cast.

- sigH: A vector of the *fractional* error on the angular diameter distance: σ_{d_A}/d_A. For percentage errors, multiply this by 100.

The module **EXT_FF_SeoEisenstein2007** also computes the errors on the Hubble parameter and angular diameter distance using the prescription set out in [25] and the sound horizon scale as given in [28]. This module contains a wrapper to call the Matlab version of the **C** code of Seo & Eisenstein, which is found online[**]. In this module the code does not need to be in the same directory as the Fisher4Cast suite, and runs completely independently of Fisher4Cast.

The module also takes an input structure (a default input structure with all fields specified is given in **EXT_FF_SeoEisenstein2007_Input.m**) with the following parameters defined:

- **Input_survey.number_density**: The galaxy number density in units of h^3 Mpc^{-3}

- **nput_survey.wmap**: A flag (1 or 3) specifying which WMAP power spectrum to use.

- **Input_survey.sigma8**: The value of σ_8, the amplitude of linear clustering on a scale of 8 Mpc.

- **Input_survey.Sigma_z** The line of sight real mean squared comoving distance error due to redshift uncertainties.

- **Input_survey.beta**: The redshift distortion parameter is entered.

- **Input_survey.volume**: The survey volume in units of h^{-3} Gpc^3.

Sigma_perp (the transverse rms Lagrangian displacement) and Sigma_par (the radial displacement) are calculated and saved to the input structure. The module is comprised of smaller functions; the flowchart of the module is shown in Figure (1.20).

As in the case of the Blake *et al.* module, the Seo and Eisenstein module code can be called from the command line using:

```
>>[Drms,Hrms,r,Rrms] = EXT_FF_SeoEisenstein2007_Main(Input_survey)
```

The outputs of the code are the root mean square error on D/s and Hs, where s is the oscillation scale. These are both given as fractional error, for percentage error multiplied by 100. In addition the correlation coefficient between D and H is given (r), and the diagonal entry in the covariance matrix between D and H.

To be sure that the extensions have access to the functions contained in Fisher4Cast we need to be sure that the extensions are either placed directly in the same folder or the path is specified to both Fisher4Cast and the extensions. One can use the path command to do this:

```
>>path(path,'/path-to-folder/Fisher4Cast-v2.0')
>>path(path,'/path-to-folder/EXT_FF_Blake_etal2005')
```

where the 'path-to-folder' is the path specifying the directory where the extension or Fisher4Cast code is kept on your local computer. Similarly, to run the Seo and Eisenstein [25] module, you will need to ensure that the respective extension is either in the same directory or the path is specified:

```
>>path(path,'/path-to-folder/Fisher4Cast-v2.0')
>>path(path,'/path-to-folder/EXT_FF_SeoEisenstein2007')
```

[**]http://cmb.as.arizona.edu/ eisenste/acousticpeak/bao_forecast.html.
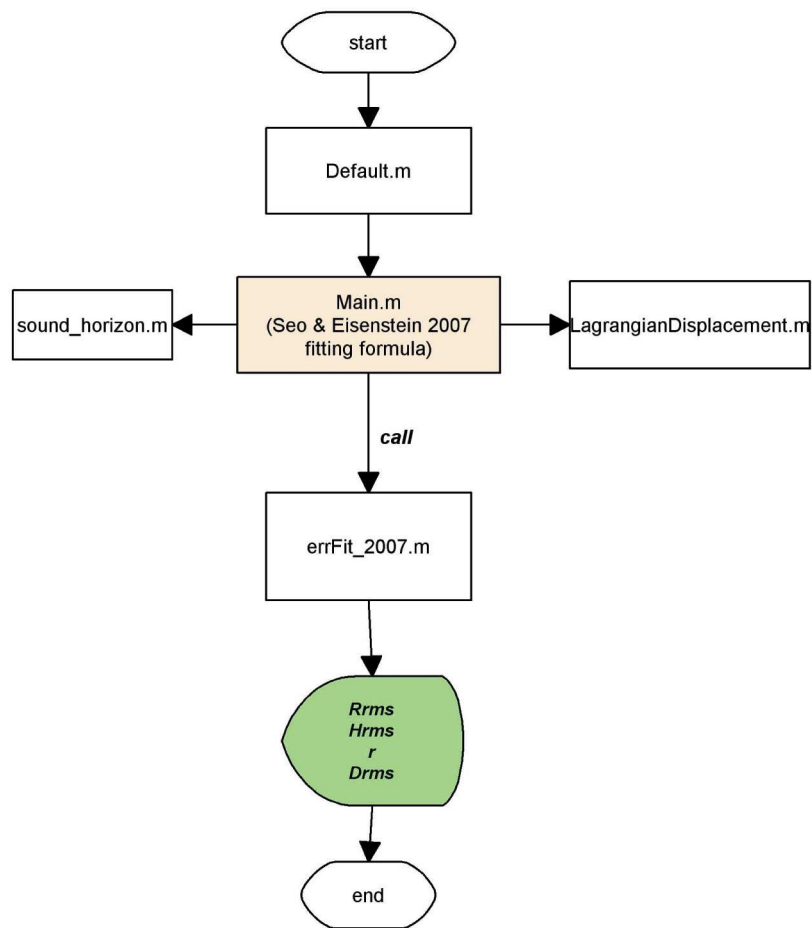
Figure 1.20: Flowchart of the code in the Seo and Eisenstein BAO extension module to Fisher4Cast.

