ADDING A THIRD NORMAL TO CLUBB

by

Sven Bergmann

A Dissertation Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Mathematics

at

The University of Wisconsin–Milwaukee

May 2024

ABSTRACT

ADDING A THIRD NORMAL TO CLUBB

by

Sven Bergmann

The University of Wisconsin–Milwaukee, 2024
Under the Supervision of Professor Vince Larson

The Cloud Layers Unified By Binormals (CLUBB) model uses the sum of two normal probability density function (pdf) components to represent subgrid variability within a single grid layer of an atmospheric model. This binormal approach, while computationally efficient, restricts the model's ability to capture the full spectrum of potential shapes encountered in real-world atmospheric data.

This thesis proposes to introduce a third normal pdf component strategically positioned between the existing two, significantly enhancing the model's representational flexibility. This trinormal representation allows for a wider range of grid-layer shapes while permitting analytic solutions for certain higher order moments.

The core of this work lies in deriving the necessary mathematical transformations for incorporating the third normal pdf seamlessly into the CLUBB framework. This thesis lists all formulas, inputs, and outputs associated with the extended model as well as gives an outline on how to check those equations. Additionally, it describes certain asymptotic behavior of the trinormal pdf under various parameter settings.

Type dedication here.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

LIST OF ACRONYMS

cas computer algebra system. 16, 47, 48

CLUBB Cloud Layers Unified By Binormals. ii, 1, 2, 4, 5, 11, 18, 26, 29, 30, 46–49

lhs left hand side. 11

pde partial differential equation. 1, 4, 5, 25

pdf probability density function. ii, 1, 6, 8, 10–14, 17, 18, 20, 23–25, 27–30, 40, 41, 46–48

rhs right hand side. 5, 11, 41

LIST OF SYMBOLS

θ_l' Standardized liquid water potential temperature ($\theta_l' = \theta_l - \overline{\theta_l}$). 15

r_t' Standardized total water mixing ratio ($r_t' = r_t - \overline{r_t}$). 15

w' Standardized upward wind ($w' = w - \overline{w}$). 15, 19

1 Introduction

The CLUBB model is a powerful tool used to simulate atmospheric behavior within climate models. This document explores an extension to the current CLUBB framework. Currently, CLUBB utilizes the sum of two normal pdfs to represent a single atmospheric grid layer. While effective, this approach limits the model's ability to capture the full spectrum of potential cloud layer shapes. This work proposes an innovative solution: incorporating a third normal pdf into the CLUBB framework. This addition aims to enhance the model's representational capabilities while maintaining computational efficiency and numerical stability. To achieve this, the document dives into the details of the proposed method.

We begin by outlining the core problem we aim to address (chapter 2) by starting with the motivation, proceeding with a short explanation on how to close turbulence partial differential equations (pdes) (section 2.2) and explaining how we derive the transformation from the formulas given by the paper "Using probability density functions to derive consistent closure relationships among higher-order moments" by Larson and Golaz (section 2.3). After that, we define the goal of this thesis (section 2.4), talk about the inputs and outputs (section 2.5) and provide steps for checking those formulas (section 2.6).

Following this motivational chapter, we establish a foundation with clear definitions of the relevant concepts, including normal distributions and the thermodynamic scalars crucial for atmospheric modeling (chapter 3).

Chapter 4 forms the heart of this work, presenting the actual formulas associated with the extended CLUBB model. This chapter details the introduction of the third normal pdf (section 4.1) and the derivation of key moments within the model (section 4.3 - section 4.4).

Additionally, section 4.6 proposes a diagnostic approach to account for the skewness of heat and moisture, while section 4.7 introduces analytic closure relations for higher-order moments based on the newly formed mixture of three normal distributions.

To handle the mathematical integrations required by the model, chapter 5 explores both exact parametric and numerical integration techniques of verifying the integrals, utilizing the SymPy library (section 5.1 & section 5.2).

Finally, chapter 6 investigates the asymptotic behavior of the extended model, providing valuable insights into its performance under various conditions.

Having talked about the trinormal representation within the CLUBB model, chapter 7 provides a concise recap of the key findings. This summary serves as a comprehensive overview of the essential concepts explored throughout this thesis.

The document concludes with an outlook in chapter 8, outlining potential future directions for research and exploration based on the findings presented here.

2 Problem

2.1 Motivation to add a third normal component

As is said in chapter 1, we try to describe more possible shapes by adding a third normal component. To illustrate that, we plot some of the shapes which are now possible with three normals but were not possible with only two. To be able to draw those plots, we are just using two variables, w, the upward wind, and θ_l, the liquid water potential temperature. To show how the binormal model handles strong winds, let us consider a scenario with a strong updraft at w_1, as well as a strong downdraft at w_2. The way the current binormal model would handle this could look like figure 2.1. However, this binormal distribution (figure 2.1)

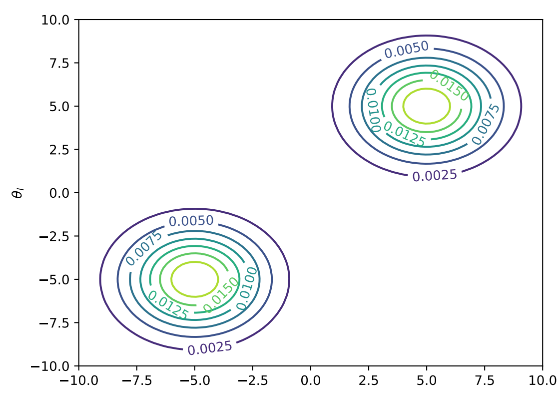

Figure 2.1: Binormal plot for two strong up-/downdrafts
$w_1 = 5$, $w_2 = -5$, $\theta_{l1} = 5$, $\theta_{l2} = -5$, $\alpha = 0.5$, $\sigma_w = 2$, $\sigma_{\theta_{l1}} = 2$, $\sigma_{\theta_{l1}} = 2$.

does not accurately reflect reality. In nature, we would not expect such strong bi-modality between the strong up- and downdrafts at w_1 and w_2. There would most likely be some weaker drafts present in-between. The current binormal model can attempt to capture this smoother transition by simply increasing the standard deviations of both wind, and liquid water potential temperature distributions. This results in a broader distribution with a connection between the two peaks, as shown in figure 2.2. Seeing figure 2.2, the issue with

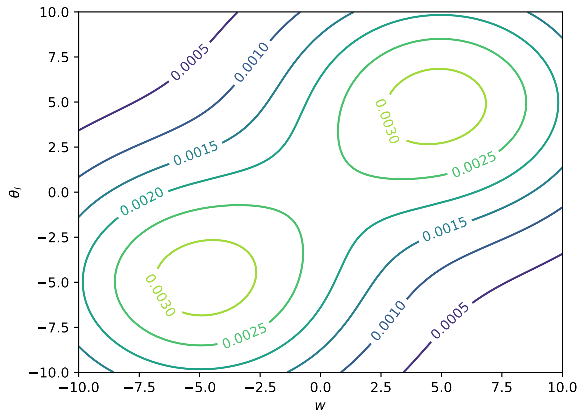

Figure 2.2: Binormal plot for two strong up-/downdrafts with increased standard deviations $w_1 = 5$, $w_2 = -5$, $\theta_{l1} = 5$, $\theta_{l2} = -5$, $\alpha = 0.5$, $\sigma_w = 5$, $\sigma_{\theta_{l1}} = 5$, $\sigma_{\theta_{l1}} = 5$.

having some values in the middle is mitigated but the general width of the normals was increased, too. Since CLUBB also has the simplification that there is no correlation between w and θ_l, and w and r_t – obviously – one cannot just increase it. Therefore, the idea is to add this third normal distribution, which actually has correlation between all three variables and especially in the bivariate case, between w and θ_l. Figure 2.2 would then change to figure 2.3. Now, one can easily model something like the described shape, as illustrated in figure 2.3. Also, some other (maybe weird) shapes are now possible, just like the one in figure 2.4.

2.2 Closing turbulence pdes by integration over a pdf

The CLUBB model relies on a set of pdes to represent atmospheric processes. These equations require closure, implying the expression of all terms solely in terms of known quantities. This closure process often involves integrals, and verifying their analytical solutions

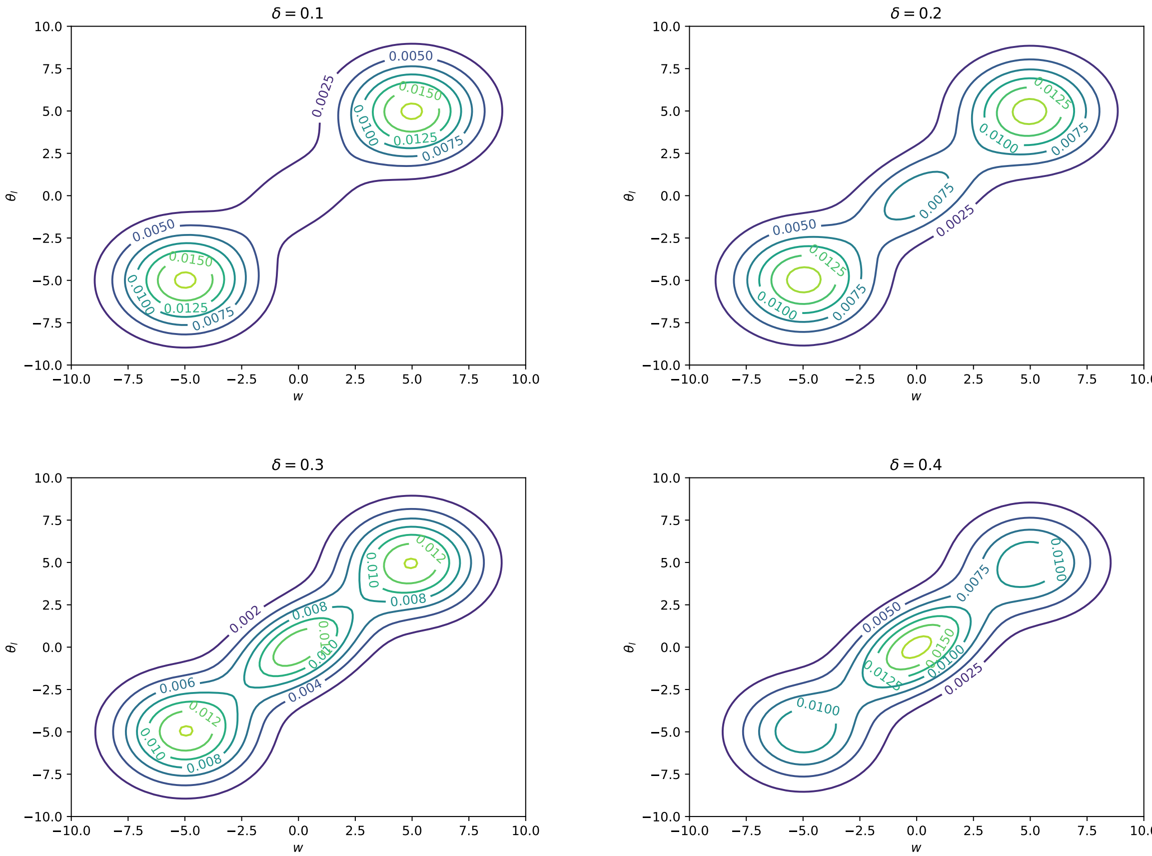

Figure 2.3: Trinormal plot for two strong up-/downdrafts with varying δ
$w_1 = 5$, $w_2 = -5$, $\theta_{l1} = 5$, $\theta_{l2} = -5$, $\alpha = 0.5$, $\sigma_w = 2$, $\sigma_{\theta_{l1}} = 2$, $\sigma_{\theta_{l2}} = 2$, $\sigma_{w3} = 2$, $\sigma_{3\theta_l} = 2$, $\rho_{w\theta_l} = 0.5$.

ensures the model's mathematical integrity. For instance, consider the following prognostic pde [Lar22, p. 21]:

$$\frac{\partial \overline{w'\theta_l'}}{\partial t} = -\overline{w}\frac{\partial \overline{w'\theta_l'}}{\partial z} - \frac{1}{\rho_s}\frac{\partial \rho_s \overline{w'^2\theta_l'}}{\partial z} - \overline{w'^2}\frac{\partial \overline{\theta_l'}}{\partial z} - \overline{w'\theta_l'}\frac{\partial \overline{w}}{\partial z} + \ldots$$

While the details of initial and boundary conditions are essential for formally closing the prognostic equations in CLUBB (omitted for brevity), this section emphasizes the importance of efficiently calculating moments on the right hand side (rhs) of these equations. That is because the model steps forward in time and therefore needs the repeated calculation of moments for each unclosed prognostic equation at every time step. Those already expensive computational steps need to use mathematically simpler moment representations

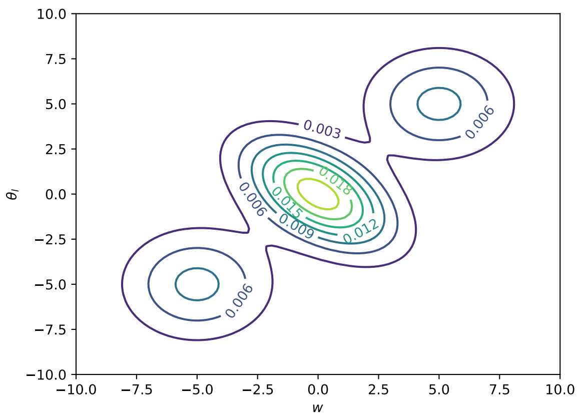

Figure 2.4: Trinormal plot for two strong up-/downdrafts with a third peak in the middle $w_1 = 5$, $w_2 = -5$, $\theta_{l1} = 5$, $\theta_{l2} = -5$, $\alpha = 0.5$, $\delta = 0.5$, $\sigma_w = 2$, $\sigma_{\theta_{l1}} = 2$, $\sigma_{\theta_{l2}} = 2$, $\sigma_{w3} = 2$, $\sigma_{\theta_l 3} = 2$, $\rho_{w\theta_l} = 0.5$.

of e.g. $\overline{w'^2\theta_l'}$, even if they introduce slight limitations in capturing the full variability of the underlying atmospheric state.

2.3 Derivation of trinormal closures by transformation of binormal closures

Analytic closures between higher and lower order moments for the binormal case are available (see CLUBB-SILHS[Lar22]). We wish to derive similar analytic closures for the proposed trinormal pdf. Deriving an analytic closure for a general trinormal pdf is difficult. However, doing so is tractable in the special case that the third normal component is located at the mean of the binormal pdf. In fact, the trinormal closures can be derived by making a simple transformation of the binormal closures [LG05], e.g.

$$\overline{w'^2} = \alpha[(w_1 - \overline{w})^2 + \sigma_w^2] + (1 - \alpha)[(w_2 - \overline{w})^2 + \sigma_w^2]. \tag{2.3.1}$$

This section will demonstrate that the following transformations, denoted by the subscript "dGn" for the binormal case, successfully achieve this conversion.

$$\overline{w'^2}\frac{1-\delta\lambda_w}{1-\delta} = \overline{w'^2}_{dGn}, \tag{2.3.2}$$

$$\overline{w'^3}\frac{1}{1-\delta} = \overline{w'^3}_{dGn}, \tag{2.3.3}$$

$$\frac{\overline{w'^3}}{\overline{w'^2}^{3/2}}\frac{(1-\delta)^{1/2}}{(1-\lambda_w\delta)^{3/2}} = \frac{\overline{w'^3}_{dGn}}{\overline{w'^2}^{3/2}_{dGn}}, \tag{2.3.4}$$

$$\overline{\theta_l'^2}\frac{1-\delta\lambda_\theta}{1-\delta} = \overline{\theta_l'^2}_{dGn}, \tag{2.3.5}$$

$$\overline{w'\theta_l'}\frac{1-\delta\lambda_{w\theta}}{1-\delta} = \overline{w'\theta_l'}_{dGn}, \tag{2.3.6}$$

$$\left(\overline{w'^4} - 3\delta\lambda_w^2\left(\overline{w'^2}\right)^2\right)\frac{1}{1-\delta} = \overline{w'^4}_{dGn} \tag{2.3.7}$$

$$\left(\frac{\overline{w'^4}}{(\overline{w'^2})^2} - 3\delta\lambda_w^2\right)\frac{1-\delta}{(1-\lambda_w\delta)^2} = \frac{\overline{w'^4}_{dGn}}{(\overline{w'^2}_{dGn})^2} \tag{2.3.8}$$

To get a sense of what those transformations mean and why they should work, we pick e.g. equation (2.3.2). If we substitute in the already defined formula for λ_w (equation (4.1.4)), we get

$$\begin{aligned}
\overline{w'^2}\left(1-\delta\frac{\sigma_{w3}^2}{\overline{w'^2}}\right) &= (1-\delta)\overline{w'^2}_{dGn} \\
\overline{w'^2} - \delta\sigma_{w3}^2 &= (1-\delta)\overline{w'^2}_{dGn} \\
\overline{w'^2} &= \overline{w'^2}_{dGn} - \delta\overline{w'^2}_{dGn} + \delta\sigma_{w3}^2 \\
\overline{w'^2} &= \overline{w'^2}_{dGn} - \delta\left(\overline{w'^2}_{dGn} - \sigma_{w3}^2\right).
\end{aligned} \tag{2.3.9}$$

Our analysis reveals a key relationship between the parameter δ and the overall variance (often referred to as "width") of the trinormal distribution. As the value of δ approaches 1 (but strictly remains less than 1), the standard deviation of the third normal distribution has a progressively stronger influence on the overall variance of the combined distribution. This intuitively makes sense because a larger weight assigned to the third normal distribution through δ will contribute more significantly to the spread of the combined pdf.

Also, if we look at equation (2.3.3), we see that there is no more λ_w present. It makes sense graphically, that as δ grows, which means that the normal pdf in the middle is growing, the overall skewness of all three normals has to change also, depending on the value of σ_{w3}. We can see this, as well as the relationship between the variance in table 2.1. Table 2.1 offers a visual representation of how the parameter δ influences the shape of the trinormal distribution. Each plot illustrates pdfs for different combinations of σ_{w3} (standard deviation of the third normal distribution) and δ. The row values in the table correspond to σ_{w3}, while the column values represent δ. We can observe two key trends within these plots:

1. *Influence of σ_{w3}:* As expected, varying σ_{w3} primarily affects the "width" or overall variance of the combined distribution. When σ_{w3} is larger than the width of the original binormal sum (orange/red line), choosing a larger δ allows the overall variance to increase significantly, as predicted by equation (2.3.2).