1.6.2 Reporting Features for the Fisher4Cast Suite

Two extension modules have been included to provide reports of the **input** and **output** structures during a run of Fisher4Cast. These reports can either generate an ASCII text file (.txt) or a LaTeX file (.tex) which detail all the input and output produced by Fisher4Cast.

In the case of the LaTeX reporting function the resulting .tex file can be compiled using LaTeX to produce a Postscript file (.ps) or Portable Document Format file (.pdf). This allows for a more polished presentation of the results generated from Fisher4Cast. It also includes a figure of the ellipse or likelihood plot which is embedded in the document. The additional benefit to generating a document in .tex format is that one can cut-and-paste the LaTeX formatted syntax of the figure or any of the tabulated data for easy inclusion of the figure in an article or document containing the results from a run of Fisher4Cast. These

Figure 1.21: The figures on the left show excerpts from the .tex files that was generated from the reporting code while the figures on the right show the extracts of the corresponding .pdf file that was generated from the .tex file.

reporting features are accessible through the Graphical User Interface, by clicking the scroll-down menu bar labelled 'Saving Features' (see Figure (1.12)) for a screenshot of the drop-down menu). This opens a dialog box which in the case of the text report prompts the user for the .txt filename that it should be

saved as. Upon choosing a LATEX report, two dialogue boxes are opened and the user is prompted for the names of both the .tex file and the .eps file.

These two extensions can just as easily be called from the command line. To generate a text report one uses the functions **FM_report_text.m**. The user is required to supply at least an input structure to generate a report. This input structure can either be a default input structures, eg **Cooray_et_al_2004.m**, or a user customised input. The function **FM_report_text.m** then calls **FM_run(input)** with the same supplied input which then produces the relevant output structure. Both the input and output used and generated from Fisher4Cast are then recorded in the report. The user can also specify a filename to saved the report as (if no .txt extension is supplied one will be added automatically). A default name of 'Fisher4Cast_Report-Day-Month-Year.txt' will be used, should no name for the report be specified, where the Day-Month-Year are the date on which the report was generated. For example the command:

```
>>FM_report_text(input,'report_name')
```

will generate a report with the name, 'report_name.txt', as described above. If the same report_name is used, the previous report will be overwritten without warning. Please specify a unique report_name to ensure the report is correctly saved.

Finally there is an option of including a specific output structure in the report function. This is useful when generating the report from the GUI, but care should be taken when using this option in the command line, as one runs the risk of generating a report where the input and output are not appropriately related. In other words,

```
>>FM_report_text(input,'report_name',output)
```

generates a report as before with the name, 'report_name.txt', using the input supplied and *assuming* that the given output is associated with the respective input.

Much the same as the text report, the LATEX report is called using **FM_report_latex.m** and requires at least an input structure. The filename the report is to be saved as can also be specified (either with or without the .tex extension). The LATEX report includes the EPS figure generated from Fisher4Cast: as a default the figure will be saved with the same name as the .tex, except an .eps extension. The default name of 'Fisher4Cast_Report-Day-Month-Year.tex' is used, should no report name be specified The commands

```
>>FM_report_latex(input,'report_name')
```

generate a report with the name, 'report_name.tex', and a figure with the name 'report_name.eps', where the names overwrite any existing files of the same name. Additionally, one can use a specific figure in the report with the command:

```
>>FM_report_latex(input,'report_name','use_fig')
```

In this case there is of course no guarantee that the figure and the output from Fisher4Cast agree.

As in the case of the .txt report, one can specify the output structure directly with:

```
>>FM_report_latex(input,'report_name','use_figure',output)
```

which generates a report as before with the name, 'report_name.tex', using a figure called 'use_figure.eps' where the output and figure are assumed to be associated with the respective input supplied.

1.6.3 FoMSWG

This Fisher4Cast extension is based on the code, "fisher_fomswg.c"[††], written by Dragan Huterer produced for the Joint Dark Energy Mission (JDEM) FoM Science Working Group[‡‡] (FoMSWG). The same default inputs are provided in this extension, as initially included in the original C code, while the outputs and files produced are identical.

In the formulation developed by FoMSWG, Principal Components (PCs) are calculated and used to determine the sensitivity of future surveys, such as JDEM. This allows one to understand the redshift at which each experiment has the greatest power to constrain w. This is achieved, in brief, by taking a 45×45 dimensional Fisher matrix composed of 9 cosmological parameters and 36 piecewise constant equation of state parameters, $w_i \in \{0, 35\}$, which is used as an input for the FoMSWG code to produce 36 PCs and their associated errors. Other useful outputs are also generated after running the code. More details of this, the original code and further suggestions on how to best interpret and plot the results are detailed in [1].

FoMSWG GUI

There is a command line version of the code, EXT_fomswg.m, and a GUI version, EXT_gui_fomswg.m. The GUI version simply calls the command line version with the user specified inputs. The GUI can be launched from the command line:

```
>>EXT_gui_fomswg
```