2. *Decreasing skewness with increasing δ:* The plots also reveal a distinct relationship between δ and the skewness of the resulting distribution. As δ increases, the skewness of the combined trinormal distribution (green line) progressively reduces. This phenomenon can be attributed to the placement of the third normal distribution. Placed directly between the two original normal distributions, the third normal distribution acts as a centralizing force. As the weight of the third normal distribution (controlled by δ) grows, its symmetric nature counteracts the potential skewness of the initial binormal sum. This effect is particularly strong in the bottom three plots, where a larger value of $\sigma_{w3} = 10$ is used. We observe a clear reduction in the skewness of the

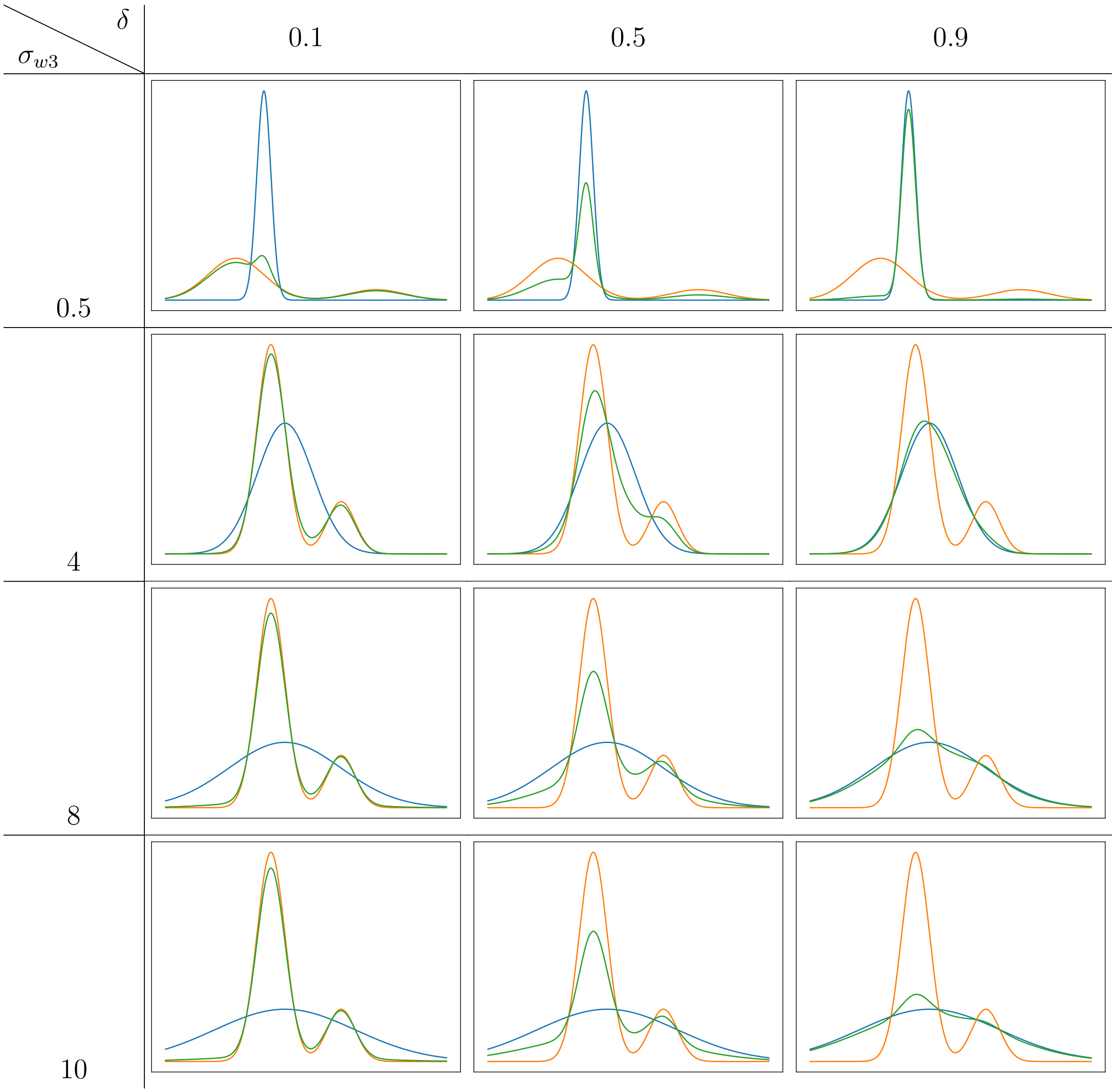

Table 2.1: 1D Plots for different δ and σ_{w3}
$w_1 = 5$, $w_2 = -5$, $\alpha = 0.2$, $\sigma_w = 2$. The blue plot represents the third normal, the orange/red one represents the binormal, and the green one represents the mixture. The x and y labels and ticks are omitted for clarity.

green plot (mixture) as δ approaches 1.

2.4 Goal of this thesis

The goal of this thesis is to verify closure for higher-order moments, such as the third order moment $\overline{w'^2\theta_l'}$ and others like it. This closure is achieved by expressing these higher-order moments – i.e. equation (4.7.3), equation (4.7.5), and equation (4.7.7) – analytically in terms of readily calculable lower-order moments. This analytic approach uses the relationships between moments within a normal distribution, enabling efficient model updates during the time-stepping process.

2.5 Inputs and outputs of the verification procedure

While defining inputs and outputs can seem challenging at first glance, it is a crucial step towards understanding a system.

2.5.1 Inputs and outputs of a forward run

When forecasting the weather ("forward run"), the code provides us with a set of moment terms: $\overline{w}$, $\overline{w'^2}$, $\overline{w'^3}$, $\overline{\theta_l}$, $\overline{w'\theta_l'}$, $\overline{r_t}$, $\overline{w'r_t'}$, $\overline{\theta_l'^2}$, $\overline{r_t'^2}$, $\overline{r_t'\theta_l'}$. These are the inputs. From these inputs, we want to determine certain parameters which describe the shape of the underlying pdf. Those pdf parameters are standardized and some also normalized. So we try to solve these pdf parameters (13), namely α, $\widehat{w}_1$, $\widehat{w}_2$, $\tilde{\theta}_{l1}$, $\tilde{\theta}_{l2}$, $\tilde{r}_{t1}$, $\tilde{r}_{t2}$, $\tilde{\sigma}_w$, $\tilde{\sigma}_{\theta_{l1}}$, $\tilde{\sigma}_{\theta_{l2}}$, $\tilde{\sigma}_{r_{t1}}$, $\tilde{\sigma}_{r_{t2}}$, and $r_{r_t\theta_l}$. All the formulas are listed in chapter 4. Ultimately, the code needs to express even higher order moments such as $\overline{w'^2\theta_l'}$ in terms of the lower order moments. These higher order moments are the outputs in the "forward run".

2.5.2 Inputs and outputs of a backward run (verification direction)

Although a "forward run" models the higher order moments in terms of the lower order moments, we want to verify these formulas, namely equation (4.7.3), equation (4.7.5), and equation (4.7.7). To achieve this, we will take a more traditional approach, working in the "backward" direction. This means we will:

1. *Specify the pdf parameters:* Start by explicitly defining the parameters that characterize the underlying pdf.

2. *Calculate the moments:* Once the pdf is defined, we can then calculate the desired moments, such as $\overline{w}$, through integration.

This can be done, e.g. by calculating the integral:

$$\overline{w} = \int_{\mathbb{R}} \int_{\mathbb{R}} \int_{\mathbb{R}} w \cdot P_{tmg} \, dw dr_t d\theta_l, \tag{2.5.1}$$

where P_{tmg} (**T**rivariate **M**ixture of **G**aussians) is the pdf of the sum of all three normal distributions. Since some integrals are challenging to verify symbolically with SymPy, we are using the quadrature method of SymPy to calculate the integrals and choose arbitrary values for the inputs. All of this can be seen in section 5.2.

2.6 Steps for checking the formulas

This section outlines a general approach for verifying all of those integral expressions employed within the CLUBB model. This approach ensures the accuracy of the computed moment relationships. Chapter 5 discusses some actual examples. We verify the expressions using the following method where the order is crucial. We always want to check if left hand side (lhs) equals rhs:

1. Choose *dimensional* parameters (parameters without any tilde or hat) that determine the pdf, i.e. choose dimensional pdf parameters, e.g. σ_{w3}. Then the pdf is known and any moments of it can be calculated by integration.

2. Calculate the means, e.g. $\overline{w} = \mathbb{E}[w]$ by integration over the pdf. The formula for $\overline{w}$ (equation (4.3.1)) in terms of the pdf parameters can be checked.

3. Once the means are known, we calculate the central variances, e.g. $\overline{w'^2} = \overline{(w - \overline{w})^2}$ by integration over the pdf. The formula for $\overline{w'^2}$ (equation (4.3.3)) in terms of the pdf parameters can be checked.

4. Once the variances, e.g. $\overline{w'^2}$, are known, then the *non-dimensional* pdf parameters such as λ_w (equation (4.1.4)) can be calculated by their definitions.

5. We can also calculate the covariances by 2D integration over a 2D pdf. Again, our formulas in terms of pdf parameters can be checked.

6. Finally, we can calculate the higher order moments, i.e. $\overline{w'^4}$ (equation (4.7.3)) or $\overline{w'^2\theta_l'}$ (equation (4.7.5)) or $\overline{w'\theta_l'^2}$ (equation (4.7.7)), by integration over the pdf.

3 Definitions

For better understanding of the topics covered in this thesis, it follows a brief introduction of all formulas and terms used.

3.1 Normal distribution

We say that a random variable X is distributed according to a normal distribution ($X \sim \mathcal{N}(\mu, \sigma^2)$) when it has the following pdf:

Definition 1 (pdf of a normal distribution)

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right). \tag{3.1.1}$$

3.1.1 Multivariate normal distribution

We say that a random vector $\boldsymbol{X}$ $(r \times r)$ is distributed according to a multivariate normal distribution when it has the following joint density function[Ize08, p. 59]:

Definition 2 (pdf of a multivariate normal distribution)

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{r}{2}} \left|\boldsymbol{\Sigma}\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})\right), \boldsymbol{x} \in \mathbb{R}^r, \tag{3.1.2}$$

where

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_r \end{pmatrix} \in \mathbb{R}^r \tag{3.1.3}$$

is the mean vector, and

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \rho_{13}\sigma_1\sigma_3 & \dots & \rho_{1r}\sigma_1\sigma_r \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 & \rho_{23}\sigma_2\sigma_3 & \dots & \vdots \\ \rho_{13}\sigma_1\sigma_3 & \rho_{23}\sigma_2\sigma_3 & \sigma_3^2 & \dots & \vdots \\ \vdots & \dots & \dots & \ddots & \vdots \\ \rho_{1r}\sigma_1\sigma_r & \dots & \dots & \dots & \sigma_r^2 \end{pmatrix} \in \mathbb{R}^{r\times r} \tag{3.1.4}$$

is the (symmetric, positive definite) covariance matrix. This is also often expressed as $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, meaning that $\boldsymbol{X}$ (r $\times$ r random vector) is distributed according to a multivariate normal distribution with the given parameters.

3.1.2 Moments

Especially for this thesis, we are interested in the moments of the given multivariate normal distribution. We can express the first order moment as the mean, denoted as $\overline{X} = \mathbb{E}[X]$, where X is a random variable. The second order moment is $\mathbb{E}[X^2]$, also denoted as the variance if it is a central moment. The standardized third and fourth order moments have special names, so-called skewness and kurtosis respectively. We denote this by the following:

$$\mathbb{E}[X^3] = \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mu_3}{\sigma^3} = \frac{\mathbb{E}[(X-\mu)^3]}{(\mathbb{E}[(X-\mu)^2])^{3/2}}, \tag{3.1.5}$$

$$\mathbb{E}[X^4] = \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{\mathbb{E}[(X-\mu)^4]}{(\mathbb{E}[(X-\mu)^2])^2} = \frac{\mu_4}{\sigma^4}. \tag{3.1.6}$$

3.2 Variates of the pdf

We denote the variates of the pdf by w, r_t, and θ_l, where w is the upward wind, r_t is the liquid water potential temperature and θ_l is the liquid water potential temperature [Lar22,

p. 10]. Variables denoted by w' are defined by $w - \overline{w}$ where $\overline{w}$ is the mean of w over the whole pdf. We define r_t' and θ_l' in the same way.

4 Formulas that define the shape of the pdf and moments in terms of pdf parameters

This chapter lists all formulas which are derived from the binormal model to this model with an additional normal. All formulas listed are either tested by using a computer algebra system (cas) and calculating the integrals analytically with ranges $-\infty$ to ∞ or using the quadrature procedure with large enough ranges such that the error is (numerically) zero. Those two procedures are explained in chapter 5.

4.1 Definition of the trinormal distribution, P_{tmg}

We would like to add a third normal to the already existing two trivariate normals, which is placed right in the middle between those two. For our proposed mixture of normals we then have

$$P_{tmg}(w, \theta_l, r_t) = \alpha(1-\delta)\mathcal{N}(\mu_1, \Sigma_1) + (1-\alpha)(1-\delta)\mathcal{N}(\mu_2, \Sigma_2) + \delta\mathcal{N}(\mu_3, \Sigma_3), \qquad (4.1.1)$$

where $\mathcal{N}$ denotes the multivariate normal distribution, $\alpha \in (0,1)$ is the mixture fraction of the binormal, and $\delta \in [0,1)$ is the weight of the third normal. The mean vectors and the covariance matrices are defined in the following.

We define the mean vectors of the first and second normal distributions as $\mu_1 = (w_1, \theta_{l1}, r_{t1})^\top$, and $\mu_1 = (w_2, \theta_{l2}, r_{t2})^\top$, where $w_1 > w_2$ (due to a convention in the code) and the covariance matrices as

$$\Sigma_1 = \begin{pmatrix} \sigma_w^2 & 0 & 0 \\ 0 & \sigma_{\theta_l 1}^2 & \rho_{\theta_l r_t}\sigma_{\theta_l 3}\sigma_{r_t 3} \\ 0 & \rho_{\theta_l r_t}\sigma_{\theta_l 3}\sigma_{r_t 3} & \sigma_{r_t 1}^2 \end{pmatrix}, \text{ and } \Sigma_2 = \begin{pmatrix} \sigma_w^2 & 0 & 0 \\ 0 & \sigma_{\theta_l 2}^2 & \rho_{\theta_l r_t}\sigma_{\theta_l 3}\sigma_{r_t 3} \\ 0 & \rho_{\theta_l r_t}\sigma_{\theta_l 3}\sigma_{r_t 3} & \sigma_{r_t 2}^2 \end{pmatrix}. \tag{4.1.2}$$

It might be of interest that there is no correlation between w and θ_l or w and r_t. That is to make the pdfs mathematically more tractable which also makes the pdf family less general. This is not the case for the third normal, though.

It has already been said, that we would like to place the third normal right at the mean, therefore μ_3 and Σ_3 are defined as

$$\mu_3 = \begin{pmatrix} \overline{w} \\ \overline{\theta_l} \\ \overline{r_t} \end{pmatrix}, \text{ and } \Sigma_3 = \begin{pmatrix} \sigma_{w3}^2 & \rho_{w\theta_l 3}\sigma_{w3}\sigma_{\theta_l 3} & \rho_{wr_t 3}\sigma_{w3}\sigma_{r_t 3} \\ \rho_{w\theta_l 3}\sigma_{w3}\sigma_{\theta_l 3} & \sigma_{\theta_l 3}^2 & \rho_{\theta_l r_t 3}\sigma_{\theta_l 3}\sigma_{r_t 3} \\ \rho_{wr_t 3}\sigma_{w3}\sigma_{r_t 3} & \rho_{\theta_l r_t 3}\sigma_{\theta_l 3}\sigma_{r_t 3} & \sigma_{r_t 3}^2 \end{pmatrix}. \tag{4.1.3}$$

The advantage over just two normal pdfs is that we can now express a greater variety of shapes. We also define some additional relationships for this third normal distribution.

$$\lambda_w \equiv \frac{\sigma_{w3}^2}{\overline{w'^2}}, \quad \lambda_\theta \equiv \frac{\sigma_{\theta_l 3}^2}{\overline{\theta_l'^2}}, \quad \lambda_r \equiv \frac{\sigma_{r_t 3}^2}{\overline{r_t'^2}}, \tag{4.1.4}$$

$$\lambda_{\theta r} \equiv \frac{\rho_{\theta_l r_t}\sigma_{\theta_l 3}\sigma_{r_t 3}}{\overline{r_t'\theta_l'}}, \quad \lambda_{w\theta} \equiv \frac{\rho_{w\theta_l}\sigma_{w3}\sigma_{\theta_l 3}}{\overline{w'\theta_l'}}, \quad \lambda_{wr} \equiv \frac{\rho_{wr_t}\sigma_{w3}\sigma_{r_t 3}}{\overline{w'r_t'}}. \tag{4.1.5}$$

Hence, we can rewrite Σ_3 as

$$\Sigma_3 = \begin{pmatrix} \sigma_{w3}^2 & \overline{w'\theta_l'}\cdot\lambda_{w\theta} & \overline{w'r_t'}\cdot\lambda_{wr} \\ \overline{w'\theta_l'}\cdot\lambda_{w\theta} & \sigma_{\theta_l 3}^2 & \overline{r_t'\theta_l'}\cdot\lambda_{\theta r} \\ \overline{w'r_t'}\cdot\lambda_{wr} & \overline{r_t'\theta_l'}\cdot\lambda_{\theta r} & \sigma_{r_t 3}^2 \end{pmatrix}. \tag{4.1.6}$$

4.2 Normalized variables

Since CLUBB is mostly using "normalized variables", we are going to list those, which are given in standard form. We are also doing that for making the transformations easier.

$$\tilde{\theta}'_l \equiv \frac{\theta_l - \overline{\theta_l}}{\sqrt{\overline{\theta_l'^2}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_\theta}{1-\delta}}}, \tag{4.2.1}$$

$$\tilde{r}'_t \equiv \frac{r_t - \overline{r_t}}{\sqrt{\overline{r_t'^2}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_r}{1-\delta}}}, \tag{4.2.2}$$

where $\overline{\theta_l}$ and $\overline{r_t}$ are the means for the full summed up pdf and $\overline{\theta_l'^2}$ as well as $\overline{r_t'^2}$ are the variances for θ_l and r_t.

For the standard deviations, we define

$$\tilde{\sigma}_w \equiv \frac{\sigma_w}{\sqrt{\overline{w'^2}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_w}{1-\delta}}}, \tag{4.2.3}$$

where σ_w denotes the standard deviation of the w-component, $\overline{w'^2}$ is the variance for w,

$$\tilde{\sigma}_{\theta_l i} \equiv \frac{\sigma_{\theta_l i}}{\sqrt{\overline{\theta_l'^2}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_\theta}{1-\delta}}}, \tag{4.2.4}$$

where $\sigma_{\theta_l i}$ denotes the standard deviation of the i^{th} θ_l-component, and

$$\tilde{\sigma}_{r_t i} \equiv \frac{\sigma_{r_t i}}{\sqrt{\overline{r_t'^2}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_r}{1-\delta}}}, \tag{4.2.5}$$

where $\sigma_{r_t i}$ denotes the standard deviation of the i^{th} σ_{r_t}-component.

4.3 A list of lower order moments expressed in terms of pdf parameters

We start by outlining the equations capturing lower-order moments, expressed in terms of pdf parameters. These equations can be presented in either *dimensional* or *non-dimensional* form. While both representations are mathematically valid, the *non-dimensional* form offers a distinct advantage: it highlights the underlying connection to the bivariate case.

4.3.1 Moments for w

The relationship for $\overline{w}$ is given as follows:

$$\overline{w} = (1-\delta)\alpha w_1 + (1-\delta)(1-\alpha)w_2 + \delta w_3, \tag{4.3.1}$$

where $w_3 \equiv \alpha w_1 + (1-\alpha)w_2$. Therefore the mean of w stays the same as in the bivariate case. The relationship for the *non-dimensional* form is:

$$0 = \alpha\widehat{w}_1 + (1-\alpha)\widehat{w}_2 \tag{4.3.2}$$

For all other moments – except for the mean – we are using the standardized versions of the variables, written as w'.

The second order moment is given as:

$$\begin{aligned}\overline{w'^2} &= (1-\delta)\alpha[(w_1-\overline{w})^2 + \sigma_w^2] \\ &\quad + (1-\delta)(1-\alpha)[(w_2-\overline{w})^2 + \sigma_w^2] \\ &\quad + \delta\sigma_{w3}^2,\end{aligned} \tag{4.3.3}$$

where σ_{w3} is defined as $\lambda_w\overline{w'^2}$. This moment is also the variance of w at the same time, since

$$\begin{aligned}\overline{w'^2} &= \mathbb{E}[w'^2] = \mathbb{E}[(w-\overline{w})^2] = \mathbb{E}[(w-\mathbb{E}[w])^2] = \mathbb{E}[w^2 - 2w\mathbb{E}[w] + \mathbb{E}[w]^2] \\ &= \mathbb{E}[w^2] - 2\mathbb{E}[w\mathbb{E}[w]] + \mathbb{E}[\mathbb{E}[w]^2] = \mathbb{E}[w^2] - 2\mathbb{E}[w]\mathbb{E}[w] + \mathbb{E}[w]^2 \\ &= \mathbb{E}[w^2] - \mathbb{E}[w]^2 = \mathrm{Var}[w].\end{aligned}$$

The *non-dimensional* relationship would then be:

$$\frac{1}{(1-\tilde{\sigma}_w^2)} = \alpha\left(\widehat{w}_1^2 + \frac{\tilde{\sigma}_w^2}{(1-\tilde{\sigma}_w)^2}\right) + (1-\alpha)\left(\widehat{w}_2^2 + \frac{\tilde{\sigma}_w^2}{(1-\tilde{\sigma}_w^2)}\right). \tag{4.3.4}$$

The third order moment is given as:

$$\overline{w'^3} = (1-\delta)\alpha[(w_1 - \overline{w})^3 + 3\sigma_w^2(w_1 - \overline{w})] \\ + (1-\delta)(1-\alpha)[(w_2 - \overline{w})^3 + 3\sigma_w^2(w_2 - \overline{w})] \tag{4.3.5}$$

Since we want to make use of the specific shape of the pdf, we also have a relationship for $\overline{w'^3}$, which is called $\widehat{Sk}_w$, meaning the skewness of the variable w:

$$\widehat{Sk}_w \equiv \frac{1}{(1-\tilde{\sigma}_w^2)^{3/2}} \frac{\overline{w'^3}}{\left(\overline{w'^2}\right)^{3/2}} \frac{1}{\left(\frac{1-\delta\lambda_w}{1-\delta}\right)^{3/2}} \frac{1}{1-\delta} \\ = \alpha\left(\widehat{w}_1^3 + 3\widehat{w}_1 \frac{\tilde{\sigma}_w^2}{(1-\tilde{\sigma}_w^2)}\right) + (1-\alpha)\left(\widehat{w}_2^3 + 3\widehat{w}_2 \frac{\tilde{\sigma}_w^2}{(1-\tilde{\sigma}_w^2)}\right) \tag{4.3.6}$$

4.3.2 Moments for θ_l

For θ_l we have a similar *non-dimensional* relationship:

$$0 = \alpha\tilde{\theta}_{l1} + (1-\alpha)\tilde{\theta}_{l2} \tag{4.3.7}$$

Similarly but with a different standard deviation, $\overline{\theta_l'^2}$ is given as:

$$\overline{\theta_l'^2} = (1-\delta)\alpha[(\theta_{l1} - \overline{\theta_l})^2 + \sigma_{\theta_{l1}}^2] \\ + (1-\delta)(1-\alpha)[(\theta_{l2} - \overline{\theta_l})^2 + \sigma_{\theta_{l2}}^2] \\ + \delta\sigma_{\theta_l 3}^2, \tag{4.3.8}$$

where $\sigma_{\theta_l 3}$ is defined as $\lambda_{\theta_l}\overline{\theta_l'^2}$. This can also be expressed as the variance following the same approach as the one for $\overline{w'^2}$.