Alternatively the FoMSWG GUI extension can be accessed from the Fisher4Cast GUI under the 'F4C Extension' menu and is listed as 'FoMSWG Pop-up Extension'.
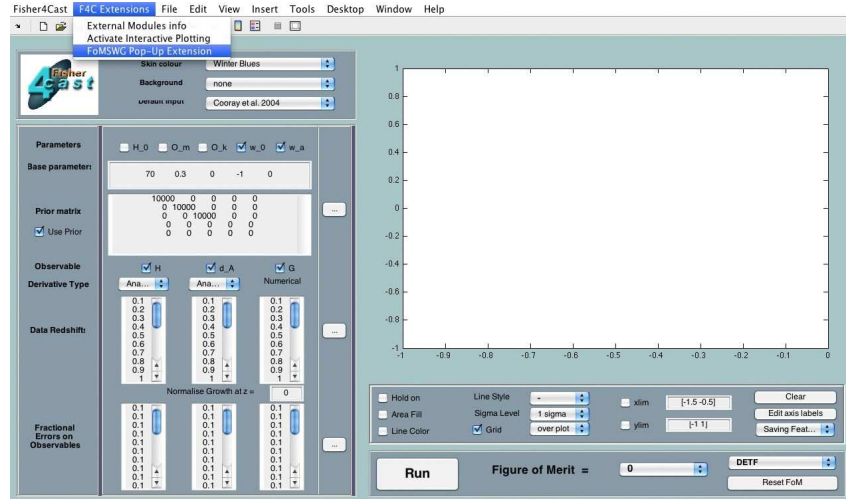
Figure 1.22: The above figure shows how the FoMSWG GUI can be launched from the Fisher4CastGUI menu at the 'F4C Extensions->FoMSWG Pop Up Extension'.

The input radio buttons listed on the left side of the the GUI are the same as supplied in the command line version of the code. There is an additional radio buttons which allows for the inclusion of a user defined Fisher matrix input file. A variety of input formats can be used ranging from the clipboard

---

[††]VERSION 8 December 2008

[‡‡]see http://jdem.gsfc.nasa.gov/science/fomswg/

memory and an Excel spreadsheet to a standard ascii file. The text output fields on the right of the GUI match that which is produced by the output structure of the command line code. The 'Run' button can be pushed to execute the code. The specifics of the inputs and outputs are explained below in the documentation for the command line code.
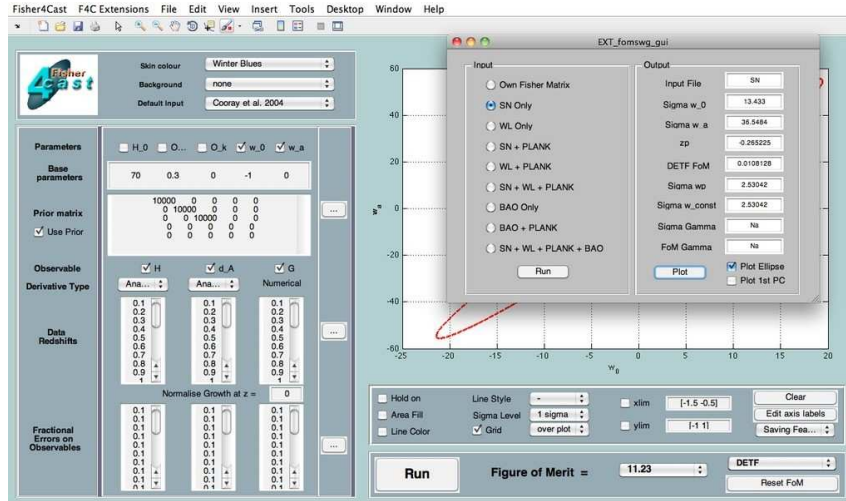

Figure 1.23: The FoMSWG GUI is shown with the Fisher4CastGUI in the background. The input panel is shown on on the left while the right panel shows the output and plot check boxes.

The two check boxes at the bottom right of the GUI allow a combination of plots to be produced. Either just the error ellipse for the DETF FoM or/and the first principal component, PC 1, as a function of redshift. The 'Plot' button is pushed to produce the plot for the associated check boxes selected.

Command Line

To run the extension using a default Fisher matrix as input a single numerical value is required, between 1-8, which specifies the Fisher matrix data file to be used. This mimics the same input options as in the original code and has retained the same option numbering:

```
The choices are:

  SN only          (1)
  WL only          (2)
  SN+PLANCK        (3)
  WL+PLANCK        (4)
  SN+WL+PLANCK     (5)
  BAO only         (6)
  BAO+PLANCK       (7)
  SN+WL+PLANCK+BAO (8)
```

The default files used in the above selection are stored in the DATA/ sub-directory of the main code's path. A user can also select their own 45×45 dimensional Fisher matrix file as an input string by giving the selected file name (and directory, if the file is in a different path to where the code is being run from) as the input variable.

```
>>output = EXT_fomswg('file_name.dat');
```

The order of the variables listed in the Fisher matrix input files are specified as:

0	n_S
1	ω_M
2	ω_B
3	ω_k
4	ω_{DE}
5	$\Delta\gamma$
6	$\Delta\mathcal{M}$
7	$\ln G_0$
8	$\ln\Delta_\zeta(k_*)$
9	w_0
10	w_1
$\vdots$	$\vdots$
44	w_{35}

It is important to note that the numbering convention begins at 0 and not 1. This is the same as stipulated in [1] and its published code. The input file should be written in ascii format with the form,

$$i \quad j \quad \mathcal{F}_{ij}, \tag{1.11}$$

where i and j are the components of the Fisher matrix value, $\mathcal{F}_{ij}$.

Once the code is run, an output structure is produced which contains key data that can easily be accessed for further use and manipulation within Matlab. The output structure is composed of the following fields:

- input_data: This field lists the Fisher matrix input file name which was used. This could be one of the default input files saved in **DATA/** or a user specified file.

- sig_w0: Here the error on the w_0 parameter marginalised over all 9 other parameters.

- sig_wa: As above it specifies the error on w_a parameter also marginalised over all 9 other parameters.

- zp: The pivot redshift, z_p, is given as defined in Eq. 27 in [1].

- Marg_F: The marginalised $2\times$ Fisher matrix is given for the w_0 and w_a parameters.

- DETF_FoM: This field lists the Dark Energy Task Force Figure of Merit (DETF FoM) defined as $(\sigma(w_a) \times \sigma(w_p))^{-1}$.

- sig_wp: Here the uncertainty in the associated pivot equation of state, w_p, is given as defined in Eq. 26 from [1]:

- PC_all: The redshifts and all the PCs are stored here in a 36×37 matrix. The first column lists the redshifts while the remaining columns list the PCs from 1 to 36. This field of the structure can easily be passed to the 'EXT_fomswg_plot.m' file as an input variable. This will plot the first Principal component as a function of redshifts making it easy to identify were the sensitivity of the PC peaks.

$$w_p = w_0 + (1 + a_p)w_a). \tag{1.12}$$

- sig_w_const: An error in constant w, marginalised over all other 9 parameters is assigned to this field.

- sig_gamma: The growth index, γ, is listed and is only given if the user selects options including weak lensing measurements (options 2, 5 and 8 include weak lensing while options 1, 3, 6 and 7 do not) else an 'Na' is displayed.

- FoM_gamma: The γ FoM, defined as $\sigma^{-2}(\delta\gamma)$, is stored here but also only for default choices including WL measurements (options 2,4,5,8). If another option is selected then an 'Na' is shown in this field.

Below is an example where the default input for SN (option 1) is selected. The output structure is assigned to the variable output.

Example:

```
>>output = EXT_fomswg(1)

>>output =
    input_data: 'SN'
         sig_w0: 13.4320
         sig_wa: 36.5445
             zp: -0.2652
         Marg_F: [2x2 double]
       DETF_FoM: 0.0108
         sig_wp: 2.5291
    sig_w_const: 2.5305
      sig_gamma: 'Na'
      FoM_gamma: 'Na'
         PC_all: [36x37 double]
```

Since an input selection was chosen that did not includes weak lensing (WL) the fields for sig_gamma and FoM_gamma do not produce numerical values but rather are given as 'Na' as described previously.

In addition to the output structure, output files are also generated for the given input. These match the same files produced by the C code used for [1]. If a default option is selected which does not include WL, the following files would be produced:

- sigma_SN_22-April-2010.dat : This file contains the errors on the 36 Principal Components (PC). This file lists the case of priors included and not included.

- PC_1234_SN_22-April-2010.dat : The first four eigenvectors are contained in this file.

- PC_all_SN_22-April-2010.dat : All 36 eigenvectors are listed here as a function of 36 redshift bins.

- w0wa_SN_22-April-2010.dat : The w_0 and w_a parameters for a 2×2 Fisher matrix is saved here.

The output files are appended with the Fisher matrix input file used and the date the file was generated on, e.g. the above files are the output for the default SN input (option 1) and was generated on the 22-April-2010. These files are saved to the OUTPUT/ sub-directory of the main code's path. A list of the files generated, after the code is run, are printed to the command window:

```
 PC accuracies (unpriored and priored) are in:

                 OUTPUT/sigma_SN_22-April-2010.dat

 First four PCs (i.e. eigenvectors) are in:

                 OUTPUT/PC_1234_SN_22-April-2010.dat
```

```
  All PCs (i.e. eigenvectors) are in:

                  OUTPUT/PC_all_SN_22-April-2010.dat

  2x2 Fisher matrix for w0-wa is in:

                  OUTPUT/w0wa_SN_22-April-2010.dat
```

If any of the default options are selected that include weak lensing, additional output files are generated with a *_gamma_* added to the the file name, e.g. if WL (option 2) is selected an additional file is generated for all the principal components, PC_all_gamma_WL_22-April-2010.dat.

1.7 How to produce small, good quality postscript images for inclusion in LaTeX documents

Much of the output from Fisher4Cast is expected to be used in research publications produced using LaTeX, in which case Fisher4Cast figures need to be saved in '.eps' or '.ps' format. Unfortunately the default Matlab '.eps' and '.ps' files produced tend to be large, often several MB in size. Apart from making printing slow, this is a problem when submitting papers to online archives, like the arXiv, which has a strict file limit.

Many of the figures in this paper exceeded 1MB after saving from Matlab, and several exceeded 2MB. To solve this, one can instead save the files as bitmaps and then use a utility such as jpeg2ps to convert the '.jpeg' file to postscript with a much smaller file size. Nevertheless, achieving good compression and good quality results can be tricky to achieve with Matlab figures, and we share the steps that have worked well, here.

- Save the files in '.eps' format in Matlab.

- Load the eps file into Photoshop (or equivalent such as GIMP), which will immediately request to rasterize the file and hence request an image size (in cm or inches) and a resolution (in dots per inch, dpi).

- Choose the size to be that which you want the figure to appear in your LaTeX document (e.g. 10cm). As with all bitmaps, choose it to be the physical size you will use in the end since rescaling bitmaps causes blurring and poor quality. 300dpi gives good resolution. The resulting file will be huge in Photoshop, but do not worry since it is only an intermediate step.

- Now save the file as a '.jpeg' with high quality factor, 80% or better.

- Using jpeg2ps or a similar utility, convert the '.jpeg' file to '.eps'. This will add approximately 10% to the jpeg file size due to the postscript wrapper. You should be able to achieve a 10-20 fold reduction in eps to eps file size with only marginal reduction in quality.

A brief word of caution: if you plot many ellipses in the GUI, or with multiple colour variations, and then save the result, Matlab will save the entire history, making the resulting file large. It is best to decide on what combinations you want and then save only that figure.

http://xxx.lanl.gov

This process is discussed at http://aps.arxiv.org/help/bitmap/index and elsewhere.