The third order moment is given as:

$$\overline{\theta_l'^3} = (1-\delta)\alpha[(\theta_{l1} - \overline{\theta_l})^3 + 3\sigma_{\theta_{l1}}^2(\theta_{l1} - \overline{\theta_l})] \\ + (1-\delta)(1-\alpha)[(\theta_{l2} - \overline{\theta_l})^3 + 3\sigma_{\theta_{l2}}^2(\theta_{l2} - \overline{\theta_l})] \tag{4.3.9}$$

Similarly to equation (4.3.6), we also list a moment which is more diagnosed than prognosed:

$$\widehat{Sk_{\theta_l}} \equiv \frac{\overline{\theta_l'^3}}{\left(\overline{\theta_l'^2}\right)^{3/2}} \left(\frac{1}{\frac{1-\delta\lambda_\theta}{1-\delta}}\right)^{3/2} \frac{1}{1-\delta}$$
$$= \alpha\left(\tilde{\theta}_{l1}^3 + 3\tilde{\theta}_{l1}\tilde{\sigma}_{\theta_l 1}^2\right) + (1-\alpha)\left(\tilde{\theta}_{l2}^3 + 3\tilde{\theta}_{l2}\tilde{\sigma}_{\theta_l 2}^2\right). \quad (4.3.10)$$

4.3.3 Moments for r_t

The relationships for r_t and $\overline{r_t'^2}$ are given as follows

$$0 = \alpha\tilde{r}_{t1} + (1-\alpha)\tilde{r}_{t2}, \quad (4.3.11)$$

and

$$1 = \alpha\left(\tilde{r}_{t1}^2 + \tilde{\sigma}_{r_{t1}}^2\right) + (1-\alpha)\left(\tilde{r}_{t2}^2 + \tilde{\sigma}_{r_{t2}}^2\right). \quad (4.3.12)$$

Since this relationship is similar to the relationships of θ_l and $\theta_l'^2$, we are using nearly the same formulas:

$$\overline{r_t'^2} = (1-\delta)\alpha[(r_{t1} - \overline{r_t})^2 + \sigma_{r_{t1}}^2]$$
$$+ (1-\delta)(1-\alpha)[(r_{t2} - \overline{r_t})^2 + \sigma_{\theta_{l2}}^2]$$
$$+ \delta\sigma_{r_t 3}^2, \quad (4.3.13)$$

and

$$\overline{r_t'^3} = (1-\delta)\alpha[(r_{t1} - \overline{r_t})^3 + 3\sigma_{r_{t1}}^2(r_{t1} - \overline{r_t})]$$
$$+ (1-\delta)(1-\alpha)[(r_{t2} - \overline{r_t})^3 + 3\sigma_{r_{t2}}^2(r_{t2} - \overline{r_t})]. \quad (4.3.14)$$

4.3.4 Mixed moments

There are also equations for two or even three variables, which are listed in the following.

$$\begin{aligned}\overline{w'\theta_l'} &= (1-\delta)\alpha[(w_1-\overline{w})(\theta_{l1}-\overline{\theta_l})] \\ &+ (1-\delta)(1-\alpha)[(w_2-\overline{w})(\theta_{l2}-\overline{\theta_l})] \\ &+ \delta\lambda_{w\theta}\overline{w'\theta_l'},\end{aligned} \tag{4.3.15}$$

$$\begin{aligned}\overline{w'r_t'} &= (1-\delta)\alpha[(w_1-\overline{w})(r_{t1}-\overline{r_t})] \\ &+ (1-\delta)(1-\alpha)[(w_2-\overline{w})(r_{t2}-\overline{r_t})] \\ &+ \delta\lambda_{wr}\overline{w'r_t'},\end{aligned} \tag{4.3.16}$$

and

$$\begin{aligned}\overline{r_t'\theta_l'} &= (1-\delta)\alpha\left[(r_{t1}-\overline{r_t})\left(\theta_{l1}-\overline{\theta_l}\right)+r_{r_t\theta_l}\sigma_{r_{t1}}\sigma_{\theta_{l1}}\right] \\ &+ (1-\delta)(1-\alpha)\left[(r_{t2}-\overline{r_t})\left(\theta_{l2}-\overline{\theta_l}\right)+r_{r_t\theta_l}\sigma_{r_{t2}}\sigma_{\theta_{l2}}\right] \\ &+ \delta\lambda_{r\theta}\overline{r_t'\theta_l'}.\end{aligned} \tag{4.3.17}$$

We have the *non-dimensional* relationship for those moments given as

$$\begin{aligned}\widehat{c}_{w\theta_l} &\equiv \frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}}\frac{\overline{w'\theta_l'}}{\sqrt{\overline{w'^2}}\sqrt{\overline{\theta_l'^2}}}\frac{1}{\sqrt{\frac{1-\delta\lambda_w}{1-\delta}}}\frac{1}{\sqrt{\frac{1-\delta\lambda_\theta}{1-\delta}}}\frac{1-\delta\lambda_{w\theta}}{1-\delta} \\ &= \alpha\widehat{w}_1\tilde{\theta}_{l1}+(1-\alpha)\widehat{w}_2\tilde{\theta}_{l2},\end{aligned} \tag{4.3.18}$$

$$\begin{aligned}\widehat{c}_{wr_t} &\equiv \frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}}\frac{\overline{w'r_t'}}{\sqrt{\overline{w'^2}}\sqrt{\overline{r_t'^2}}}\frac{1}{\sqrt{\frac{1-\delta\lambda_w}{1-\delta}}}\frac{1}{\sqrt{\frac{1-\delta\lambda_r}{1-\delta}}}\frac{1-\delta\lambda_{wr}}{1-\delta} \\ &= \alpha\widehat{w}_1\tilde{r}_{t1}+(1-\alpha)\widehat{w}_2\tilde{r}_{t2},\end{aligned} \tag{4.3.19}$$

and

$$\begin{aligned}\widehat{c}_{r_t\theta_l} &\equiv \frac{\overline{r_t'\theta_l'}}{\sqrt{\overline{r_t'^2}}\sqrt{\overline{\theta_l'^2}}}\frac{1}{\sqrt{\frac{1-\delta\lambda_q}{1-\delta}}}\frac{1}{\sqrt{\frac{1-\delta\lambda_\theta}{1-\delta}}}\frac{1-\delta\lambda_{\theta r}}{1-\delta} \\ &= \alpha\left(\tilde{r}_{t1}\tilde{\theta}_{l1}+r_{r_t\theta_l}\tilde{\sigma}_{q_{t1}}\tilde{\sigma}_{\theta_{l1}}\right)+(1-\alpha)\left(\tilde{r}_{t2}\tilde{\theta}_{l2}+r_{r_t\theta_l}\tilde{\sigma}_{r_{t2}}\tilde{\sigma}_{\theta_{l2}}\right),\end{aligned} \tag{4.3.20}$$

where we can think about $\widehat{c}$ as the correlation.

We also list a trivariate moment ($\overline{w'r_t'\theta_l'}$), given by:

$$\overline{w'r_t'\theta_l'} = (1-\delta)\alpha(w_1-\overline{w})\left[(r_{t1}-\overline{r_t})\left(\theta_{l1}-\overline{\theta_l}\right)+r_{r_t\theta_l}\sigma_{r_{t1}}\sigma_{\theta_{l1}}\right] \\ +(1-\delta)(1-\alpha)(w_2-\overline{w})\left[(r_{t2}-\overline{r_t})\left(\theta_{l2}-\overline{\theta_l}\right)+r_{r_t\theta_l}\sigma_{r_{t2}}\sigma_{\theta_{l2}}\right]. \quad (4.3.21)$$

4.4 Solving for pdf parameters by using the moment terms

Having established the prognosed moments for the desired pdf, we now try to retrieve the specific pdf that generates these moments. This process essentially involves inverting the relationship between the moments and the parameters that define the pdf.

In our case, we refer back to the normal mixture family of pdfs (equation (4.1.1)), which offers a representation for atmospheric grid layers. To select a particular member within this family that best aligns with the prognosed moments, we perform a parameter retrieval step.

This retrieval is achieved by inverting equations (4.3.2) to (4.3.20). These equations express the prognosed moments (mean, variance, covariances, etc.) as functions of the underlying pdf parameters (weights, means, and standard deviations). By inverting these relationships, we aim to find a set of pdf parameters that produces the distribution corresponding to the prognosed moments.

However, it is important to mention that this inversion is not a straightforward process. That is because the equations are non-linear with respect to the pdf parameters. Despite this non-linearity, the relatively simple structure of the normal mixture pdf (equation (4.1.1)) allows for an analytical solution to the inversion problem. This analytical solution enables us to efficiently map the prognosed moments back to the corresponding pdf parameters.

The proposed solution procedure [LG05] is as follows.

1. Solve for α, $\widehat{w}_1$, and $\widehat{w}_2$ from the equations for $\overline{w}$ (equation (4.3.2)), $\overline{w'^2}$ (equation (4.3.4)), $\overline{w'^3}$ (equation (4.3.6)):

$$\alpha = \frac{1}{2}\left[1 - \widehat{Sk}_w \sqrt{\frac{1}{4+\widehat{Sk}_w^2}}\right], \tag{4.4.1}$$

$$\widehat{w}_1 = \sqrt{\frac{1-\alpha}{\alpha}}, \tag{4.4.2}$$

$$\widehat{w}_2 = -\sqrt{\frac{\alpha}{1-\alpha}}. \tag{4.4.3}$$

   Without loss of generality, it has been chosen to set $\widehat{w}_1 > \widehat{w}_2$.

2. Looking at equation equation (4.4.1), we see that $\widehat{Sk}_w$ is determined only by α:

$$\widehat{Sk}_w = \frac{1-2\alpha}{\sqrt{\alpha(1-\alpha)}}. \tag{4.4.4}$$

3. $\tilde{\theta}_{l1}$ and $\tilde{\theta}_{l2}$ are taken from solving equation (4.3.7) for $\overline{\theta_l}$, and equation (4.3.18) for $\overline{w'\theta_l'}$:

$$\tilde{\theta}_{l1} = -\frac{\widehat{c}_{w\theta_l}}{\widehat{w}_2}, \tag{4.4.5}$$

$$\tilde{\theta}_{l2} = -\frac{\widehat{c}_{w\theta_l}}{\widehat{w}_1}. \tag{4.4.6}$$

4. We can get $\tilde{\sigma}_{\theta_l 1}$ and $\tilde{\sigma}_{\theta_l 2}$ by fulfilling equation (4.3.8) for $\overline{\theta_l'^2}$, and equation (4.3.9) for $\overline{\theta_l'^3}$:

$$\tilde{\sigma}_{\theta_l 1}^2 = \left(1 - \widehat{c}_{w\theta_l}^2\right) + \left(\sqrt{\frac{1-\alpha}{\alpha}}\right)\frac{1}{3\widehat{c}_{w\theta_l}}\left(\widehat{Sk}_{\theta_l} - \widehat{c}_{w\theta_l}^3 \widehat{Sk}_w\right), \tag{4.4.7}$$

$$\tilde{\sigma}^2_{\theta_l 2} = \left(1 - \widehat{c}^2_{w\theta_l}\right) - \left(\sqrt{\frac{\alpha}{1-\alpha}}\right) \frac{1}{3\widehat{c}_{w\theta_l}} \left(\widehat{Sk_{\theta_l}} - \widehat{c}^3_{w\theta_l}\widehat{Sk}_w\right). \tag{4.4.8}$$

Sk_{θ_l} represents the skewness of θ_l, which has to be provided by an equation such as equation (4.3.10) below.

5. Finding formulas for $\tilde{r}_{t1}$, $\tilde{r}_{t2}$, $\tilde{\sigma}^2_{r_t 1}$, and $\tilde{\sigma}^2_{r_t 2}$ can be done by replacing θ_l by r_t everywhere in the equations (4.4.5), (4.4.6), (4.4.7), and (4.4.8).

6. The last step is to get a relationship between $r_{r_t\theta_l}$, the in-between normal correlation and $c_{r_t\theta_l}$, the total correlation. This can be done by using equation (4.3.17):

$$r_{r_t\theta_l} = \frac{\widehat{c}_{r_t\theta_l} - \widehat{c}_{wr_t}\widehat{c}_{w\theta_l}}{\alpha\tilde{\sigma}_{r_t 1}\tilde{\sigma}_{\theta_l 1} + (1-\alpha)\tilde{\sigma}_{r_t 2}\tilde{\sigma}_{\theta_l 2}}. \tag{4.4.9}$$

4.5 Expressions for higher-order moments in terms of pdf parameters

Upon determining the pdf parameters, we gain the ability to compute all higher-order moments associated with the distribution. These moments play a crucial role for closing the already described pdes. The symbolic calculation of higher-order moments can be achieved through integration over the specified pdf. Formulas for calculating various higher-order moments within the context of a binormal pdf are readily available in the literature [LG05].

We state the transformed formulas needed for closure in the following.

$$
\frac{1}{(1-\tilde{\sigma}_w^2)^2}\frac{(1-\delta)}{(1-\delta\lambda_w)^2}\frac{\overline{w'^4}}{\left(\overline{w'^2}\right)^2} = \alpha\left[\widehat{w}_1^4 + 6\widehat{w}_1^2\frac{\tilde{\sigma}_w^2}{(1-\tilde{\sigma}_w^2)} + 3\frac{\tilde{\sigma}_w^4}{(1-\tilde{\sigma}_w^2)^2}\right]
$$

$$
+ (1-\alpha)\left[\widehat{w}_2^4 + 6\widehat{w}_2^2\frac{\tilde{\sigma}_w^2}{(1-\tilde{\sigma}_w^2)} + 3\frac{\tilde{\sigma}_w^4}{(1-\tilde{\sigma}_w^2)^2}\right]
$$

$$
+ \frac{1}{(1-\tilde{\sigma}_w^2)^2}\frac{(1-\delta)}{(1-\delta\lambda_w)^2}\delta 3\lambda_w^2, \tag{4.5.1}
$$

$$
\frac{1}{(1-\tilde{\sigma}_w^2)}\frac{(1-\delta)^{1/2}}{(1-\delta\lambda_w)(1-\delta\lambda_\theta)^{1/2}}\frac{\overline{w'^2\theta_l'}}{\overline{w'^2}\left(\overline{\theta_l'^2}\right)^{1/2}} = \alpha\left[\widehat{w}_1^2 + \frac{\tilde{\sigma}_w^2}{(1-\tilde{\sigma}_w^2)}\right]\tilde{\theta}_{l1}
$$

$$
+ (1-\alpha)\left[\widehat{w}_2^2 + \frac{\tilde{\sigma}_w^2}{(1-\tilde{\sigma}_w^2)}\right]\tilde{\theta}_{l2}, \tag{4.5.2}
$$

$$
\frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}}\frac{(1-\delta)^{1/2}}{(1-\delta\lambda_w)^{1/2}(1-\delta\lambda_\theta)}\frac{\overline{w'\theta_l'^2}}{\left(\overline{w'^2}\right)^{1/2}\overline{\theta_l'^2}} = \alpha\widehat{w}_1\left(\tilde{\theta}_{l1}^2 + \tilde{\sigma}_{\theta_{l1}}^2\right) + (1-\alpha)\widehat{w}_2\left(\tilde{\theta}_{l2}^2 + \tilde{\sigma}_{\theta_{l2}}^2\right), \tag{4.5.3}
$$

$$
\frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}}\frac{(1-\delta)^{1/2}}{(1-\delta\lambda_w)^{1/2}(1-\delta\lambda_\theta)^{1/2}(1-\delta\lambda_{q_t})^{1/2}}\frac{\overline{w'r_t'\theta_l'}}{\left(\overline{w'^2}\right)^{1/2}\left(\overline{r_t'^2}\right)^{1/2}\left(\overline{\theta_l'^2}\right)^{1/2}}
$$

$$
= \alpha\widehat{w}_1\left(\tilde{r}_{t1}\tilde{\theta}_{l1} + r_{r_t\theta_l}\tilde{\sigma}_{r_{t1}}\tilde{\sigma}_{\theta_{l1}}\right) + (1-\alpha)\widehat{w}_2\left(\tilde{r}_{t2}\tilde{\theta}_{l2} + r_{r_t\theta_l}\tilde{\sigma}_{r_{t2}}\tilde{\sigma}_{\theta_{l2}}\right) \tag{4.5.4}
$$

Equations for $\overline{w'^2r_t'}$ and $\overline{w'r_t'^2}$ are similar to equation (4.5.2) and equation (4.5.3) by replacing θ_l with r_t everywhere.

4.6 Approximation of scalar skewnesses

Closing the system of prognostic equations within CLUBB needs the specification of the skewness terms Sk_{θ_l} and Sk_{r_t}. These skewness values appear in the solutions for $\tilde{\sigma}_{\theta_l 1}$ (equation (4.4.7)) and $\tilde{\sigma}_{\theta_l 2}$ (equation (4.4.8)), respectively. Traditionally, these skewness terms could be treated as prognostic variables, requiring their own prognostic equations and adding to the overall intense computation.

Therefore, the paper "Using probability density functions to derive consistent closure relationships among higher-order moments" by Larson and Golaz which this work is based on, proposes an alternative approach that uses a diagnostic formula for skewness. The formula provides a reasonable estimate of the skewness terms based on the readily available prognostic moments, avoiding the need for dedicated prognostic equations for skewness. This strategy results in closure of the system of equations while maintaining a computationally tractable model. The proposed formula is the following:

$$\widehat{Sk}_{\theta_l} = \widehat{Sk}_w \widehat{c}_{w\theta_l} \left[\beta + (1-\beta)\widehat{c}^2_{w\theta_l}\right], \tag{4.6.1}$$

which is similar for $\widehat{Sk}_{r_t}$ by again just replacing r_t with θ_l. They define a parameter β which is dimensionless. We also solve for β because we are going to need the equation later on to show that other equations are true.

$$\implies \beta = \frac{\frac{\widehat{Sk}_{\theta_l}}{\widehat{Sk}_w \widehat{c}_{w\theta_l}} - \widehat{c}^2_{w\theta_l}}{1 - \widehat{c}^2_{w\theta_l}} \tag{4.6.2}$$

Equation (4.6.1) presents a diagnostic formula for estimating the skewness of θ_l. This formula offers a physically intuitive relationship. It proposes a proportionality between Sk_{θ_l} and Sk_w. However, it's crucial to acknowledge the limitations in this diagnostic. The formula suggests that an increase in the parameter β leads to a larger magnitude of Sk_{θ_l}. This translates to a pdf with a more extended tail in the θ_l domain. Furthermore, the formula captures the behavior when w and θ_l are correlated. That is, positive skewness in w leads to positive skewness in θ_l (positive correlation), and vice versa (negative correlation). However, it is important to mention that real-world large eddy simulations may show deviations from this simplified relationship.

Another limitation appears when either Sk_w or the covariance between w and θ_l ($c_{w\theta_l}$)

approaches zero. The formula predicts a vanishing Sk_{θ_l} in these scenarios, which may not always be true.

Finally, the diagnostic approach allows for the magnitude of $|Sk_{\theta_l}|$ to be either smaller or larger than $|Sk_w|$. This behavior depends on the interplay between the variance of w ($\tilde{\sigma}_w^2$), the covariance ($c_{w\theta_l}$), and the parameter β. This highlights the potential for an inconsistency between the estimated skewness and the other binormal moments, e.g. a single value of β may not correspond exactly to any trivariate normal distribution

To summarize, equation (4.6.1) offers a computationally efficient method for skewness estimation, but it comes with limitations. While it captures some key aspects of the relationship between the skewness in w and the skewness in θ_l, one should be aware of deviations from its predictions.

We proceed with using equation (4.3.10) for $\widehat{Sk}_{\theta_l}$ and find the following relationships [LG05] for $\tilde{\sigma}_{\theta_l 1}^2$ and $\tilde{\sigma}_{\theta_l 2}^2$:

$$\tilde{\sigma}_{\theta_l 1}^2 = \frac{\left(1 - \widehat{c}_{w\theta_l}^2\right)}{\alpha} \left[\frac{1}{3}\beta + \alpha \left(1 - \frac{2}{3}\beta\right)\right], \tag{4.6.3}$$

and

$$\tilde{\sigma}_{\theta_l 2}^2 = \frac{\left(1 - \widehat{c}_{w\theta_l}^2\right)}{1 - \alpha} \left\{1 - \left[\frac{1}{3}\beta + \alpha \left(1 - \frac{2}{3}\beta\right)\right]\right\}. \tag{4.6.4}$$

By using the previously stated expressions for the standard deviations (equations (4.6.3) and (4.6.4), with their r_t counterparts), we can substitute them into the formula for the correlation between r_t and θ_l (equation (4.4.9)). This substitution leads to a more concise representation.

$$r_{r_t\theta_l} = \frac{c_{r_t\theta_l} - \widehat{c}_{wr_t}\widehat{c}_{w\theta_l}}{\left(1 - \widehat{c}_{wr_t}^2\right)^{1/2} \left(1 - \widehat{c}_{w\theta_l}^2\right)^{1/2}}, \tag{4.6.5}$$

where the correlation of r_t and θ_l within the individual normal distributions is $r_{r_t\theta_l}$, and $c_{r_t\theta_l}$ represents the total correlation across the entire trinormal pdf.

4.7 Formulating closure relationships for higher-order moments in terms of lower-order moments

This section delves into the derivation of closure relationships for crucial higher-order moments employed within the CLUBB parameterization.

Our focus here lies on achieving closure for the following terms:

- $\overline{w'^4}$: The fourth-order moment of up-/downdrafts,
- $\overline{w'^2\theta_l'}$: the so-called flux of flux,
- $\overline{w'\theta_l'^2}$: the flux of variance,
- $\overline{w'r_t'\theta_l'}$: and the flux of covariance.

Closure, in this context, refers to expressing these higher-order moments completely in terms of known quantities, typically lower-order moments that are directly prognosed by the model. The approach to derive those formulas is based on the previously established expressions for the pdf parameters (equations (4.4.1) to (4.4.9)). By substituting those derived pdf parameter expressions into the relevant equations for the higher-order moments (equations (4.5.1) to (4.5.4)), we find the desired closure relationships.

We first present the equation for the third moment of θ_l', $\overline{\theta_l'^3}$ This expression is derived by dimensionalizing equation (4.6.1), which relates the skewness of θ_l to the skewness of w and other model parameters.

$$\overline{\theta_l'^3} = \frac{(1-\delta\lambda_{w\theta})(1-\delta\lambda_\theta)}{\left(1-\tilde{\sigma}_w^2\right)^2(1-\delta\lambda_w)^2}\frac{\overline{w'^3}}{\left(\overline{w'^2}\right)^2}\overline{\theta_l'^2}\,\overline{w'\theta_l'}\left(\beta+(1-\beta)\frac{(1-\delta\lambda_{w\theta})^2}{1-\tilde{\sigma}_w^2(1-\delta\lambda_w)(1-\delta\lambda_\theta)}\frac{\left(\overline{w'\theta_l'}\right)^2}{\overline{w'^2}\,\overline{\theta_l'^2}}\right). \tag{4.7.1}$$

While the scalar third moments (e.g., $\overline{\theta_l'^3}$) may not directly participate in solving the prognostic equations within CLUBB, they hold an indirect yet crucial role in shaping cloud properties within atmospheric simulations. This influence is coming from the connection between the pdf and the cloud formation.

Cumulus cloud formation mostly occurs at the edges, or "tails" of the pdf for a specific variable. These tails represent regions where the probability of encountering extreme values of the variable is relatively higher. As the relative "width" of the normal distribution representing w increases ($\tilde{\sigma}_w$), the magnitude of $\overline{\theta_l'^3}$ also grows (refer to equation (4.7.1) for details). In simpler terms, a larger value of $\overline{\theta_l'^3}$ corresponds to a broader pdf for the up-/downdraft variable. This broader pdf deviates more significantly from a double delta function, which is a construct with two spikes at zero.

Unlike the scalar third moment, $\overline{w'^4}$ does not depend on the thermodynamic scalar moments (such as $\overline{\theta_l'^3}$). Consequently, it is independent of the parameter β. To derive the explicit formula for $\overline{w'^4}$, we can substitute the previously stated expressions for $\widehat{w_1}$ (equation (4.4.2)) and $\widehat{w_2}$ (equation (4.4.3)) into equation (4.5.1).

$$\frac{1}{(1-\tilde{\sigma}_w^2)^2}\frac{(1-\delta)}{(1-\delta\lambda_w)^2}\frac{\overline{w'^4}}{\left(\overline{w'^2}\right)^2} = 3\frac{\tilde{\sigma}_w^4}{(1-\tilde{\sigma}_w^2)^2} + 6\frac{\tilde{\sigma}_w^2}{(1-\tilde{\sigma}_w^2)} + 1 + \widehat{Sk}_w^2 + \frac{1}{(1-\tilde{\sigma}_w^2)^2}\frac{(1-\delta)}{(1-\delta\lambda_w)^2}\delta 3\lambda_w^2, \tag{4.7.2}$$

leading to

$$\begin{aligned}\overline{w'^4} = {} & \left(\overline{w'^2}\right)^2 \frac{(1-\delta\lambda_w)^2}{(1-\delta)}\left(3\tilde{\sigma}_w^4 + 6\left(1-\tilde{\sigma}_w^2\right)\tilde{\sigma}_w^2 + \left(1-\tilde{\sigma}_w^2\right)^2\right) \\ & + \frac{1}{(1-\tilde{\sigma}_w^2)}\frac{1}{(1-\delta\lambda_w)}\frac{\left(\overline{w'^3}\right)^2}{\overline{w'^2}} \\ & + \delta 3\lambda_w^2\left(\overline{w'^2}\right)^2.\end{aligned} \tag{4.7.3}$$

As observed with $\overline{w'^4}$, $\overline{w'^2\theta_l'}$ displays independence from the parameter β within the context of the chosen pdf.

To proceed, we can substitute the previously derived expressions for $\widehat{w}_1$ (equation (4.4.2)), $\widehat{w}_2$ (equation (4.4.3)), $\widehat{Sk}_w$ (equation (4.3.6)), $\tilde{\theta}_{l1}$ (equation (4.4.5)), and $\tilde{\theta}_{l2}$ (equation (4.4.6)) into equation (4.5.2). This substitution process will yield an explicit formula for $\overline{w'^2\theta_l'}$ that solely relies on known quantities, such as the prognostic moments directly calculated by the model.

$$\frac{1}{(1-\tilde{\sigma}_w^2)}\frac{(1-\delta)^{1/2}}{(1-\delta\lambda_w)(1-\delta\lambda_\theta)^{1/2}}\frac{\overline{w'^2\theta_l'}}{\overline{w'^2}\left(\overline{\theta_l'^2}\right)^{1/2}} = \widehat{c}_{w\theta_l}\widehat{Sk}_w, \tag{4.7.4}$$

and

$$\overline{w'^2\theta_l'} = \frac{1}{(1-\tilde{\sigma}_w^2)}\frac{1-\delta\lambda_{w\theta}}{1-\delta\lambda_w}\frac{\overline{w'^3}}{\overline{w'^2}}\overline{w'\theta_l'}. \tag{4.7.5}$$

$\overline{w'\theta_l'^2}$ depends explicitly on Sk_{θ_l}. Substituting equation (4.4.2) - equation (4.4.8) into equation (4.5.4) yields

$$\frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}}\frac{(1-\delta)^{1/2}}{(1-\delta\lambda_w)^{1/2}(1-\delta\lambda_\theta)}\frac{\overline{w'\theta_l'^2}}{\left(\overline{w'^2}\right)^{1/2}\overline{\theta_l'^2}} = \frac{2}{3}\widehat{c}_{w\theta_l}^2\widehat{Sk}_w + \frac{1}{3}\frac{\widehat{Sk}_{\theta_l}}{\widehat{c}_{w\theta_l}}, \tag{4.7.6}$$

and

$$\begin{aligned}\overline{w'\theta_l'^2} = &\frac{2}{3}\frac{(1-\delta\lambda_{w\theta})^2}{(1-\delta\lambda_w)^2}\frac{1}{(1-\tilde{\sigma}_w^2)^2}\frac{\overline{w'^3}}{\left(\overline{w'^2}\right)^2}\left(\overline{w'\theta_l'}\right)^2 \\ &+\frac{1}{3}\frac{(1-\delta\lambda_w)}{(1-\delta\lambda_{w\theta})}\left(1-\tilde{\sigma}_w^2\right)\frac{\overline{w'^2}\,\overline{\theta_l'^3}}{\overline{w'\theta_l'}}.\end{aligned} \tag{4.7.7}$$

5 Integration using SymPy

Throughout this thesis, symbolic manipulation plays a crucial role in verifying mathematical expressions, particularly integrals. To achieve this, we rely on SymPy [Meu+17] – a powerful Python library for symbolic mathematics. This chapter dives into the world of SymPy, showcasing its capabilities through a detailed example.

We begin by demonstrating the analytical approach to solving an integral. Next, we will explore the numerical side of integration. We are going to demonstrate how SymPy can be seamlessly integrated with numerical computing libraries to evaluate the integral for specific input values. This combined approach allows us to not only verify our analytical solution but also gain valuable insights into the integral's behavior for different scenarios. By following this step-by-step example, the reader will gain a solid understanding of how SymPy can be used, not only for this thesis.

5.1 Analytic integration

For simplicity and readability, we choose the check for the formula of $\overline{w'^2}$ (this is item 3 from section 2.6). One starts by importing and – obviously – installing the packages if they are not there yet. Importing the package `display` is useful for later on printing the equations. Thus, this results in the code in listing 5.1. In this listing, `sympy` was defined to be called `sp` and from `sympy` we directly imported some packages, too, which are needed later on.

Next, we define all symbols which are needed to calculate the given integral and therefore also to print the equations nicely. Since we are checking $\overline{w'^2}$, we need the (self-defined)

Listing 5.1: Import statements

```
import sympy as sp
from IPython.display import display
from sympy import abc, oo, Symbol, Integral
from sympy.stats import Normal, density
```

symbols listed in listing 5.2. Having defined the symbols, we can proceed with defining

Listing 5.2: Defining symbols

```
sigma_w = Symbol('\\sigma_w')
w_1 = Symbol('w_1')
w_2 = Symbol('w_2')
w_bar = Symbol('\\overline{w}')
sigma_w_3 = Symbol('\\sigma_{w3}')
w_prime_2_bar = Symbol('\\overline{w\'^2}')
```

the marginal distribution. Now, we are also using `sympy.abc` for displaying some standard symbols (listing 5.3). Having done that we can actually display the integral which we want

Listing 5.3: Defining the marginals

```
G_1_w = Normal(name='G_1_w', mean=w_1, std=sigma_w)
G_1_w_density = density(G_1_w)(sp.abc.w)
G_2_w = Normal(name='G_2_w', mean=w_2, std=sigma_w)
G_2_w_density = density(G_2_w)(sp.abc.w)
G_3_w = Normal(name='G_3_w', mean=w_bar, std=sigma_w_3)
G_3_w_density = density(G_3_w)(sp.abc.w)
G_w = ((1 - sp.abc.delta) * sp.abc.alpha * G_1_w_density +
       (1 - sp.abc.delta) * (1 - sp.abc.alpha) * G_2_w_density +
       sp.abc.delta * G_3_w_density)
```

to compute (listing 5.4). Looking at figure 5.1, this is exactly the integral which we

Listing 5.4: Defining and displaying the needed integral

```
w_prime_2_bar_int = sp.Integral((sp.abc.w - w_bar) ** 2 * G_w, [sp.abc.w, -oo, oo])
display(sp.Eq(w_prime_2_bar, w_prime_2_bar_int))
```

want to compute. Using the command `.doit(conds='none')` in listing 5.5, we can actually

Figure 5.1: Output of listing 5.4

$$\overline{w'^2} = \int\limits_{-\infty}^{\infty} (-\overline{w} + w)^2 \left(\frac{\sqrt{2}\delta e^{-\frac{(-\overline{w}+w)^2}{2\sigma_{w3}^2}}}{2\sqrt{\pi}\sigma_{w3}} + \frac{\sqrt{2}\alpha\,(1-\delta)\,e^{-\frac{(w-w_1)^2}{2\sigma_w^2}}}{2\sqrt{\pi}\sigma_w} + \frac{\sqrt{2}\cdot(1-\alpha)\,(1-\delta)\,e^{-\frac{(w-w_2)^2}{2\sigma_w^2}}}{2\sqrt{\pi}\sigma_w} \right) dw$$

calculate the given integral, where we assume that all given constants are real. We are also using `.simplify()` here to make the output more readable as well as more comparable to the actual function we want to check. We can now compare figure 5.2 to the given equation.

Listing 5.5: Calculating and printing the integral

```
w_prime_2_bar_int_val = w_prime_2_bar_int.doit(conds='none').simplify()
display(sp.Eq(w_prime_2_bar, w_prime_2_bar_int_val))
```

Figure 5.2: Output of listing 5.5

$$\begin{aligned}\overline{w'^2} = -\overline{w}^2\delta + \overline{w}^2 + 2\overline{w}\alpha\delta w_1 - 2\overline{w}\alpha\delta w_2 - 2\overline{w}\alpha w_1 + 2\overline{w}\alpha w_2 + 2\overline{w}\delta w_2 - 2\overline{w}w_2 \\ -\sigma_w^2\delta + \sigma_w^2 + \sigma_{w3}^2\delta - \alpha\delta w_1^2 + \alpha\delta w_2^2 + \alpha w_1^2 - \alpha w_2^2 - \delta w_2^2 + w_2^2,\end{aligned}$$

To do this, we first need to define the equation for equation (4.3.3) in listing 5.6. We can

Listing 5.6: Python function for the second order moment

```
def w_prime_2_bar_check(delta=sp.abc.delta, alpha=sp.abc.alpha, w_1=w_1, w_2=w_2,
↪   w_bar=w_bar, sigma_w=sigma_w, sigma_w_3=sigma_w_3):
    return (((1 - delta) * alpha * ((w_1 - w_bar) ** 2 + sigma_w ** 2))
        + ((1 - delta) * (1 - alpha) * ((w_2 - w_bar) ** 2 + sigma_w ** 2))
        + (delta * sigma_w_3 ** 2))
```

print this equation using `display` again (listing 5.7). The last step is to check if those two

Listing 5.7: Printing the symbolic equation

```
display(sp.Eq(w_prime_2_bar, w_prime_2_bar_check()))
```

Figure 5.3: Output of listing 5.7

$$\overline{w'^2} = \sigma_{w3}^2\delta + \alpha\left(1-\delta\right)\left(\sigma_w^2 + \left(-\overline{w} + w_1\right)^2\right) + \left(1-\alpha\right)\left(1-\delta\right)\left(\sigma_w^2 + \left(-\overline{w} + w_2\right)^2\right)$$

formulas are equivalent to each other. We can do this by using `Eq(..)` from the package `SymPy`. `factor(..)` tries to factor the given variables to make the comparison easier. All of this can be seen in listing 5.8. This code (listing 5.8) just displays `True`, which is exactly

Listing 5.8: Check if the integral and the given formula are the same

```
display(sp.factor(sp.Eq(w_prime_2_bar_int_val, w_prime_2_bar_check()), sp.abc.alpha,
↪   sp.abc.delta))
```

what we wanted to have.

5.2 Numeric integration

Again, for better readability, we choose to check the formula for $\overline{w'^2\theta_l'}$ (this is item 6 from section 2.6). As in section 5.1, there needs to be some packages imported. We are importing the same packages as in listing 5.1 together with some more (listing 5.9). Since we are going

Listing 5.9: Import statements

```
from itertools import product
import pandas as pd
import numpy as np
```

to need some more symbols, we also need to define those. We still use the symbols as in listing 5.2, together with the ones in listing 5.10.

Listing 5.10: Defining symbols

```
theta_l_1 = Symbol('\\theta_{l1}')
theta_l_2 = Symbol('\\theta_{l2}')
theta_l_bar = Symbol('\overline{\\theta_l}')
sigma_theta_l_1 = Symbol('\\sigma_{\\theta_{l1}}')
sigma_theta_l_2 = Symbol('\\sigma_{\\theta_{l2}}')
sigma_theta_l_3 = Symbol('\\sigma_{\\theta_l 3}')
rho_w_theta_l = Symbol('\\rho_{w\\theta_l}')
w_prime_3_bar = Symbol('\\overline{w\'^3}')
w_prime_theta_l_prime_bar = Symbol('\\overline{w\'\\theta\'_l}')
w_prime_2_theta_prime_l_bar = Symbol('\\overline{w\'^2\\theta\'_l}')
sigma_tilde_w = Symbol('\\tilde{\\sigma}_w')
lambda_w_theta = Symbol('\\lambda_{w\\theta}')
lambda_w = Symbol('\\lambda_w')
```

We start defining the integral by defining the marginals (listing 5.11).

Listing 5.11: Defining the marginals

```
G_1_w_theta = Normal(name='G_1_w_theta', mean=sp.Matrix([w_1, theta_l_1]),
    std=sp.Matrix([[sigma_w ** 2, 0], [0, sigma_theta_l_1 ** 2]]))
G_1_w_theta_density = density(G_1_w_theta)(sp.abc.w, sp.abc.theta)
G_2_w_theta = Normal(name='G_2_w_theta', mean=sp.Matrix([w_2, theta_l_2]),
    std=sp.Matrix([[sigma_w ** 2, 0], [0, sigma_theta_l_2 ** 2]]))
G_2_w_theta_density = density(G_2_w_theta)(sp.abc.w, sp.abc.theta)
G_3_w_theta = Normal(name='G_3_w_theta', mean=sp.Matrix([w_bar, theta_l_bar]),
    std=sp.Matrix([[sigma_w_3 ** 2,
        rho_w_theta_l * sigma_w_3 * sigma_theta_l_3],
        [rho_w_theta_l * sigma_w_3 * sigma_theta_l_3,
        sigma_theta_l_3 ** 2]]))
G_3_w_theta_density = sp.simplify(density(G_3_w_theta)(sp.abc.w, sp.abc.theta))
G_w_theta = (
(1 - sp.abc.delta) * sp.abc.alpha * density(G_1_w_theta)(sp.abc.w, sp.abc.theta)
+ (1 - sp.abc.delta) * (1 - sp.abc.alpha) * density(G_2_w_theta)(sp.abc.w, sp.abc.theta)
+ sp.abc.delta * G_3_w_theta_density)
```

The integral which needs to be computed is then defined as in listing 5.12.

Here (listing 5.12), the output is omitted for better readability. We do not yet compute the integral, because due to the complexity, unfortunately this is not working with `SymPy`.

Listing 5.12: Defining and displaying the needed integral

```
w_prime_2_theta_l_prime_bar = sp.Integral((sp.abc.w - w_bar) ** 2 *
    (sp.abc.theta - theta_l_bar) * G_w_theta,
    [sp.abc.w, -oo, oo], [sp.abc.theta, -oo, oo])
display(sp.Eq(w_prime_2_theta_prime_l_bar, w_prime_2_theta_l_prime_bar))
```

Since there is still the equation to check needed, we proceed by defining a function for that in listing 5.13. Looking at figure 5.4, there are some other equations needed like

Listing 5.13: Python function for $\overline{w'^2\theta_l}$

```
def w_prime_2_theta_l_prime_bar_check(sigma_tilde_w = sigma_tilde_w,
    delta = sp.abc.delta, lambda_w_theta = lambda_w_theta, lambda_w = lambda_w,
    w_prime_3_bar = w_prime_3_bar, w_prime_2_bar = w_prime_2_bar,
    w_prime_theta_l_prime_bar = w_prime_theta_l_prime_bar):
    return ((1 / (1 - sigma_tilde_w ** 2)) *
        ((1 - delta * lambda_w_theta) / (1 - delta * lambda_w)) *
        (w_prime_3_bar / w_prime_2_bar) *
        w_prime_theta_l_prime_bar)
display(sp.Eq(w_prime_2_theta_prime_l_bar, w_prime_2_theta_l_prime_bar_check()))
```

Figure 5.4: Output of listing 5.13

$$\overline{w'^2\theta_l'} = \frac{\overline{w'\theta_l'} \cdot \overline{w'^3}\left(-\lambda_{w\theta}\delta + 1\right)}{\overline{w'^2} \cdot \left(1 - \tilde{\sigma}_w^2\right)\left(-\lambda_w\delta + 1\right)}$$

equation (4.3.15), equation (4.3.5), equation (4.3.3), equation (4.2.3), and equation (4.1.4). We do not list the functions to those equations here, because they are defined the same way as the other equations are defined as functions.