1.8 Tests of the Code

Various tests were performed to check the correctness and accuracy of all components of Fisher4Cast, namely the integration routines in Fisher4Cast, the derivatives and their validity (especially relevant for the growth function) and the matrix manipulation and generation of Fisher error ellipses.

1.8.1 Integration Tests

The integration routines in Fisher4Cast are tested by comparing them to standard results and fitting formulae for the angular diameter and growth function respectively.

1.8.2 Angular Diameter Distance

The angular diameter distance, $d_A(z)$, defined in Eq. (1.5), is the ratio of an object's physical transverse size to its angular size (in radians). Characteristically it does not increase indefinitely as $z \to \infty$, rather it inverts at $z \sim 1$ – thereafter more distant objects actually appear *larger* in angular size. In the case of $\Omega_\Lambda = 0$, one can write Eq. (1.5) as an analytical function of redshift and cosmic parameters as [30, 23]:

$$d_A(z) = \frac{c}{H_0} \frac{2\left[2 - \Omega_m(1-z) - (2-\Omega_m)\sqrt{1+\Omega_m z}\right]}{\Omega_m^2(1+z)}. \tag{1.13}$$

The angular diameter distances for three particular cosmologies (the same cosmologies as found as Figure (2) of [12]) are shown in Figure (1.24). The same axis lengths and line styles are used for easy comparison of the two plots, which show agreement between the Fisher4Cast algorithms and the know results.
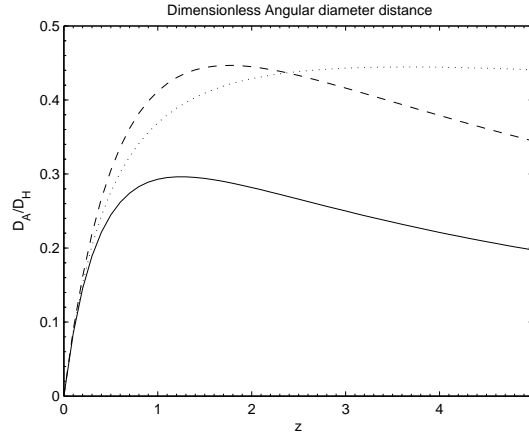

Figure 1.24: **The angular diameter distance -** plotted for Universes with $(\Omega_m, \Omega_\Lambda) = (1, 0)$ (solid), $(0.05, 0)$(dotted) and $(0.2, 0.8)$(dashed line). Our results are in agreement with a similar plot, Figure (2) in [12].

1.8.3 Growth function

In order to test the numerical growth function, the solution is compared to other numerical solutions in the literature (code is available for comparison at http://gyudon.as.utexas.edu/~komatsu/CRL/, which implements the growth equation given as Eq. (76) in [14]), to analytical approximations for particular

cosmologies, such as Eq. (1.8), or to fitting formulae, such as that originally suggested by Carroll, Press & Turner [6]. This fitting formula is given by $G(z) = \frac{g(z)}{1+z}$ [15] where

$$g(z) = 5\frac{\Omega_m(z)}{2}\left(\Omega_m^{4/7}(z) - \gamma(z) + \left[1 + \frac{\Omega_m(z)}{2}\right]\left[1 + \frac{\gamma(z)}{70}\right]\right)^{-1} \tag{1.14}$$

and

$$\begin{aligned}\gamma(z) &= \Omega_{DE}\left[\frac{H_0}{H(z)}\right]^2 \\ &= \frac{\Omega_{DE}}{\Omega_m(1+z)^3 + (1 - \Omega_m - \Omega_{DE})(1+z)^2 + \Omega_{DE}}.\end{aligned}$$

The comparison between the growth function from Fisher4Cast and the fitting formula is shown in the left-hand panel of Figure 1.25, together with the relative difference, $(G - G_{fit})/G$, between the two methods, in the right-hand panel.
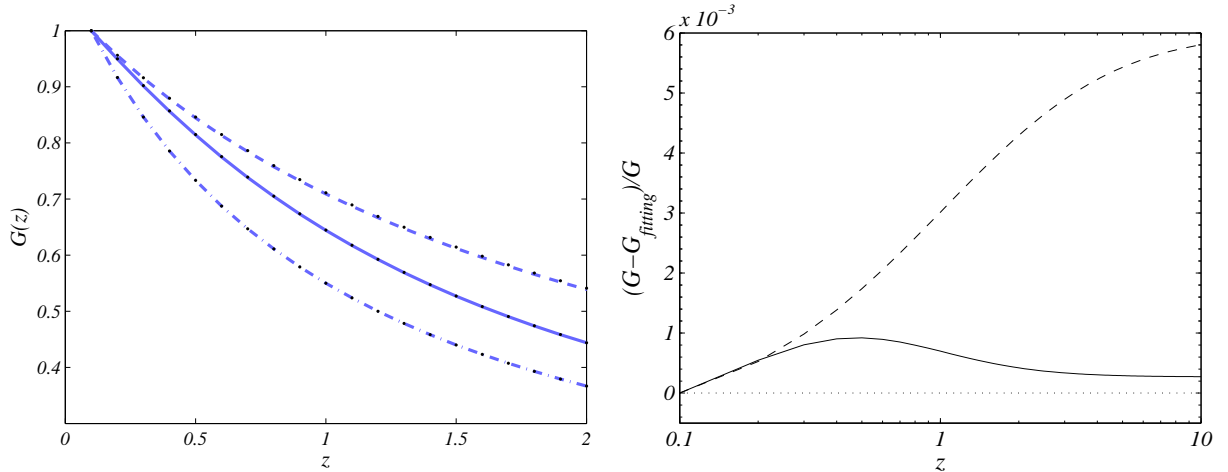

Figure 1.25: **The Fisher4Cast growth function compared to a fitting formula.** The left-hand panel shows the growth function implemented in Fisher4Cast (blue curves) as compared to the fitting formula Eq. (1.14) [22, 15] (black dots). The models represented are $(\Omega_m, \Omega_k) = (0.3, 0.7)$ (dashed), $(0.3, 0)$ (solid) and $(1, 0)$ (dot dashed). The normalised differences between the growth function used in Fisher4Cast and the fitting formula for the various models are of order 10^{-3}, which is shown in the residuals of the right-hand panel.