Instead, since we cannot compute the integral analytically, we can create a `dataframe` using `pandas`[McK10]. The columns for this dataframe are going to be all the inputs we have. To get all permutations, this code (listing 5.14) is also using `product(..)` from the `itertools` package.

Listing 5.14: Create a dataframe and putting in arbitrary numbers

```
df = pd.DataFrame(product([0, 1],[-2, 2],[-1, 2],[0, 3], [Rational(1, 10)],
    [Rational(3, 10)],[Rational(4, 10)],[Rational(7, 10)],[Rational(6, 10)],
    [Rational(5, 10)],[Rational(1, 10), Rational(5, 10)],[Rational(5, 10)]),
    columns=[w_1, w_2, theta_l_1, theta_l_2, sigma_theta_l_1, sigma_theta_l_2,
    sigma_lambda_theta_l, sigma_w, sigma_lambda_w, sp.abc.alpha, sp.abc.delta,
    rho_w_theta_l])
```

We append another column which is called "checkval" and lists the values for the given equation to check. This code also uses the function defined in listing 5.13, where all other

Listing 5.15: Attaching the "checkval" column to the dataframe

```
df['checkval'] = (df.apply(lambda x: w_prime_2_theta_l_prime_bar_check_val.subs({
        w_1: x[w_1], w_2: x[w_2], theta_l_1: x[theta_l_1], theta_l_2: x[theta_l_2],
        sigma_theta_l_1: x[sigma_theta_l_1], sigma_theta_l_2: x[sigma_theta_l_2],
        sigma_lambda_theta_l: x[sigma_lambda_theta_l], sigma_w: x[sigma_w],
        sigma_lambda_w: x[sigma_lambda_w], sp.abc.alpha: x[sp.abc.alpha],
        sp.abc.delta: x[sp.abc.delta], rho_w_theta_l: x[rho_w_theta_l]}), axis=1))
```

equations are substituted into. The function `df.apply(..)` is used to apply the function given in the parenthesis to all rows of the dataframe by specifying a `lambda x`, where `x` is corresponding to the given dataframe, `df`. Lastly, there is also the `axis=1` parameter, which specifies the direction of applying the function.

Next, we are actually computing $\overline{w'^2\theta_l}$ numerically by using the quadrature method and applying the values of this integrals to a new column in the dataframe (listing 5.16). Here,

Listing 5.16: Attaching the "numint" column to the dataframe

```
df['numint'] = (df.apply(lambda x: Rational(w_prime_2_theta_l_prime_bar.subs({
    w_1: x[w_1],w_2: x[w_2], theta_l_1: x[theta_l_1], theta_l_2: x[theta_l_2],
    sigma_theta_l_1: x[sigma_theta_l_1], sigma_theta_l_2: x[sigma_theta_l_2],
    sigma_lambda_theta_l: x[sigma_lambda_theta_l], sigma_w: x[sigma_w],
    sigma_lambda_w: x[sigma_lambda_w], sp.abc.alpha: x[sp.abc.alpha],
    sp.abc.delta: x[sp.abc.delta], rho_w_theta_l: x[rho_w_theta_l]
}).doit(conds='none', method='quad').evalf()), axis=1))
```

we are using the integral which has been specified earlier and specify the numerical integration method by adding the parameter `method='quad'` [1] to the function `.doit(..)`. After that, `.evalf(..)` just gives the numerical value. We try to prove that the integral value equals the function value, hence we are computing the error between those two columns (listing 5.17) and take the mean (`numpy.mean(..)` from the package NumPy [Har+20]) of these new columns (listing 5.18) to see if the error is actually numerically 0.

Listing 5.17: Attaching the "diffnum" column to the dataframe

```
df['diffnum'] = abs(df['checkval'].astype(float) - df['numint'].astype(float))
```

Listing 5.18: Calculating the mean difference

```
print('The mean error between the rhs and the lhs is:', np.mean(df['diffnum']))
```

Figure 5.5: Output of listing 5.18

The mean error between the rhs and the lhs is: 1.3753423344481015e-124

In figure 5.5, we see that the mean error is basically 0 which we wanted. It should be noted that based on the configuration of each individual computer, the solutions can slightly differ due to floating point arithmetic.

[1] This parameter is telling SymPy to use the quadrature method to compute the given integral numerically. For further explanation on how this method is working, we refer to the SymPy documentation which can be found here: `https://docs.sympy.org/latest/modules/integrals/integrals.html`, as well as the SciPy documentation on integration, which can be found here: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.quad.html`. Some SciPy [Vir+20] functions are called by the SymPy library. Therefore, this reference shows up here.

6 Asymptotics

Once we defined all functions, we see that we want certain behaviors for certain values as well as there is a need to restrict some parameter values.

We start with the "obvious" restrictions for the pdf parameters. The mixture fractions α and δ are meant to be $\alpha \in [0,1]$ and $\delta \in [0,1)$. But since the code tries to simplify a lot of things, the binormal representation also does not revert back to a single normal distribution. Therefore, we have $\alpha \in (0,1)$ due to the code. The restriction to δ makes sense in a way that we do not really want just the third normal to predict the whole shape. Also, most of the formulas, e.g. equation (4.3.6) have a $1-\delta$ in the denominator.

In section 2.3, we saw, how the transformations between the sum of two normal distributions and the sum of three normal distributions are working. From those transformations, we see that we want

$$\begin{aligned}
& 0<\delta\lambda_w<1, && 0<\delta\lambda_\theta<1, && 0<\delta\lambda_r<1,\\
\iff\quad & 0<\delta\tfrac{\sigma_{w3}^2}{\overline{w'^2}}<1, && 0<\delta\tfrac{\sigma_{\theta_l 3}^2}{\overline{\theta_l'^2}}<1, && 0<\delta\tfrac{\sigma_{r_t 3}^2}{\overline{r_t'^2}}<1,\\
\iff\quad & 0<\delta\sigma_{w3}^2<\overline{w'^2}, && 0<\delta\sigma_{\theta_l 3}^2<\overline{\theta_l'^2}, && 0<\delta\sigma_{r_t 3}^2<\overline{r_t'^2}.
\end{aligned}$$

That is, for instance, that the variance of w over the whole pdf has to be strictly greater than δ times the squared standard deviation in w of the third normal distribution.

To make the resulting pdfs realizable, it turns out[Lar22], that we need to have $-1 < \widehat{c}_{w\theta_l}, \widehat{c}_{wr_t}, \widehat{c}_{r_t\theta_l} < 1$. This is for instance:

$$c_{wr_t}^2 < \left(1-\tilde{\sigma}_w^2\right)\left(\frac{(1-\delta\lambda_w)(1-\delta\lambda_r)}{(1-\delta\lambda_{wr})^2}\right). \tag{6.0.1}$$

So it might be safer to set

$$\lambda_w, \lambda_r < \lambda_{wr} \iff \left(\frac{\sigma_{w3}^2}{\overline{w'^2}} < \frac{\rho_{wr_t}\sigma_{w3}\sigma_{r_t3}}{\overline{w'r_t'}}\right) \wedge \left(\frac{\sigma_{r_t3}^2}{\overline{r_t'^2}} < \frac{\rho_{wr_t}\sigma_{w3}\sigma_{r_t3}}{\overline{w'r_t'}}\right)$$
$$\iff \left(\sigma_{w3}\overline{w'r_t'} < \rho_{wr_t}\sigma_{r_t3}\overline{w'^2}\right) \wedge \left(\sigma_{r_t3}\overline{w'r_t'} < \rho_{wr_t}\sigma_{w3}\overline{r_t'^2}\right) \tag{6.0.2}$$

so that the rhs is greater and the bound is less restrictive. If we assume $\lambda_\theta = \lambda_r$ and $\lambda_{w\theta} = \lambda_{wr}$, then the model gains 5 new pdf parameters: δ, λ_w, λ_θ, $\lambda_{w\theta}$, $\lambda_{\theta r}$. Considering that the skewness goes to zero ($Sk_w \to 0$), we want the pdf to revert to a single normal distribution. Therefore, we need

$$\delta, \lambda_w, \lambda_r, \lambda_\theta, \lambda_{wr}, \lambda_{w\theta}, \lambda_{\theta r} \to 1. \tag{6.0.3}$$

In this limit, there are no third-order moments anymore, so they have to go to 0 as well. Also, we want to have that the kurtosis is approaching 3 (value of the kurtosis of a standard normal distribution). To ensure those points, as well as no division by zero in the code, we can define the following properties:

$$\lim_{\delta\to 1}(1-\delta) \propto |Sk_w|, \tag{6.0.4}$$

which means that in the limit as $\delta \to 1$, $(1-\delta)$ should "behave as" the absolute value of the skewness of w,

$$0 < \lim_{\delta\to 1}\left(\frac{1-\delta\lambda_x}{1-\delta}\right) < \infty, \tag{6.0.5}$$

where x means any of w, r_t, or θ_l,

$$0 < \lim_{\delta\to 1}\left(\frac{1-\delta\lambda_x}{1-\delta\lambda_y}\right) < \infty \tag{6.0.6}$$

where x is the same as above and y means any of w, r_t, or θ_l, $x \neq y$,

$$0 < \lim_{\delta\to 1}\left(\tilde{\sigma}_w\right)^2 = \lim_{\delta\to 1}\left(\frac{\sigma_w^2}{\overline{w'^2}}\frac{1-\delta}{1-\delta\lambda_w}\right) < 1. \tag{6.0.7}$$

To ensure equation (6.0.4), we can use a linear "fit", which looks like

$$\lambda_w = \lambda_\theta = \lambda_q = (1-c_1)\delta + c_1 + \epsilon(1-c_1), \tag{6.0.8}$$

where c_1 is some constant and ϵ is a very small number to avoid a division by zero in any of the equations. The fit for the other λ's is

$$\lambda_{w\theta} = \lambda_{q\theta} = \lambda_{wq} = (1-c_2)\delta + c_2 - \epsilon(1-c_2), \tag{6.0.9}$$

where again c_2 is some constant and ϵ is the same as above. Note, that we already have a definition for the λ's (equation (4.1.4)). This definition is just for the backward run though, because we actually have to choose λ in the forward direction.

If we now look at the limit with the proposed fit (equation (6.0.8)), where we treat ϵ as zero, and x is one of the three variates, we get:

$$\begin{aligned}
\lim_{\delta\to 1}\left(\frac{1-\delta\lambda_x}{1-\delta}\right) &= \lim_{\delta\to 1}\left(\frac{1-\delta((1-c_1)\delta+c_1)}{1-\delta}\right) = \lim_{\delta\to 1}\left(\frac{1-((1-c_1)\delta^2+c_1\delta)}{1-\delta}\right) && (6.0.10)\\
&= \lim_{\delta\to 1}\left(\frac{1-\delta^2+c_1\delta^2-c_1\delta}{1-\delta}\right) = \lim_{\delta\to 1}\left(\frac{1-\delta^2+c_1\delta(\delta-1)}{1-\delta}\right) && (6.0.11)\\
&= \lim_{\delta\to 1}\left(\frac{1-\delta^2}{1-\delta}\right) - \lim_{\delta\to 1}\left(\frac{c_1\delta(1-\delta)}{1-\delta}\right) && (6.0.12)\\
\text{(L'Hôpital)} &\overset{\left[\frac{0}{0}\right]}{=} \lim_{\delta\to 1}\left(\frac{-2\delta}{-1}\right) - \lim_{\delta\to 1}(c_1\delta) && (6.0.13)\\
&= 2-c_1. && (6.0.14)
\end{aligned}$$

Then, we can also define the range of c_1, which should be $(0,2)$ because we want to have $0<\delta\lambda_w<1$. For the reciprocal, we then have

$$\begin{aligned}
\lim_{\delta\to 1}\left(\frac{1-\delta}{1-\delta\lambda_x}\right) &= \lim_{\delta\to 1}\left(\frac{1-\delta}{1-\delta^2+c_1\delta^2-c_1\delta}\right) \overset{\left[\frac{0}{0}\right]}{=} \lim_{\delta\to 1}\left(\frac{-1}{-2\delta+2c_1\delta-c_1}\right) && (6.0.15)\\
&= \frac{-1}{-2+c_1} = \frac{1}{2-c_1}. && (6.0.16)
\end{aligned}$$

Another limit also show up very often, which is

$$\lim_{\delta\to 1}\left(\frac{(1-\delta\lambda_x)^2}{1-\delta}\right) = \lim_{\delta\to 1}\left(\frac{(1-\delta\lambda_x)(1-\delta\lambda_x)}{1-\delta}\right) \tag{6.0.17}$$

$$= (2-c_1)\lim_{\delta\to 1}(1-\delta\lambda_x) \tag{6.0.18}$$

$$= (2-c_1)\lim_{\delta\to 1}(1-\delta^2+\delta^2 c_2-\delta c_2) = 0. \tag{6.0.19}$$

Since the formula for $\lambda_{w\theta} = \lambda_{q\theta} = \lambda_{wq}$ is the same just with a different constant c_2, we can also calculate another limit for a fraction which appears very frequently. Again, we use xy for $w\theta_l$, wr_t, or $\theta_l r_t$.

$$\lim_{\delta\to 1}\left(\frac{1-\delta\lambda_{xy}}{1-\delta\lambda_x}\right) = \lim_{\delta\to 1}\left(\frac{1-\delta((1-c_2)\delta+c_2)}{1-\delta((1-c_1)\delta+c_1)}\right) = \lim_{\delta\to 1}\left(\frac{1-\delta(\delta-\delta c_2+c_2)}{1-\delta(\delta-\delta c_1+c_1)}\right) \tag{6.0.20}$$

$$= \lim_{\delta\to 1}\left(\frac{1-\delta^2+\delta^2 c_2-\delta c_2}{1-\delta^2+\delta^2 c_1-\delta c_1}\right) \overset{\left[\frac{0}{0}\right]}{=} \lim_{\delta\to 1}\left(\frac{-2\delta+2\delta c_2-c_2}{-2\delta+2\delta c_1-c_1}\right) \tag{6.0.21}$$

$$= \lim_{\delta\to 1}\left(\frac{-2\delta+2\delta c_2-c_2}{-2\delta+2\delta c_1-c_1}\right) \tag{6.0.22}$$

$$= \frac{-2+c_1}{-2+c_2}, \tag{6.0.23}$$

For the reciprocal it is just

$$\lim_{\delta\to 1}\left(\frac{1-\delta\lambda_x}{1-\delta\lambda_{xy}}\right) = \frac{-2+c_2}{-2+c_1}. \tag{6.0.24}$$

We now proceed and calculate all the other limits. First, we use (based on equation (4.2.3))

$$\tilde{\sigma}_w^2 = \frac{\sigma_w^2(1-\delta)}{\overline{w'^2}(1-\delta\lambda_w)}, \tag{6.0.25}$$

and

$$\tilde{\sigma}_w^4 = \frac{\sigma_w^4(1-\delta)^2}{\left(\overline{w'^2}\right)^2(1-\delta\lambda_w)^2}. \tag{6.0.26}$$

We also treat the lower-order moments as fixed. To make the calculation even more readable, we calculate two limits here.

$$\lim_{\delta\to 1}\left(\tilde{\sigma}_w^2\right) = \lim_{\delta\to 1}\left(\frac{\sigma_w^2(1-\delta)}{\overline{w'^2}(1-\delta\lambda_w)}\right) = \left(\frac{1}{2-c_1}\right)\cdot\frac{\sigma_w^2}{\overline{w'^2}}, \tag{6.0.27}$$

and

$$\lim_{\delta\to 1}\left(\tilde{\sigma}_w^4\right) = \lim_{\delta\to 1}\left(\frac{\sigma_w^4(1-\delta)^2}{\left(\overline{w'^2}\right)^2(1-\delta\lambda_w)^2}\right) = \left(\frac{\sigma_w^4}{\left(\overline{w'^2}\right)^2}\right)\lim_{\delta\to 1}\left(\frac{1-\delta}{1-\delta\lambda_w}\right)^2 \tag{6.0.28}$$

$$= \left(\frac{1}{2-c_1}\right)^2 \cdot \left(\frac{\sigma_w^2}{\overline{w'^2}}\right)^2. \tag{6.0.29}$$

We want to see how the "main" equations behave in the limit of $\delta \to 1$. For those limits, we assume all lower-order moments as constants because they are just given to us in the code.

6.1 Limits for $\delta \to 1$

6.1.1 Limit for $\overline{w'^4}$ as δ goes to 1

$$\lim_{\delta\to 1}\left(\overline{w'^4}\right) = \left(\frac{\left(\overline{w'^3}\right)^2}{(1-\delta\lambda_w)\left(\overline{w'^2}-\frac{\sigma_w^2}{2-c_1}\right)}\right) + 3\left(\overline{w'^2}\right)^2 \tag{6.1.1}$$

Looking at this limit (for the calculation refer to appendix A.1), where we held the lower-order moments fixed and not depending on δ, unfortunately, we see that the fourth-order moment of w diverges.

6.1.2 Limit for $\overline{w'^2\theta_l'}$ as δ goes to 1

$$\lim_{\delta\to 1}\left(\overline{w'^2\theta_l'}\right) = \frac{(c_1-2)^2\overline{w'\theta_l'}\cdot\overline{w'^3}}{(c_2-2)\left((c_1-2)\overline{w'^2}+\sigma_w^2\right)} \tag{6.1.2}$$

This limit (for the calculation refer to appendix A.2) is finite, where one just needs to pay attention on how to choose c_1 and c_2 respectively.

6.1.3 Limit for $\overline{w'^2\theta'_l}$ as δ goes to 1

$$\lim_{\delta\to 1}\left(\overline{w'^2\theta'_l}\right) = \frac{1}{3}\left(\frac{2\overline{w'^3}\left(\overline{w'\theta'_l}\right)^2}{\left(\overline{w'^2}^2 - 2\overline{w'^2}\frac{\sigma_w^2}{2-c_1} + \left(\frac{\sigma_w^2}{2-c_1}\right)^2\right)} + \frac{\overline{w'^2}(2-c_2)}{\overline{w'\theta'_l}(2-c_1)} - \frac{\sigma_w^2\overline{\theta'^2_l}}{\overline{w'\theta'_l}(2-c_2)}\right) \quad (6.1.3)$$

This limit (for the calculation refer to appendix A.3) is at least finite.

6.1.4 Limit for $\overline{w'r'_t\theta'_l}$ as δ goes to 1

$$\begin{aligned}\lim_{\delta\to 1}\left(\overline{w'r'_t\theta'_l}\right) = \lim_{\delta\to 1}\Big((1-\delta)\alpha(w_1-\overline{w})\left[(r_{t1}-\overline{r_t})\left(\theta_{l1}-\overline{\theta_l}\right) + r_{r_t\theta_l}\sigma_{r_{t1}}\sigma_{\theta_{l1}}\right] \\ + (1-\delta)(1-\alpha)(w_2-\overline{w})\left[(r_{t2}-\overline{r_t})\left(\theta_{l2}-\overline{\theta_l}\right) + r_{r_t\theta_l}\sigma_{r_{t2}}\sigma_{\theta_{l2}}\right]\Big) \\ = \quad 0 \end{aligned} \quad (6.1.4)$$

7 Summary

This thesis explores the potential benefits of incorporating a third normal distribution into the CLUBB model, a framework used for parameterizing atmospheric processes. While the initial motivation for this exploration may not be immediately clear, section 2.1 delves into graphical illustrations that reveal distinct advantages associated with this trinormal representation for capturing specific atmospheric behaviors.

Following this initial groundwork, section 2.4 formally establishes the central objective of this research. That is, to demonstrate the continued validity of existing CLUBB model formulas within the context of the proposed trinormal distribution. Achieving this is based on using certain transformations, shown in section 2.3.

To validate the accuracy of the transformed formulas, a verification framework is established. This framework defines the inputs and outputs (section 2.5) associated with both the forward run (code used in the model) and the backward run (verification direction). This is followed by presenting a step-by-step verification procedure, ensuring that the transformed formulas are still valid.

The foundation for this investigation is laid out in chapter 3. This chapter provides a brief overview of pdfs, including both, the univariate and the multivariate normal distribution. Additionally, it explores the concepts of second, third, and fourth-order moments, which play a crucial role in characterizing certain statistical properties, but more importantly – at least for this work – the shapes of the distributions.

Building upon these definitions, section 4.1 introduces the newly proposed trinormal distribution, denoted as P_{tmg}. This distribution strategically positions the third normal component in-between the two existing components within the CLUBB model. Furthermore, the chapter establishes key properties associated with this new trinormal mixture pdf.

Chapter 4 details the transformation of the existing CLUBB formulas to work with the trinormal representation. This chapter serves as a comprehensive reference for the transformed equations employed throughout the thesis.

Following the establishment of the theoretical framework and the transformed formulas, chapter 5 delves into the actual calculation of the integrals. This chapter explores the application of a cas – specifically, SymPy – for tackling both, symbolic and numerical integration tasks.

The final chapter – chapter 6 – investigates the asymptotic behavior of the newly derived functions. This analysis shows on how the functions behave as their arguments approach specific values (e.g., one or zero). Additionally, the chapter identifies parameter value ranges or limitations that must be considered when working with these functions.

In essence, this thesis presents a comprehensive investigation into incorporating a trinormal distribution into the CLUBB model. The research demonstrates the compatibility of existing CLUBB model formulas with the proposed trinormal representation.

8 Outlook

For CLUBB there are new parameters, e.g. δ or λ_w, which can be chosen to "tweak" the representation of the underlying pdf for the prognosed moments. Ultimately one would like to fit this resulting pdf to real data, to get better relationships, as well as thresholds for some variables. To do this there are some approaches which unfortunately have not been discussed. A following thesis could e.g. incorporate some machine learning approach to learn optimal values for certain parameters.

The methodology established in chapter 5 extends far beyond the immediate application within the CLUBB model. This chapter serves as a blueprint for a generalizable approach to verifying and analyzing integral expressions. Its core strength lies in the utilization of SymPy, a powerful and well-supported cas. SymPy's community-driven nature provides continuous development and a vast library of mathematical capabilities. By leveraging this versatile tool, we can tackle a wide range of integral expressions, both analytically and numerically. This approach offers several advantages:

- *Symbolic Verification:* SymPy allows us to perform symbolic manipulations, enabling the derivation of exact solutions for integrals whenever possible.

- *Numerical Approximation:* For integrals that are analytically intractable, SymPy seamlessly integrates with numerical computing libraries. This allows us to efficiently approximate the integral's value for specific parameter choices. This combined approach ensures we can handle a broader range of integral expressions.

- *Generalizability and Reusability:* The framework outlined in chapter 5 is not specific to the context of CLUBB. By focusing on the core functionalities of SymPy, this approach can be adapted to various scientific disciplines.

Overall, the methods we developed in this thesis using SymPy are not just useful for the CLUBB model. These methods can be applied to many other scientific problems because they can both solve integrals exactly (symbolically) and get close answers (numerically) for a wide range of equations. SymPy, being a powerful and widely-used tool, makes this possible.

Bibliography

[Har+20] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[Ize08] Alan Julian Izenman. *Modern multivariate statistical techniques: regression, classification, and manifold learning*. Springer texts in statistics. OCLC: ocn225427579. New York ; London: Springer, 2008. ISBN: 9780387781884.

[Lar22] Vincent E. Larson. *CLUBB-SILHS: A parameterization of subgrid variability in the atmosphere*. 2022. arXiv: 1711.03675 [physics.ao-ph].

[LG05] Vincent E Larson and Jean-Christophe Golaz. "Using probability density functions to derive consistent closure relationships among higher-order moments". In: *Monthly Weather Review* 133.4 (2005), pp. 1023–1042.

[McK10] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[Meu+17] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://doi.org/10.7717/peerj-cs.103.

[Vir+20] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

A Calculation of limits

A.1 Calculation for the limit for $\overline{w'^4}$ as δ goes to 1

$$\lim_{\delta\to1}\left(\overline{w'^4}\right) = \lim_{\delta\to1}\left(\left(\overline{w'^2}\right)^2 \underbrace{\cancel{\frac{(1-\delta\lambda_w)^2}{(1-\delta)}}}_{\to 0}\left(3\tilde{\sigma}_w^4 + 6\left(1-\tilde{\sigma}_w^2\right)\tilde{\sigma}_w^2 + \left(1-\tilde{\sigma}_w^2\right)^2\right)\right.$$
$$\left. + \frac{1}{(1-\tilde{\sigma}_w^2)}\frac{1}{(1-\delta\lambda_w)}\frac{\left(\overline{w'^3}\right)^2}{\overline{w'^2}} + \delta 3\lambda_w^2\left(\overline{w'^2}\right)^2\right) \tag{A.1.1}$$

$$= \lim_{\delta\to1}\left(\frac{\left(\overline{w'^3}\right)^2}{\overline{w'^2}\left(\left(\frac{\overline{w'^2}(1-\delta\lambda_w)-\sigma_w^2(1-\delta)}{\overline{w'^2}(1-\delta\lambda_w)}\right)\right)(1-\delta\lambda_w)} + 3\delta\lambda_w^2\left(\overline{w'^2}\right)^2\right) \tag{A.1.2}$$

$$= \lim_{\delta\to1}\left(\frac{\left(\overline{w'^3}\right)^2}{\overline{w'^2}(1-\delta\lambda_w)-\sigma_w^2(1-\delta)} + 3\delta\lambda_w^2\left(\overline{w'^2}\right)^2\right) \tag{A.1.3}$$

$$= \lim_{\delta\to1}\left(\frac{\left(\overline{w'^3}\right)^2}{(1-\delta\lambda_w)\left(\overline{w'^2}-\sigma_w^2\underbrace{\left(\frac{1-\delta}{1-\delta\lambda_w}\right)}_{\to\frac{1}{2-c_1}}\right)} + 3\delta\lambda_w^2\left(\overline{w'^2}\right)^2\right) \tag{A.1.4}$$

$$= \lim_{\delta\to1}\left(\frac{\left(\overline{w'^3}\right)^2}{(1-\delta\lambda_w)\left(\overline{w'^2}-\frac{\sigma_w^2}{2-c_1}\right)}\right.$$
$$\left. + 3\delta\left(c_1^2\delta^2 - 2c_1^2\delta + c_1^2 - 2c_1\delta^2 + 2c_1\delta + \delta^2\right)\left(\overline{w'^2}\right)^2\right) \tag{A.1.5}$$

$$= \lim_{\delta\to1}\left(\frac{\left(\overline{w'^3}\right)^2}{(1-\delta\lambda_w)\left(\overline{w'^2}-\frac{\sigma_w^2}{2-c_1}\right)}\right) + 3\left(\overline{w'^2}\right)^2 \tag{A.1.6}$$

A.2 Calculation for the limit for $\overline{w'^2\theta'_l}$ as δ goes to 1

$$\lim_{\delta\to 1}\left(\overline{w'^2\theta'_l}\right) = \lim_{\delta\to 1}\left(\frac{1}{(1-\tilde{\sigma}_w^2)} \quad \frac{1-\delta\lambda_{w\theta}}{1-\delta\lambda_w}\frac{\overline{w'^3}}{\overline{\overline{w'^2}}}\overline{w'\theta'_l}\right) \tag{A.2.1}$$

$$= \lim_{\delta\to 1}\left(\frac{(1-\delta\lambda_{w\theta})\cdot\overline{w'^3}\cdot\overline{w'\theta'_l}}{(1-\delta\lambda_w)\overline{w'^2}(1-\frac{\sigma_w^2(1-\delta)}{\overline{w'^2}(1-\delta\lambda_w)})}\right) \tag{A.2.2}$$

$$= \lim_{\delta\to 1}\left(\underbrace{\left(\frac{1-\delta\lambda_{w\theta}}{1-\delta\lambda_w}\right)}_{\to\frac{-2+c_1}{-2+c_2}}\frac{\overline{w'^3}\cdot\overline{w'\theta'_l}}{\left(\overline{w'^2}-\sigma_w^2\underbrace{\left(\frac{1-\delta}{1-\delta\lambda_w}\right)}_{\to\frac{1}{2-c_1}}\right)}\right) \tag{A.2.3}$$

$$= \frac{-2+c_1}{-2+c_2}\cdot\frac{\overline{w'^3}\cdot\overline{w'\theta'_l}}{\overline{w'^2}-\frac{\sigma_w^2}{2-c_1}} = \frac{\overline{w'^3}\cdot\overline{w'\theta'_l}\cdot(c_1-2)}{(c_2-2)\overline{w'^2}-\frac{(c_2-2)\sigma_w^2}{2-c_1}} \tag{A.2.4}$$

$$= \frac{\overline{w'^3}\cdot\overline{w'\theta'_l}\cdot(c_1-2)}{\frac{(2-c_1)(\overline{w'^2}c_2-2\overline{w'^2})-\sigma_w^2c_2+2\sigma_w^2}{2-c_1}} = \frac{\overline{w'^3}\cdot\overline{w'\theta'_l}\cdot(c_1-2)}{\frac{2\overline{w'^2}c_2-4\overline{w'^2}-c_1c_2\overline{w'^2}+2c_1\overline{w'^2}-\sigma_w^2c_2+2\sigma_w^2}{2-c_1}} \tag{A.2.5}$$

$$= \frac{\overline{w'^3}\cdot\overline{w'\theta'_l}\cdot(c_1-2)\cdot(2-c_1)}{2\overline{w'^2}c_2-4\overline{w'^2}-c_1c_2\overline{w'^2}+2c_1\overline{w'^2}-\sigma_w^2c_2+2\sigma_w^2} \tag{A.2.6}$$

$$= \frac{\overline{w'^3}\cdot\overline{w'\theta'_l}\cdot(c_1-2)\cdot(2-c_1)}{(2-c_2)(c_1\overline{w'^2}+\sigma_w^2-2\overline{w'^2})} \tag{A.2.7}$$

$$= \frac{(c_1-2)^2\overline{w'\theta'_l}\cdot\overline{w'^3}}{(c_2-2)\left((c_1-2)\overline{w'^2}+\sigma_w^2\right)} \tag{A.2.8}$$

A.3 Calculation for the limit for $\overline{w'\theta'^2_l}$ as δ goes to 1

$$\lim_{\delta\to 1}\left(\overline{w'\theta'^2_l}\right) = \lim_{\delta\to 1}\left(\frac{2}{3}\frac{(1-\delta\lambda_{w\theta})^2}{(1-\delta\lambda_w)^2}\frac{1}{(1-\tilde{\sigma}_w^2)^2}\frac{\overline{w'^3}}{\left(\overline{w'^2}\right)^2}\left(\overline{w'\theta'_l}\right)^2\right.$$

$$
\left. + \frac{1}{3}\frac{(1-\delta\lambda_w)}{(1-\delta\lambda_{w\theta})}\left(1-\tilde{\sigma}_w^2\right)\frac{\overline{w'^2}\,\overline{\theta_l'^3}}{\overline{w'\theta_l'}}\right)
$$

$$
= \lim_{\delta\to 1}\left(\frac{2(1-\delta\lambda_{w\theta})^2\overline{w'^3}\left(\overline{w'\theta_l'}\right)^2}{3(1-\delta\lambda_w)^2\left(\frac{\overline{w'^2}(1-\delta\lambda_w)-\sigma_w^2(1-\delta)}{\overline{w'^2}(1-\delta\lambda_w)}\right)^2\left(\overline{w'^2}\right)^2}\right.
$$

$$
\left. + \frac{\overline{w'^2}(1-\delta\lambda_w)-\sigma_w^2(1-\delta)\overline{\theta_l'^2}}{3(1-\delta\lambda_{w\theta})\overline{w'\theta_l'}}\right) \tag{A.3.1}
$$

$$
= \lim_{\delta\to 1}\left(\frac{2(1-\delta\lambda_{w\theta})^2\overline{w'^3}\left(\overline{w'\theta_l'}\right)^2}{3\left(\overline{w'^2}^2(1-\delta\lambda_w)^2-2\overline{w'^2}(1-\delta\lambda_w)\sigma_w^2(1-\delta)+\sigma_w^4(1-\delta)^2\right)}\right.
$$

$$
\left. + \underbrace{\left(\frac{1-\delta\lambda_w}{1-\delta\lambda_{w\theta}}\right)}_{\to\frac{-2+c_2}{-2+c_1}}\frac{\overline{w'^2}}{3\overline{w'\theta_l'}} - \underbrace{\left(\frac{1-\delta}{1-\delta\lambda_{w\theta}}\right)}_{\to\frac{1}{2-c_2}}\frac{\sigma_w^2\overline{\theta_l'^2}}{3\overline{w'\theta_l'}}\right) \tag{A.3.2}
$$

$$
= \lim_{\delta\to 1}\left(\frac{2\overline{w'^3}\left(\overline{w'\theta_l'}\right)^2}{3\left(\overline{w'^2}^2-2\overline{w'^2}\sigma_w^2\underbrace{\left(\frac{1-\delta}{1-\delta\lambda_w}\right)}_{\to\frac{1}{2-c_1}}+\sigma_w^4\underbrace{\left(\frac{1-\delta}{1-\delta\lambda_w}\right)^2}_{\to\left(\frac{1}{2-c_1}\right)^2}\right)}\right.
$$

$$
\left. + \frac{-2+c_2}{-2+c_1}\cdot\frac{\overline{w'^2}}{3\overline{w'\theta_l'}} - \frac{1}{2-c_2}\cdot\frac{\sigma_w^2\overline{\theta_l'^2}}{3\overline{w'\theta_l'}}\right) \tag{A.3.3}
$$

$$
= \left(\frac{2\overline{w'^3}\left(\overline{w'\theta_l'}\right)^2}{3\left(\overline{w'^2}^2-2\overline{w'^2}\sigma_w^2\frac{1}{2-c_1}+\sigma_w^4\left(\frac{1}{2-c_1}\right)^2\right)} + \frac{-2+c_2}{-2+c_1}\cdot\frac{\overline{w'^2}}{3\overline{w'\theta_l'}} - \frac{1}{2-c_2}\cdot\frac{\sigma_w^2\overline{\theta_l'^2}}{3\overline{w'\theta_l'}}\right) \tag{A.3.4}
$$

$$
= \frac{1}{3}\left(\frac{2\overline{w'^3}\left(\overline{w'\theta_l'}\right)^2}{\left(\overline{w'^2}^2-2\overline{w'^2}\frac{\sigma_w^2}{2-c_1}+\left(\frac{\sigma_w^2}{2-c_1}\right)^2\right)} + \frac{\overline{w'^2}(2-c_2)}{\overline{w'\theta_l'}(2-c_1)} - \frac{\sigma_w^2\overline{\theta_l'^2}}{\overline{w'\theta_l'}(2-c_2)}\right) \tag{A.3.5}
$$

B Code

B.1 User Guide

B.1.1 Accessing the code

The code used for checking functions mentioned in the thesis is attached and can be accessed alongside this document.

B.1.2 Key files and their purposes

- checked_functions.py: This file hosts the definitions of all functions used for checking integrals.
- symbols.py: This file contains definitions of all symbols employed for computations with SymPy.

B.1.3 Working with functions

To obtain a function with symbols already incorporated, call it with an empty list of arguments (e.g., `function_name()`).

B.1.4 Displaying equations effectively

1. Import the `display` function from `IPython.display`.
2. Use `display(..)` to present statements or even equations visually.

B.1.5 Handling integrals

1. *Displaying an integral:* Use `sympy.Integral(..)` and put it into `display(..)` to visualize the integral.

2. *Computing an integral symbolically:* Apply `.doit(..)` to the integral object for symbolic computation.

3. *Approximating an integral numerically:* Add the option `"method=quad"` within the `.doit(..)` call to calculate the integral using the quadrature approximation method.

B.2 symbols.py

```python
from sympy import Symbol, symbols

# sigma
sigma_w = Symbol('\\sigma_w')

sigma_r_t_i, sigma_r_t_1, sigma_r_t_2 = (
    symbols('\\sigma_{r_{ti}} \\sigma_{r_{t1}} \\sigma_{r_{t2}}'))

sigma_theta_l_i, sigma_theta_l_1, sigma_theta_l_2 = (
    symbols('\\sigma_{\\theta_{li}} \\sigma_{\\theta_{l1}} \\sigma_{\\theta_{l2}}'))

sigma_tilde_r_t_i, sigma_tilde_r_t_1, sigma_tilde_r_t_2 = (
    symbols('\\tilde{\\sigma}_{r_ti} \\tilde{\\sigma}_{r_t1} \\tilde{\\sigma}_{r_t2}')
)

sigma_tilde_theta_l_i, sigma_tilde_theta_l_1, sigma_tilde_theta_l_2 = (
    symbols(
        '\\tilde{\\sigma}_{\\theta_li} \\tilde{\\sigma}_{\\theta_l1}
        ↪  \\tilde{\\sigma}_{\\theta_l2}')
)

sigma_tilde_w_3 = Symbol('\\tilde{\\sigma}_{w3}')
sigma_tilde_w = Symbol('\\tilde{\\sigma}_w')
sigma_w_3 = Symbol('\\sigma_{w3}')
sigma_theta_l_3 = Symbol('\\sigma_{\\theta_l 3}')
sigma_r_t_3 = Symbol('\\sigma_{r_t 3}')

# w
w_bar = Symbol('\\overline{w}')
w_prime = Symbol('w\'')
w_i, w_1, w_2 = symbols('w_i w_1 w_2')

w_hat_i, w_hat_1, w_hat_2, w_hat_3, w_hat_prime = (
    symbols('\\hat{w}_i \\hat{w}_1 \\hat{w}_2 \\hat{w}_3 \\hat{w}\'')
)

w_prime_2_bar, w_prime_3_bar, w_prime_4_bar = symbols(
    '\\overline{w\'^2} \\overline{w\'^3} \\overline{w\'^4}'
)

w_prime_r_t_prime_bar = Symbol('\\overline{w\'r\'_t}')
```

```python
w_prime_2_theta_prime_l_bar = Symbol('\\overline{w\'^2\\theta\'_l}')
w_prime_theta_prime_l_2_bar = Symbol('\\overline{w\'\\theta\'^2_l}')
w_prime_theta_l_prime_bar = Symbol('\\overline{w\'\\theta\'_l}')

w_prime_r_t_prime_theta_l_prime_bar = Symbol('\\overline{w\'{r_t}\'{\\theta_l}\'}')

# r
r_t = Symbol('r_t')

r_t_bar, r_t_prime_2_bar, r_t_prime_3_bar = symbols(
    '\\overline{r_t} \\overline{r_t\'^2} \\overline{r_t\'^3}'
)

r_t_i, r_t_1, r_t_2 = symbols('r_{ti} r_{t1} r_{t2}')

r_t_tilde_prime, r_t_i_tilde, r_t_1_tilde, r_t_2_tilde = symbols(
    '\\tilde{r_t}\' \\tilde{r_{ti}} \\tilde{r_{t1}} \\tilde{r_{t2}}'
)

r_t_prime_theta_l_prime_bar = Symbol('\\overline{r_t\'\\theta_l\'}')

# theta
theta_l = Symbol('\\theta_l')

theta_l_bar, theta_l_prime_2_bar, theta_l_prime_3_bar = symbols(
    '\\overline{\\theta_l} \\overline{\\theta_l\'^2} \\overline{\\theta_l\'^3}'
)

theta_l_i, theta_l_1, theta_l_2 = symbols(
    '\\theta_{li} \\theta_{l1} \\theta_{l2}'
)

theta_tilde_prime_l = Symbol('\\tilde{\\theta}_l\'')

theta_tilde_l_i, theta_tilde_l_1, theta_tilde_l_2 = symbols(
    '\\tilde{\\theta}_{li} \\tilde{\\theta}_{l1} \\tilde{\\theta}_{l2}'
)

# lambda
lambda_theta = Symbol('\\lambda_\\theta')
lambda_theta_r = Symbol('\\lambda_\\theta_r')
lambda_r = Symbol('\\lambda_r')
lambda_w = Symbol('\\lambda_w')
lambda_w_theta = Symbol('\\lambda_{w\\theta}')
lambda_w_r = Symbol('\\lambda_{wr}')

r_r_t_theta_l = Symbol('r_{r_t\\theta_l}')
triple_gaussian = Symbol('P_{tmg}(\\hat{w\'}, \\tilde{\\theta\'_l}, \\tilde{r\'_t})')
sk_w_hat = Symbol('\\widehat{Sk}_w')
sk_theta_l_hat = Symbol('\\widehat{Sk}_{\\theta_l}')
c_w_theta_l_hat = Symbol('\\widehat{c}_{w\\theta_l}')
sk_r_t_hat = Symbol('\\widehat{Sk}_{r_t}')
c_w_r_t_hat = Symbol('\\widehat{c}_{wr_t}')
```

```
beta_theta_l = Symbol('\\beta_{\\theta_l}')
beta_r_t = Symbol('\\beta_{r_t}')

G_3 = Symbol('G_3')
G_3_hat = Symbol('\\hat{G_3}')

G_w, G_1_w, G_2_w, G_3_w = symbols(
    'G_{w}(w) G_{1w}(w) G_{2w}(w) G_{3w}(w)'
)

G_w_1_2 = Symbol('G_{w_{12}}')

G_w_theta = Symbol('G_{w\\theta}(w,\\theta)')
G_1_w_theta = Symbol('G_{1w\\theta}(w,\\theta)')
G_2_w_theta = Symbol('G_{2w\\theta}(w,\\theta)')
G_3_w_theta = Symbol('G_{3w\\theta}(w,\\theta)')

G_theta_r = Symbol('G_{\\theta r}(\\theta, r)')
G_1_theta_r = Symbol('G_{1\\theta r}(\\theta, r)')
G_2_theta_r = Symbol('G_{2\\theta r}(\\theta, r)')
G_3_theta_r = Symbol('G_{3\\theta r}(\\theta, r)')

G_w_theta_l_r_t = Symbol('G_{w{\\theta_l}{r_t}}(w,{\\theta_l},{r_t})')
G_1_w_theta_l_r_t = Symbol('G_{1w{\\theta_l}{r_t}}(w,{\\theta_l},{r_t})')
G_2_w_theta_l_r_t = Symbol('G_{2w{\\theta_l}{r_t}}(w,{\\theta_l},{r_t})')
G_3_w_theta_l_r_t = Symbol('G_{3w{\\theta_l}{r_t}}(w,{\\theta_l},{r_t})')

G_theta, G_1_theta, G_2_theta, G_3_theta = symbols(
    'G_{\\theta}(\\theta) G_{1\\theta}(\\theta) G_{2\\theta}(\\theta)
    ↪  G_{3\\theta}(\\theta)'
)

rho_w_theta_l = Symbol('\\rho_{w\\theta_l}')
rho_w_r_t = Symbol('\\rho_{wr_t}')
rho_theta_l_r_t = Symbol('\\rho_{\\theta_lr_t}')
```