Degeneracy Tests

The degeneracy direction of a Fisher ellipse (we consider w_0, w_a) is a useful diagnostic of whether or not the Fisher ellipses are being calculated correctly. The direction of degeneracy can be computed analytically for a given redshift by assuming that the specific observable X^α is constant at a particular redshift, and to then solve for w_a as a function of w_0, or by computing the likelihood over a grid of $w_0 - w_a$, and then taking contours of the likelihood that correspond to the same cosmology as that assumed in the run of Fisher4Cast. This is particularly important when considering numerical derivative routines, such as those needed for the growth function.
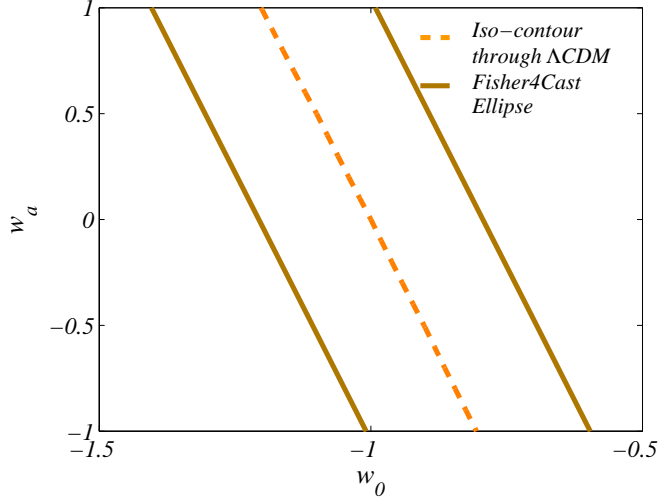
Figure 1.26: **Grid test of the numerical growth derivatives evaluated at a single redshift:** The growth function is evaluated at a single redshift $z = 1$ for a range of models on a grid of $-1.5 < w_0 < 0.5$ and $-1 < w_a < 1$. The $w_0 - w_a$ degeneracy direction in an assumed ΛCDM cosmology is then the iso-growth contour corresponding to ΛCDM ($w_0 = -1, w_a = 0$), shown here as the orange dashed line. The Fisher4Cast degeneracy direction (computed assuming an arbitrary value of 1.5% error on growth) is shown as brown solid lines.

Growth function

As a test of the correctness of the solution (and Fisher derivatives) of Eq. (1.10), the Fisher ellipse from a survey consisting of a single measurement of the growth function at $z = 3$ (as computed with Fisher4Cast) are shown in Figure (1.26), overlaid with contours of the likelihood corresponding to the growth value of the fiducial cosmology. The agreement of the degeneracy directions indicates that numerical computation of both the function and its Fisher derivatives is sound.

Hubble parameter

In the case of the Hubble parameter one can compute the degeneracy direction analytically: consider a single perfect measurement $H(z)$ at some particular redshift z. Solving for w_a in terms of w_0 by substituting Eq. (1.4) into (1.1), yields

$$
\begin{aligned}
C(z) &\equiv \frac{\ln\left(\frac{H^2}{H_0^2} - \Omega_m(1+z)^3 - \Omega_k(1+z)^2\right)}{\ln(1 - \Omega_m - \Omega_k)} \\
&= 3(1 + w_0 + w_a)\ln(1+z) - 3w_a\frac{z}{1+z},
\end{aligned}
\tag{1.15}
$$

hence

$$
w_a = \frac{C(z) - 3(1 + w_0)\ln(1+z)}{3(\ln(1+z) - z/(1+z))}
\tag{1.16}
$$

This degeneracy direction is shown in Figure (1.27) for two different values of the central redshift z along with the ellipses from the ellipses from Fisher4Cast. The agreement between the analytical solution for the degeneracy direction, and the degenerate ellipse from the Fisher4Cast code confirms that the derivatives are being calculated correctly, and the matrix operations within Fisher4Cast are sound.
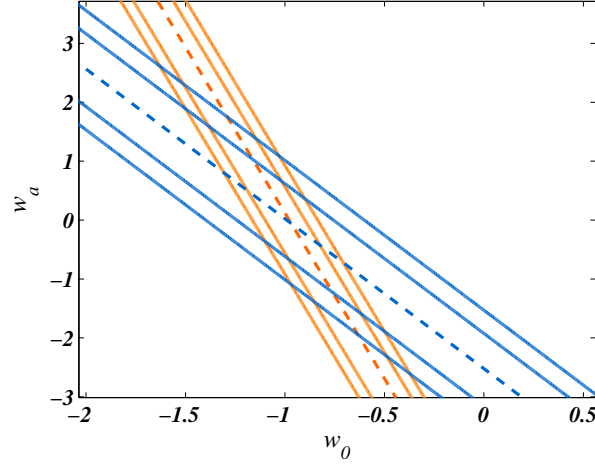
Figure 1.27: **Lines of degeneracy** − comparing the analytical degeneracy directions for the Hubble parameter (long dashed lines) with the degenerate error ellipse produced using Fisher4Cast (solid lines) for two separate surveys of 10% measurements of H, one at $z = 0.5$, and another at $z = 2$. The analytical solution and that of the error ellipse coincide, which is a systematic check of the computations and correctness of Fisher4Cast.