Listing B.1: symbols.py

B.3 checked_functions.py

```
import sympy as sp
from sympy import abc, Rational
from sympy.stats import Normal, density

import symbols as sym


# w equations
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```python

def w_bar(alpha=sp.abc.alpha, delta=sp.abc.delta, w_1=sym.w_1, w_2=sym.w_2):
    return ((1 - delta) * alpha * w_1
            + (1 - delta) * (1 - alpha) * w_2
            + delta * (alpha * w_1 + (1 - alpha) * w_2))


def w_prime_2_bar(alpha=sp.abc.alpha, delta=sp.abc.delta, w_1=sym.w_1, w_2=sym.w_2,
                  w_bar=sym.w_bar, sigma_w=sym.sigma_w, sigma_w_3=sym.sigma_w_3):
    return (((1 - delta) * alpha * ((w_1 - w_bar) ** 2 + sigma_w ** 2)) +
            ((1 - delta) * (1 - alpha) * ((w_2 - w_bar) ** 2 + sigma_w ** 2)) +
            (delta * sigma_w_3 ** 2))


def w_prime_3_bar(alpha=sp.abc.alpha, delta=sp.abc.delta, w_1=sym.w_1, w_2=sym.w_2,
                  w_bar=sym.w_bar, sigma_w=sym.sigma_w):
    return (((1 - delta) * alpha * ((w_1 - w_bar) ** 3 +
                                    3 * sigma_w ** 2 * (w_1 - w_bar))) +
            ((1 - delta) * (1 - alpha) * ((w_2 - w_bar) ** 3 +
                                          3 * sigma_w ** 2 * (w_2 - w_bar))))


def w_prime_4_bar(w_prime_2_bar=sym.w_prime_2_bar,
                  w_prime_3_bar=sym.w_prime_3_bar,
                  delta=sp.abc.delta,
                  sigma_tilde_w=sym.sigma_tilde_w,
                  sigma_w_3=sym.sigma_w_3):
    return (w_prime_2_bar ** 2 *
            ((1 - delta * (sigma_w_3 ** 2 / w_prime_2_bar)) ** 2 / (1 - delta)) *
            (3 * sigma_tilde_w ** 4 +
             6 * (1 - sigma_tilde_w ** 2) *
             sigma_tilde_w ** 2 +
             (1 - sigma_tilde_w ** 2) ** 2) +
            ((1 / (1 - sigma_tilde_w ** 2)) *
             (1 / (1 - delta * (sigma_w_3 ** 2 / w_prime_2_bar))) *
             (w_prime_3_bar ** 2 / w_prime_2_bar)) +
            (delta * 3 * sigma_w_3 ** 4))


# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

# theta_l equations
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

def theta_l_bar(alpha=sp.abc.alpha, delta=sp.abc.delta,
                theta_l_1=sym.theta_l_1, theta_l_2=sym.theta_l_2):
    return ((1 - delta) * alpha * theta_l_1
            + (1 - delta) * (1 - alpha) * theta_l_2
            + delta * (alpha * theta_l_1 + (1 - alpha) * theta_l_2))


def theta_l_prime_2_bar(alpha=sp.abc.alpha,
                        delta=sp.abc.delta,
                        theta_l_1=sym.theta_l_1,
```

```
                             theta_l_2=sym.theta_l_2,
                             theta_l_bar=sym.theta_l_bar,
                             sigma_theta_l_1=sym.sigma_theta_l_1,
                             sigma_theta_l_2=sym.sigma_theta_l_2,
                             sigma_theta_l_3_l=sym.sigma_theta_l_3):
    return (((1 - delta) * alpha * ((theta_l_1 - theta_l_bar) ** 2 + sigma_theta_l_1 **
    ↪  2)) +
            ((1 - delta) * (1 - alpha) *
             ((theta_l_2 - theta_l_bar) ** 2 + sigma_theta_l_2 ** 2)) +
            (delta * sigma_theta_l_3_l ** 2))


def theta_l_prime_3_bar(delta=sp.abc.delta,
                             alpha=sp.abc.alpha,
                             theta_l_1=sym.theta_l_1,
                             theta_l_2=sym.theta_l_2,
                             theta_l_bar=sym.theta_l_bar,
                             sigma_theta_l_1=sym.sigma_theta_l_1,
                             sigma_theta_l_2=sym.sigma_theta_l_2):
    return (((1 - delta) * alpha *
             ((theta_l_1 - theta_l_bar) ** 3 +
              3 * sigma_theta_l_1 ** 2 * (theta_l_1 - theta_l_bar))) +
            ((1 - delta) * (1 - alpha) *
             ((theta_l_2 - theta_l_bar) ** 3 +
              3 * sigma_theta_l_2 ** 2 * (theta_l_2 - theta_l_bar))))


# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

# r_t equations
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

def r_t_bar(alpha=sp.abc.alpha, delta=sp.abc.delta, r_t_1=sym.r_t_1, r_t_2=sym.r_t_2):
    return ((1 - delta) * alpha * r_t_1
            + (1 - delta) * (1 - alpha) * r_t_2
            + delta * (alpha * r_t_1 + (1 - alpha) * r_t_2))


def r_t_prime_2_bar(delta=sp.abc.delta,
                     alpha=sp.abc.alpha,
                     r_t_1=sym.r_t_1,
                     r_t_2=sym.r_t_2,
                     r_t_bar=sym.r_t_bar,
                     sigma_r_t_1=sym.sigma_r_t_1,
                     sigma_r_t_2=sym.sigma_r_t_2,
                     sigma_r_t_3_t=sym.sigma_r_t_3):
    return (((1 - delta) * alpha * ((r_t_1 - r_t_bar) ** 2 + sigma_r_t_1 ** 2)) +
            ((1 - delta) * (1 - alpha) * ((r_t_2 - r_t_bar) ** 2 + sigma_r_t_2 ** 2)) +
            (delta * sigma_r_t_3_t ** 2))


def r_t_prime_3_bar(alpha=sp.abc.alpha,
                     delta=sp.abc.delta,
                     r_t_1=sym.r_t_1,
```

```python
                     r_t_2=sym.r_t_2,
                     r_t_bar=sym.r_t_bar,
                     sigma_r_t_1=sym.sigma_r_t_1,
                     sigma_r_t_2=sym.sigma_r_t_2):
    return (((1 - delta) * alpha *
             ((r_t_1 - r_t_bar) ** 3 +
              3 * sigma_r_t_1 ** 2 * (r_t_1 - r_t_bar))) +
            ((1 - delta) * (1 - alpha) *
             ((r_t_2 - r_t_bar) ** 3 +
              3 * sigma_r_t_2 ** 2 * (r_t_2 - r_t_bar))))


# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

# Mixed equations
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

def w_prime_theta_l_prime_bar(delta=sp.abc.delta,
                              alpha=sp.abc.alpha,
                              w_1=sym.w_1,
                              w_2=sym.w_2,
                              w_bar=sym.w_bar,
                              theta_l_1=sym.theta_l_1,
                              theta_l_2=sym.theta_l_2,
                              theta_l_bar=sym.theta_l_bar,
                              cov_lambda_w_theta=sym.rho_w_theta_l * sym.sigma_w_3 *
                              ↪  sym.sigma_theta_l_3):
    return (((1 - delta) * alpha * ((w_1 - w_bar) * (theta_l_1 - theta_l_bar))) +
            ((1 - delta) * (1 - alpha) * ((w_2 - w_bar) * (theta_l_2 - theta_l_bar)))
            + delta * cov_lambda_w_theta)


def w_prime_r_t_prime_bar(alpha=sp.abc.alpha,
                          delta=sp.abc.delta,
                          w_1=sym.w_1,
                          w_2=sym.w_2,
                          w_bar=sym.w_bar,
                          r_t_1=sym.r_t_1,
                          r_t_2=sym.r_t_2,
                          r_t_bar=sym.r_t_bar,
                          cov_lambda_w_r=sym.rho_w_r_t * sym.sigma_w_3 *
                          ↪  sym.sigma_r_t_3):
    return (((1 - delta) * alpha * ((w_1 - w_bar) * (r_t_1 - r_t_bar))) +
            ((1 - delta) * (1 - alpha) * ((w_2 - w_bar) * (r_t_2 - r_t_bar)))
            + delta * cov_lambda_w_r)


def w_prime_2_theta_l_prime_bar(sigma_tilde_w=sym.sigma_tilde_w,
                                delta=sp.abc.delta,
                                lambda_w_theta=sym.lambda_w_theta,
                                lambda_w=sym.lambda_w,
                                w_prime_3_bar=sym.w_prime_3_bar,
                                w_prime_2_bar=sym.w_prime_2_bar,
                                w_prime_theta_l_prime=sym.w_prime_theta_l_prime_bar):
```

```
    return ((1 / (1 - sigma_tilde_w ** 2)) *
            ((1 - delta * lambda_w_theta) / (1 - delta * lambda_w)) *
            (w_prime_3_bar / w_prime_2_bar) *
            w_prime_theta_l_prime)


def w_prime_theta_l_prime_2_bar(delta=sp.abc.delta,
                                lambda_w_theta=sym.lambda_w_theta,
                                lambda_w=sym.lambda_w,
                                sigma_tilde_w=sym.sigma_tilde_w,
                                w_prime_3_bar=sym.w_prime_3_bar,
                                w_prime_2_bar=sym.w_prime_2_bar,
                                w_prime_theta_l_prime_bar=sym.w_prime_theta_l_prime_bar,
                                theta_l_prime_3_bar=sym.theta_l_prime_3_bar):
    return (Rational(2, 3) *
            ((1 - delta * lambda_w_theta) ** 2 / (1 - delta * lambda_w) ** 2) *
            (1 / (1 - sigma_tilde_w ** 2) ** 2) *
            (w_prime_3_bar / w_prime_2_bar ** 2) *
            w_prime_theta_l_prime_bar ** 2 +
            Rational(1, 3) *
            ((1 - delta * lambda_w) / (1 - delta * lambda_w_theta)) *
            (1 - sigma_tilde_w ** 2) *
            ((w_prime_2_bar * theta_l_prime_3_bar) / w_prime_theta_l_prime_bar))


def w_prime_theta_l_prime_2_bar_beta(
        sigma_tilde_w=sym.sigma_tilde_w,
        delta=sp.abc.delta,
        lambda_theta=sym.lambda_theta,
        lambda_w=sym.lambda_w,
        w_prime_3_bar=sym.w_prime_3_bar,
        w_prime_2_bar=sym.w_prime_2_bar,
        beta=sp.abc.beta,
        theta_l_prime_2_bar=sym.theta_l_prime_2_bar,
        lambda_w_theta=sym.lambda_w_theta,
        w_prime_theta_l_prime_bar=sym.w_prime_theta_l_prime_bar):
    from sympy import Rational
    return (
            (1 / (1 - sigma_tilde_w ** 2)) *
            ((1 - delta * lambda_theta) / (1 - delta * lambda_w)) *
            (w_prime_3_bar / w_prime_2_bar) *
            (
                    Rational(1, 3) * beta * theta_l_prime_2_bar +
                    (
                            ((1 - Rational(1, 3) * beta) / (1 - sigma_tilde_w ** 2)) *
                            ((1 - delta * lambda_w_theta) ** 2 /
                             ((1 - delta * lambda_w) * (1 - delta * lambda_theta))) *
                            (w_prime_theta_l_prime_bar ** 2 / w_prime_2_bar)
                    )
            )
    )


def w_prime_r_t_prime_theta_l_prime_bar(delta=sp.abc.delta,
```

```
                                        alpha=sp.abc.alpha,
                                        w_1=sym.w_1,
                                        w_2=sym.w_2,
                                        w_bar=sym.w_bar,
                                        r_t_1=sym.r_t_1,
                                        r_t_2=sym.r_t_2,
                                        r_t_bar=sym.r_t_bar,
                                        theta_l_1=sym.theta_l_1,
                                        theta_l_2=sym.theta_l_2,
                                        theta_l_bar=sym.theta_l_bar,
                                        r_r_t_theta_l=sym.r_r_t_theta_l,
                                        sigma_r_t_1=sym.sigma_r_t_1,
                                        sigma_r_t_2=sym.sigma_r_t_2,
                                        sigma_theta_l_1=sym.sigma_theta_l_1,
                                        sigma_theta_l_2=sym.sigma_theta_l_2):
    return (((1 - delta) * alpha * (w_1 - w_bar) *
             (
                     (r_t_1 - r_t_bar) * (theta_l_1 - theta_l_bar) +
                     r_r_t_theta_l * sigma_r_t_1 * sigma_theta_l_1
             )) +
            ((1 - delta) * (1 - alpha) * (w_2 - w_bar) *
             (
                     (r_t_2 - r_t_bar) * (theta_l_2 - theta_l_bar) +
                     r_r_t_theta_l * sigma_r_t_2 * sigma_theta_l_2
             )))


def w_prime_r_t_prime_theta_l_prime_bar_beta(
        beta=sp.abc.beta,
        sigma_tilde_w=sym.sigma_tilde_w,
        delta=sp.abc.delta,
        lambda_theta_r=sym.lambda_theta_r,
        lambda_w=sym.lambda_w,
        r_t_prime_theta_l_prime_bar=sym.r_t_prime_theta_l_prime_bar,
        w_prime_3_bar=sym.w_prime_3_bar,
        w_prime_2_bar=sym.w_prime_2_bar,
        lambda_w_r=sym.lambda_w_r,
        lambda_w_theta=sym.lambda_w_theta,
        w_prime_r_t_prime_bar=sym.w_prime_r_t_prime_bar,
        w_prime_theta_l_prime_bar=sym.w_prime_theta_l_prime_bar):
    from sympy import Rational
    return (
            (
                    ((Rational(1, 3) * beta) / (1 - sigma_tilde_w ** 2)) *
                    ((1 - delta * lambda_theta_r) / (1 - delta * lambda_w)) *
                    r_t_prime_theta_l_prime_bar *
                    (w_prime_3_bar / w_prime_2_bar)
            ) +
            (
                    ((1 - Rational(1, 3) * beta) / (1 - sigma_tilde_w ** 2) ** 2) *
                    (((1 - delta * lambda_w_r) * (1 - delta * lambda_w_theta)) /
                     ((1 - delta * lambda_w) ** 2)) *
                    w_prime_r_t_prime_bar *
                    w_prime_theta_l_prime_bar *
```

```
                    (w_prime_3_bar / w_prime_2_bar ** 2)
            )
    )


def w_prime_r_t_prime_theta_l_prime_bar_E(
        E=sp.abc.E,
        sigma_tilde_w=sym.sigma_tilde_w,
        delta=sp.abc.delta,
        lambda_theta_r=sym.lambda_theta_r,
        lambda_w=sym.lambda_w,
        r_t_prime_theta_l_prime_bar=sym.r_t_prime_theta_l_prime_bar,
        w_prime_3_bar=sym.w_prime_3_bar,
        w_prime_2_bar=sym.w_prime_2_bar,
        lambda_w_r=sym.lambda_w_r,
        lambda_w_theta=sym.lambda_w_theta,
        w_prime_r_t_prime_bar=sym.w_prime_r_t_prime_bar,
        w_prime_theta_l_prime_bar=sym.w_prime_theta_l_prime_bar):
    from sympy import Rational
    return (
            (
                    ((Rational(1, 2) * E) / (1 - sigma_tilde_w ** 2)) *
                    ((1 - delta * lambda_theta_r) / (1 - delta * lambda_w)) *
                    r_t_prime_theta_l_prime_bar *
                    (w_prime_3_bar / w_prime_2_bar)
            ) +
            (
                    ((1 - Rational(1, 2) * E) / (1 - sigma_tilde_w ** 2) ** 2) *
                    (((1 - delta * lambda_w_r) * (1 - delta * lambda_w_theta)) /
                     ((1 - delta * lambda_w) ** 2)) *
                    w_prime_r_t_prime_bar *
                    w_prime_theta_l_prime_bar *
                    (w_prime_3_bar / w_prime_2_bar ** 2)
            )
    )


def r_t_prime_theta_l_prime_bar(
        alpha=sp.abc.alpha, delta=sp.abc.delta,
        r_t_1=sym.r_t_1, r_t_2=sym.r_t_2, r_t_prime_bar=sym.r_t_bar,
        theta_l_1=sym.theta_l_1, theta_l_2=sym.theta_l_2,
        theta_l_bar=sym.theta_l_bar,
        r_r_t_theta_l=sym.r_r_t_theta_l,
        sigma_r_t_1=sym.sigma_r_t_1, sigma_r_t_2=sym.sigma_r_t_2,
        sigma_theta_l_1=sym.sigma_theta_l_1,
        sigma_theta_l_2=sym.sigma_theta_l_2,
        cov_lambda_r_theta=sym.rho_theta_l_r_t * sym.sigma_theta_l_3 * sym.sigma_r_t_3):
    return ((1 - delta) * alpha * (
            (r_t_1 - r_t_prime_bar) * (theta_l_1 - theta_l_bar) +
            r_r_t_theta_l * sigma_r_t_1 * sigma_theta_l_1) +
            ((1 - delta) * (1 - alpha) * (
                    (r_t_2 - r_t_prime_bar) * (theta_l_2 - theta_l_bar) +
                    r_r_t_theta_l * sigma_r_t_2 * sigma_theta_l_2)) +
            delta * cov_lambda_r_theta)
```

```python


# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

# Distributions
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

G_1_theta_l = Normal(name='G_1_theta_l', mean=sym.theta_l_1, std=sym.sigma_theta_l_1)
G_1_theta_l_density = density(G_1_theta_l)(sym.theta_l)

G_2_theta_l = Normal(name='G_2_theta_l', mean=sym.theta_l_2, std=sym.sigma_theta_l_2)
G_2_theta_l_density = density(G_2_theta_l)(sym.theta_l)

G_3_theta_l = Normal(name='G_3_theta_l', mean=sym.theta_l_bar, std=sym.sigma_theta_l_3)
G_3_theta_l_density = density(G_3_theta_l)(sym.theta_l)

G_theta = ((1 - sp.abc.delta) * sp.abc.alpha * G_1_theta_l_density +
           (1 - sp.abc.delta) * (1 - sp.abc.alpha) * G_2_theta_l_density +
           sp.abc.delta * G_3_theta_l_density)

# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

G_1_w = Normal(name='G_1_w', mean=sym.w_1, std=sym.sigma_w)
G_1_w_density = density(G_1_w)(sp.abc.w)

G_2_w = Normal(name='G_2_w', mean=sym.w_2, std=sym.sigma_w)
G_2_w_density = density(G_2_w)(sp.abc.w)

G_3_w = Normal(name='G_3_w', mean=sym.w_bar, std=sym.sigma_w_3)
G_3_w_density = density(G_3_w)(sp.abc.w)

G_w = ((1 - sp.abc.delta) * sp.abc.alpha * G_1_w_density +
       (1 - sp.abc.delta) * (1 - sp.abc.alpha) * G_2_w_density +
       sp.abc.delta * G_3_w_density)

# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

G_1_w_theta = Normal(name='G_1_w_theta', mean=sp.Matrix([sym.w_1, sym.theta_l_1]),
                     std=sp.Matrix([[sym.sigma_w ** 2, 0], [0, sym.sigma_theta_l_1 **
                     ↪  2]]))
G_1_w_theta_density = density(G_1_w_theta)(sp.abc.w, sp.abc.theta)

G_2_w_theta = Normal(name='G_2_w_theta', mean=sp.Matrix([sym.w_2, sym.theta_l_2]),
                     std=sp.Matrix([[sym.sigma_w ** 2, 0], [0, sym.sigma_theta_l_2 **
                     ↪  2]]))
G_2_w_theta_density = density(G_2_w_theta)(sp.abc.w, sp.abc.theta)

G_3_w_theta = Normal(name='G_3_w_theta', mean=sp.Matrix([sym.w_bar, sym.theta_l_bar]),
                     std=sp.Matrix([
                         [sym.sigma_w_3 ** 2,
                          sym.rho_w_theta_l * sym.sigma_w_3 * sym.sigma_theta_l_3],
                         [sym.rho_w_theta_l * sym.sigma_w_3 * sym.sigma_theta_l_3,
                          sym.sigma_theta_l_3 ** 2]
                     ]))
```

```python
G_3_w_theta_density = sp.simplify(density(G_3_w_theta)(sp.abc.w, sp.abc.theta))

G_w_theta = ((1 - sp.abc.delta) * sp.abc.alpha * G_1_w_theta_density +
             (1 - sp.abc.delta) * (1 - sp.abc.alpha) * G_2_w_theta_density +
             sp.abc.delta * G_3_w_theta_density)

# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

mu_1_theta_l_r_t = sp.Matrix([sym.theta_l_1, sym.r_t_1])
Sigma_1_theta_l_r_t = sp.Matrix([[sym.sigma_theta_l_1 ** 2,
                                  sym.r_r_t_theta_l * sym.sigma_theta_l_1 *
                                  ↪  sym.sigma_r_t_1],
                                 [sym.r_r_t_theta_l * sym.sigma_theta_l_1 *
                                 ↪  sym.sigma_r_t_1,
                                  sym.sigma_r_t_1 ** 2]])

G_1_theta_l_r_t = Normal(name='G_1_theta_l_r_t',
                         mean=mu_1_theta_l_r_t,
                         std=Sigma_1_theta_l_r_t)

G_1_theta_l_r_t_density = density(G_1_theta_l_r_t)(sym.theta_l, sym.r_t)

mu_2_theta_l_r_t = sp.Matrix([sym.theta_l_2, sym.r_t_2])
Sigma_2_theta_l_r_t = sp.Matrix(
    [[sym.sigma_theta_l_2 ** 2,
      sym.r_r_t_theta_l * sym.sigma_theta_l_2 * sym.sigma_r_t_2],
     [sym.r_r_t_theta_l * sym.sigma_theta_l_2 * sym.sigma_r_t_2,
      sym.sigma_r_t_2 ** 2]])

G_2_theta_l_r_t = Normal(name='G_2_theta_l_r_t',
                         mean=mu_2_theta_l_r_t,
                         std=Sigma_2_theta_l_r_t)

G_2_theta_l_r_t_density = density(G_2_theta_l_r_t)(sym.theta_l, sym.r_t)

mu_3_theta_l_r_t = sp.Matrix([sym.theta_l_bar, sym.r_t_bar])
Sigma_3_theta_l_r_t = sp.Matrix(
    [[sym.sigma_theta_l_3 ** 2,
      sym.rho_theta_l_r_t * sym.sigma_theta_l_3 * sym.sigma_r_t_3],
     [
         sym.rho_theta_l_r_t * sym.sigma_theta_l_3 * sym.sigma_r_t_3,
         sym.sigma_r_t_3 ** 2]])

G_3_theta_l_r_t = Normal(name='G_3_theta_l_r_t',
                         mean=mu_3_theta_l_r_t,
                         std=Sigma_3_theta_l_r_t)

G_3_theta_l_r_t_density = density(G_3_theta_l_r_t)(sym.theta_l, sym.r_t)

G_theta_l_r_t = ((1 - sp.abc.delta) * sp.abc.alpha * G_1_theta_l_r_t_density +
                 (1 - sp.abc.delta) * (1 - sp.abc.alpha) * G_2_theta_l_r_t_density +
                 sp.abc.delta * G_3_theta_l_r_t_density)

# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```python

mu_1_w_theta_l_r_t = sp.Matrix([sym.w_1, sym.theta_l_1, sym.r_t_1])
Sigma_1_w_theta_l_r_t = sp.Matrix(
    [[sym.sigma_w ** 2,
      0,
      0],
     [0,
      sym.sigma_theta_l_1 ** 2,
      sym.r_r_t_theta_l * sym.sigma_theta_l_1 * sym.sigma_r_t_1],
     [0,
      sym.r_r_t_theta_l * sym.sigma_theta_l_1 * sym.sigma_r_t_1,
      sym.sigma_r_t_1 ** 2]])

G_1_w_theta_l_r_t = Normal(name='G_1_w_theta_l_r_t',
                           mean=mu_1_w_theta_l_r_t,
                           std=Sigma_1_w_theta_l_r_t)

G_1_w_theta_l_r_t_density = density(G_1_w_theta_l_r_t)(sp.abc.w, sym.theta_l, sym.r_t)

mu_2_w_theta_l_r_t = sp.Matrix([sym.w_2, sym.theta_l_2, sym.r_t_2])
Sigma_2_w_theta_l_r_t = sp.Matrix([[sym.sigma_w ** 2,
                                    0,
                                    0],
                                   [0,
                                    sym.sigma_theta_l_2 ** 2,
                                    sym.r_r_t_theta_l * sym.sigma_theta_l_2 *
                                    ↪  sym.sigma_r_t_2],
                                   [0,
                                    sym.r_r_t_theta_l * sym.sigma_theta_l_2 *
                                    ↪  sym.sigma_r_t_2,
                                    sym.sigma_r_t_2 ** 2]])

G_2_w_theta_l_r_t = Normal(name='G_2_w_theta_l_r_t',
                           mean=mu_2_w_theta_l_r_t,
                           std=Sigma_2_w_theta_l_r_t)

G_2_w_theta_l_r_t_density = density(G_2_w_theta_l_r_t)(sp.abc.w, sym.theta_l, sym.r_t)

mu_3_w_theta_l_r_t = sp.Matrix([sym.w_bar, sym.theta_l_bar, sym.r_t_bar])
Sigma_3_w_theta_l_r_t = sp.Matrix(
    [[sym.sigma_w_3 ** 2,
      sym.rho_w_theta_l * sym.sigma_w_3 * sym.sigma_theta_l_3,
      sym.rho_w_r_t * sym.sigma_theta_l_3 * sym.sigma_r_t_3],
     [
         sym.rho_w_theta_l * sym.sigma_w_3 * sym.sigma_theta_l_3,
         sym.sigma_theta_l_3 ** 2,
         sym.rho_theta_l_r_t * sym.sigma_w_3 * sym.sigma_r_t_3],
     [sym.rho_w_r_t * sym.sigma_theta_l_3 * sym.sigma_r_t_3,
      sym.rho_theta_l_r_t * sym.sigma_w_3 * sym.sigma_r_t_3,
      sym.sigma_r_t_3 ** 2]])

G_3_w_theta_l_r_t = Normal(name='G_3_w_theta_l_r_t',
                           mean=mu_3_w_theta_l_r_t,
                           std=Sigma_2_w_theta_l_r_t)
```

```python
G_3_w_theta_l_r_t_density = density(G_3_w_theta_l_r_t)(sp.abc.w, sym.theta_l, sym.r_t)

G_w_theta_l_r_t = ((1 - sp.abc.delta) * sp.abc.alpha * G_1_w_theta_l_r_t_density +
                   (1 - sp.abc.delta) * (1 - sp.abc.alpha) * G_2_w_theta_l_r_t_density +
                   sp.abc.delta * G_3_w_theta_l_r_t_density)


# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

# sigma equations
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

def sigma_tilde_w(sigma_w=sym.sigma_w, w_prime_2_bar=sym.w_prime_2_bar,
                  delta=sp.abc.delta, lambda_w=sym.lambda_w):
    from sympy import sqrt
    return ((sigma_w / sqrt(w_prime_2_bar)) *
            (1 / sqrt((1 - delta * lambda_w) / (1 - delta))))


def sigma_tilde_r_t_1(sigma_r_t_1=sym.sigma_r_t_1, r_t_prime_2_bar=sym.r_t_prime_2_bar,
                      delta=sp.abc.delta, lambda_r=sym.lambda_r):
    from sympy import sqrt
    return ((sigma_r_t_1 / sqrt(r_t_prime_2_bar)) *
            (1 / sqrt((1 - delta * lambda_r) / (1 - delta))))


def sigma_tilde_r_t_2(sigma_r_t_2=sym.sigma_r_t_2, r_t_prime_2_bar=sym.r_t_prime_2_bar,
                      delta=sp.abc.delta, lambda_r=sym.lambda_r):
    from sympy import sqrt
    return ((sigma_r_t_2 / sqrt(r_t_prime_2_bar)) *
            (1 / sqrt((1 - delta * lambda_r) / (1 - delta))))


def sigma_tilde_theta_l_1(sigma_theta_l_1=sym.sigma_theta_l_1,
                          theta_l_prime_2_bar=sym.theta_l_prime_2_bar,
                          delta=sp.abc.delta, lambda_theta=sym.lambda_theta):
    from sympy import sqrt
    return ((sigma_theta_l_1 / sqrt(theta_l_prime_2_bar)) *
            (1 / sqrt((1 - delta * lambda_theta) / (1 - delta))))


def sigma_tilde_theta_l_2(sigma_theta_l_2=sym.sigma_theta_l_2,
                          theta_l_prime_2_bar=sym.theta_l_prime_2_bar,
                          delta=sp.abc.delta, lambda_theta=sym.lambda_theta):
    from sympy import sqrt
    return ((sigma_theta_l_2 / sqrt(theta_l_prime_2_bar)) *
            (1 / sqrt((1 - delta * lambda_theta) / (1 - delta))))


# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

# lambda equations
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