Angular Diameter Distance

Furthermore, it is possible to obtain an analytical solution for the direction of degeneracy between the parameters w_0, w_a from measurements of the angular diameter distance, $d_A(z)$ (Eq (1.5)). The dark energy parameters only enter Eq (1.5) in χ, the integral of $1/E$, hence we need only take derivatives $\partial \chi / \partial w_i$ in order to determine the degeneracy, via:

$$
\begin{aligned}
w_a &= \frac{\partial w_a}{\partial w_0} w_0 + Q \\
&= \frac{\partial \chi / \partial w_0}{\partial \chi / \partial w_a} w_0 + Q,
\end{aligned}
\tag{1.17}
$$

where Q is an arbitrary constant. Figure (1.28) once again confirms the agreement between the degeneracy direction of the ellipse produced using Fisher4Cast and the analytical solution, for two values of the central redshift z.

Combined Degeneracies

Combining measurements from multiple probes is key to tightening constraints on parameters in a cosmological model. Summing over different observables X^α means that one effectively "combines" the different degeneracy directions of each observable together. Given the analytical degeneracy directions discussed earlier, one can investigate (qualitatively at least) how the these degeneracy directions combine to form the Fisher ellipse. This is illustrated for the Hubble parameter and angular diameter distance at a redshift of $z = 0.5$. The orange dashed line shows the degeneracy direction for the angular diameter distance, the degeneracy direction for the Hubble parameter is shown in blue − the combined ellipse in the $w_0 - w_a$ plane for these two observables (black ellipse) is a combination of both degeneracy directions. Qualitatively, the magnitude of the Fisher derivatives for either observable (an indication of how sensitive the observable X^α is to the parameter θ_A) determines how much its degeneracy direction contributes to the final ellipse. This can be used to investigate how the degeneracy directions change with redshift.
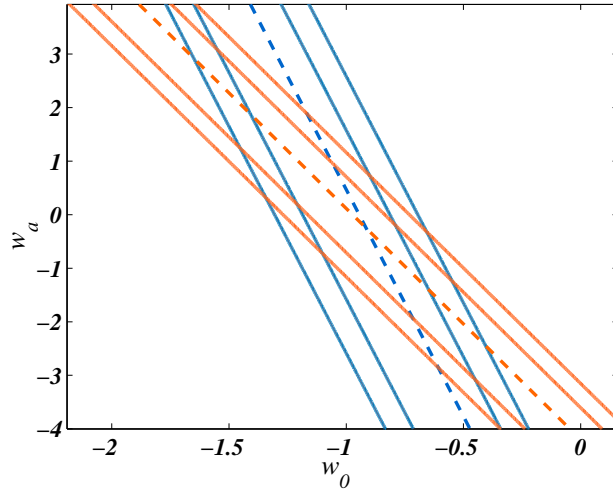
Figure 1.28: **Lines of degeneracy** − similarly to Figure (1.27), the analytical degeneracy directions for the angular diameter distance (long dashed line) are superimposed on the degenerate error ellipses produced from Fisher4Cast (solid lines) for two surveys consisting of one measurement each of d_A with a 10% error at redshifts $z = 0.5$ and $z = 2$ respectively.
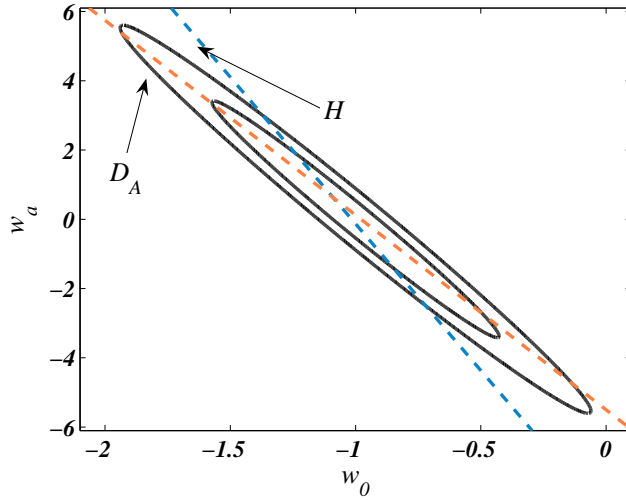

Figure 1.29: **Contributions to Fisher ellipse** The separate degeneracy directions (orange dashed line for d_A and (blue dashed line for H) are plotted along with the Fisher ellipse for $w_0 - w_a$ computed at a redshift of $z = 0.5$, illustrating how the various observables contribute to the overall direction of the Fisher ellipse. One can then employ this strategy at varying redshifts to track how the contributions change as a function of redshift, as the observables become more or less sensitive to the dark energy parameters w_0, w_a.

[1] A. Albrecht, L. Amendola, G. Bernstein, D. Clowe, D. Eisenstein, L. Guzzo, C. Hirata, D. Huterer, R. Kirshner, E. Kolb, and R. Nichol. Findings of the joint dark energy mission figure of merit science working group. *ArXiv e-prints*, January 2009.

[2] A. Albrecht, G. Bernstein, R. Cahn, W. L. Freedman, J. Hewitt, W. Hu, J. Huth, M. Kamionkowski, E. W. Kolb, L. Knox, J. C. Mather, S. Staggs, and N. B. Suntzeff. Report of the Dark Energy Task Force. *ArXiv Astrophysics e-prints*, September 2006.

[3] L. Amendola, C. Quercellini, and E. Giallongo. Constraints on perfect fluid and scalar field dark energy models from future redshift surveys. MNRAS, 357:429–439, February 2005.

[4] Bruce A. Bassett, Yabebal Fantaye, Renee Hlozek, and Jacques Kotze. Fisher matrix preloaded – fisher4cast. 2009.

[5] C. Blake, D. Parkinson, B. Bassett, K. Glazebrook, M. Kunz, and R. C. Nichol. Universal fitting formulae for baryon oscillation surveys. MNRAS, 365:255–264, January 2006.

[6] S. M. Carroll, W. H. Press, and E. L. Turner. The cosmological constant. ARA&A, 30:499–542, 1992.

[7] M. Chevallier and D. Polarski. Accelerating Universes with Scaling Dark Matter. *Int. J. Mod. Phys. D*, 10:213–223, 2001.

[8] A. Cooray, D. Huterer, and D. Baumann. Growth rate of large-scale structure as a powerful probe of dark energy. Phys. Rev. D, 69(2):027301–+, January 2004.

[9] C. Di Porto and L. Amendola. Observational constraints on the linear fluctuation growth rate. *ArXiv e-prints*, 707, July 2007.

[10] D. J. Eisenstein. An Analytic Expression for the Growth Function in a Flat Universe with a Cosmological Constant. *ArXiv Astrophysics e-prints*, September 1997.

[11] D. J. Heath. The growth of density perturbations in zero pressure Friedmann-Lemaitre universes. MNRAS, 179:351–358, May 1977.

[12] D. W. Hogg. Distance measures in cosmology. *ArXiv Astrophysics e-prints*, May 1999.

[13] D. Huterer and G. Starkman. Parametrization of Dark-Energy Properties: A Principal-Component Approach. *Physical Review Letters*, 90(3):031301, January 2003.

[14] E. Komatsu, J. Dunkley, M. R. Nolta, C. L. Bennett, B. Gold, G. Hinshaw, N. Jarosik, D. Larson, M. Limon, L. Page, D. N. Spergel, M. Halpern, R. S. Hill, A. Kogut, S. S. Meyer, G. S. Tucker, J. L. Weiland, E. Wollack, and E. L. Wright. Five-Year Wilkinson Microwave Anisotropy Probe Observations: Cosmological Interpretation. ApJS, 180:330–376, February 2009.

[15] O. Lahav and Y. Suto. Measuring our Universe from Galaxy Redshift Surveys. *Living Reviews in Relativity*, 7:8, July 2004.

[16] E. V. Linder. Exploring the Expansion History of the Universe. *Physical Review Letters*, 90(9):091301–+, March 2003.

[17] E. V. Linder. Cosmic growth history and expansion history. Phys. Rev. D, 72(4):043529–+, August 2005.

[18] E. V. Linder. Extending the gravitational growth framework. Phys. Rev. D, 79(6):063519–+, March 2009.

[19] E. V. Linder and A. Jenkins. Cosmic structure growth and dark energy. MNRAS, 346:573–583, December 2003.

[20] A. Loeb and S. Wyithe. Precise Measurement of the Cosmological Power Spectrum With a Dedicated 21cm Survey After Reionization. *ArXiv e-prints*, 801, January 2008.

[21] Joaquim R. R. A. Martins, Peter Sturdza, and Juan J. Alonso. The complex-step derivative approximation. *ACM Trans. Math. Softw.*, 29(3):245–262, 2003.

[22] J. A. Peacock and S. J. Dodds. Reconstructing the Linear Power Spectrum of Cosmological Mass Fluctuations. MNRAS, 267:1020, April 1994.

[23] P. J. E. Peebles. *Principles of physical cosmology*. Princeton Series in Physics, Princeton, NJ: Princeton University Press, —c1993, 1993.

[24] H.-J. Seo and D. J. Eisenstein. Probing Dark Energy with Baryonic Acoustic Oscillations from Future Large Galaxy Redshift Surveys. ApJ, 598:720–740, December 2003.

[25] H.-J. Seo and D. J. Eisenstein. Improved Forecasts for the Baryon Acoustic Oscillations and Cosmological Distance Scale. ApJ, 665:14–24, August 2007.

[26] M. Tegmark, A. N. Taylor, and A. F. Heavens. Karhunen-Loeve Eigenvalue Problems in Cosmology: How Should We Tackle Large Data Sets? ApJ, 480:22–+, May 1997.

[27] L. Wang and P. J. Steinhardt. Cluster Abundance Constraints for Cosmological Models with a Time-varying, Spatially Inhomogeneous Energy Component with Negative Pressure. ApJ, 508:483–490, December 1998.

[28] Y. Wang. Dark Energy Constraints from Baryon Acoustic Oscillations. ApJ, 647:1–7, August 2006.

[29] Y. Wang. Differentiating dark energy and modified gravity with galaxy redshift surveys. *ArXiv e-prints*, 710, October 2007.

[30] S. Weinberg. *Gravitation and Cosmology: Principles and Applications of the General Theory of Relativity*. Gravitation and Cosmology: Principles and Applications of the General Theory of Relativity, by Steven Weinberg, pp. 688. ISBN 0-471-92567-5. Wiley-VCH , July 1972., July 1972.