```

```python
def lambda_w(sigma_w_3=sym.sigma_w_3, w_prime_2_bar=sym.w_prime_2_bar):
    return sigma_w_3 ** 2 / w_prime_2_bar


def lambda_theta(sigma_theta_l_3=sym.sigma_theta_l_3,
                 theta_l_prime_2_bar=sym.theta_l_prime_2_bar):
    return sigma_theta_l_3 ** 2 / theta_l_prime_2_bar


def lambda_r(sigma_r_t_3=sym.sigma_r_t_3,
             r_t_prime_2_bar=sym.r_t_prime_2_bar):
    return sigma_r_t_3 ** 2 / r_t_prime_2_bar


def lambda_w_theta(cov_lambda_w_theta=sym.rho_w_theta_l * sym.sigma_w_3 *
↪  sym.sigma_theta_l_3,
                   w_prime_theta_l_prime_bar=sym.w_prime_theta_l_prime_bar):
    return cov_lambda_w_theta / w_prime_theta_l_prime_bar


def lambda_w_r(cov_lambda_w_r=sym.rho_w_r_t * sym.sigma_w_3 * sym.sigma_r_t_3,
               w_prime_r_t_prime_bar=sym.w_prime_r_t_prime_bar):
    return cov_lambda_w_r / w_prime_r_t_prime_bar


def lambda_r_theta(cov_lambda_r_theta=sym.rho_theta_l_r_t * sym.sigma_r_t_3 *
↪  sym.sigma_theta_l_3,
                   r_t_prime_theta_l_prime_bar=sym.r_t_prime_theta_l_prime_bar):
    return cov_lambda_r_theta / r_t_prime_theta_l_prime_bar


# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

# sk
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

def sk_theta_l_hat_beta(sk_w_hat=sym.sk_w_hat, c_hat_w_theta_l=sym.c_w_theta_l_hat,
                        beta=sp.abc.beta):
    return sk_w_hat * c_hat_w_theta_l * (beta + (1 - beta) * c_hat_w_theta_l ** 2)


def sk_theta_l_hat(theta_l_prime_3_bar=sym.theta_l_prime_3_bar,
                   theta_l_prime_2_bar=sym.theta_l_prime_2_bar,
                   delta=sp.abc.delta,
                   lambda_theta=sym.lambda_theta):
    from sympy import Rational
    return ((theta_l_prime_3_bar / theta_l_prime_2_bar ** Rational(3, 2)) *
            (1 / ((1 - delta * lambda_theta) / (1 - delta)) ** Rational(3, 2)) *
            (1 / (1 - delta)))


def sk_r_t_hat(r_t_prime_3_bar=sym.r_t_prime_3_bar,
               r_t_prime_2_bar=sym.r_t_prime_2_bar,
               delta=sp.abc.delta,
```

```
                lambda_r=sym.lambda_r):
    from sympy import Rational
    return ((r_t_prime_3_bar / r_t_prime_2_bar ** Rational(3, 2)) *
            (1 / ((1 - delta * lambda_r) / (1 - delta)) ** Rational(3, 2)) *
            (1 / (1 - delta)))


def sk_w_hat(sigma_tilde_w=sym.sigma_tilde_w,
             w_prime_3_bar=sym.w_prime_3_bar,
             w_prime_2_bar=sym.w_prime_2_bar,
             delta=sp.abc.delta,
             lambda_w=sym.lambda_w):
    from sympy import Rational
    return ((1 / (1 - sigma_tilde_w ** 2) ** Rational(3, 2)) *
            (w_prime_3_bar / (w_prime_2_bar ** Rational(3, 2))) *
            (1 / ((1 - delta * lambda_w) / (1 - delta)) ** Rational(3, 2)) *
            (1 / (1 - delta)))


# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

# c

def c_w_theta_l_hat(sigma_w_tilde=sym.sigma_tilde_w,
                    w_prime_theta_l_prime_bar=sym.w_prime_theta_l_prime_bar,
                    w_prime_2_bar=sym.w_prime_2_bar,
                    theta_l_prime_2_bar=sym.theta_l_prime_2_bar,
                    delta=sp.abc.delta,
                    lambda_w=sym.lambda_w,
                    lambda_theta=sym.lambda_theta,
                    lambda_w_theta=sym.lambda_w_theta):
    from sympy import sqrt
    return ((1 / sqrt(1 - sigma_w_tilde ** 2)) *
            (w_prime_theta_l_prime_bar / ((sqrt(w_prime_2_bar)) *
            ↪  sqrt(theta_l_prime_2_bar))) *
            (1 / sqrt((1 - delta * lambda_w) / (1 - delta))) *
            (1 / sqrt((1 - delta * lambda_theta) / (1 - delta))) *
            ((1 - delta * lambda_w_theta) / (1 - delta)))


def c_w_r_t_hat(sigma_w_tilde=sym.sigma_tilde_w,
                w_prime_r_t_prime_bar=sym.w_prime_r_t_prime_bar,
                w_prime_2_bar=sym.w_prime_2_bar,
                r_t_prime_2_bar=sym.r_t_prime_2_bar,
                delta=sp.abc.delta,
                lambda_w=sym.lambda_w,
                lambda_r=sym.lambda_r,
                lambda_w_r=sym.lambda_w_r):
    from sympy import sqrt
    return ((1 / sqrt(1 - sigma_w_tilde ** 2)) *
            (w_prime_r_t_prime_bar / ((sqrt(w_prime_2_bar)) * sqrt(r_t_prime_2_bar))) *
            (1 / sqrt((1 - delta * lambda_w) / (1 - delta))) *
            (1 / sqrt((1 - delta * lambda_r) / (1 - delta))) *
            ((1 - delta * lambda_w_r) / (1 - delta)))
```

```
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

# beta
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

def beta_w_theta_l(c_w_theta_l_hat=sym.c_w_theta_l_hat,
                   sk_w_hat=sym.sk_w_hat,
                   sk_theta_l_hat=sym.sk_theta_l_hat):
    return (((c_w_theta_l_hat ** 3 * sk_w_hat) - sk_theta_l_hat) /
            (c_w_theta_l_hat * sk_w_hat * (c_w_theta_l_hat ** 2 - 1)))


def beta_w_r_t(c_w_r_t_hat=sym.c_w_r_t_hat,
               sk_w_hat=sym.sk_w_hat,
               sk_r_t_hat=sym.sk_r_t_hat):
    return (((c_w_r_t_hat ** 3 * sk_w_hat) - sk_r_t_hat) /
            (c_w_r_t_hat * sk_w_hat * (c_w_r_t_hat ** 2 - 1)))


# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

# E
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

def E(alpha=sp.abc.alpha, xi=sp.abc.xi):
    from sympy import Rational
    return ((1 - Rational(1, 2) * ((2 * alpha) / (1 - 2 * alpha)) * xi) /
            (1 + Rational(1, 2) * xi))


# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --


# xi
# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

def xi(alpha=sp.abc.alpha, sigma_tilde_r_t_1=sym.sigma_tilde_r_t_1,
       sigma_tilde_r_t_2=sym.sigma_tilde_r_t_2,
       ↪  sigma_tilde_theta_l_1=sym.sigma_tilde_theta_l_1,
       sigma_tilde_theta_l_2=sym.sigma_tilde_theta_l_2):
    return (((1 - alpha) / alpha) *
            (sigma_tilde_r_t_1 / sigma_tilde_r_t_2) *
            (sigma_tilde_theta_l_1 / sigma_tilde_theta_l_2) - 1)

# -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Listing B.2: checked_functions.py

B.4 w_prime_4_bar.py

```python
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import sympy as sp
from IPython.display import display
from sympy import abc, oo, init_printing

import checked_functions as c_f
import symbols as sym
init_printing()


# # This document aims to analytically check £\overline{w'^4}£

# ## Define the marginal distributions with those parameters.

# In[2]:


display(sp.Eq(sym.G_1_w, c_f.G_1_theta_l_density))


# In[3]:


display(sp.Eq(sym.G_2_w, c_f.G_2_theta_l_density))


# In[4]:


display(sp.Eq(sym.G_3_w, c_f.G_3_theta_l_density))


# In[5]:


display(sp.Eq(sym.G_w, c_f.G_w))


# Calculate the moment analytically:

# In[6]:


w_prime_4_bar_int = sp.Integral((sp.abc.w - sym.w_bar) ** 4 * c_f.G_w, [sp.abc.w, -oo,
↪  oo])
```

```python
display(sp.Eq(sym.w_prime_4_bar, w_prime_4_bar_int))


# In[7]:


w_prime_4_bar_int_val = w_prime_4_bar_int.doit(conds='none').simplify()
display(sp.Eq(sym.w_prime_4_bar, w_prime_4_bar_int_val))


# The equation in the document is:

# In[8]:


display(sp.Eq(sym.w_prime_4_bar, c_f.w_prime_4_bar()))


# where

# In[9]:


display(sp.Eq(sym.sigma_tilde_w, c_f.sigma_tilde_w()))


# and

# In[10]:


display(sp.Eq(sym.w_prime_2_bar, c_f.w_prime_2_bar()))


# and

# In[11]:


display(sp.Eq(sym.w_prime_3_bar, c_f.w_prime_2_bar()))


# So,

# In[12]:


lambda_w_val = c_f.lambda_w().subs({
    sym.w_prime_2_bar: c_f.w_prime_2_bar()
})
display(sp.Eq(sym.lambda_w, lambda_w_val))


# In[13]:
```

```python


w_prime_4_bar_check_val = c_f.w_prime_4_bar().subs({
    sym.w_prime_2_bar: c_f.w_prime_2_bar(),
    sym.w_prime_3_bar: c_f.w_prime_3_bar(),
    sym.sigma_tilde_w: c_f.sigma_tilde_w().subs({
        sym.w_prime_2_bar: c_f.w_prime_2_bar(),
        sym.lambda_w: c_f.lambda_w()
    })
})

display(sp.Eq(sym.w_prime_4_bar, w_prime_4_bar_check_val))


# In[14]:


display(sp.Eq(w_prime_4_bar_int_val, w_prime_4_bar_check_val, evaluate=True)
        .subs({sym.w_bar: c_f.w_bar()}).simplify())

```

Listing B.3: w_prime_4_bar.py

B.5 w_prime_2_theta_l_prime_bar.py

```python
#!/usr/bin/env python
# coding: utf-8

# In[1]:


from itertools import product

import pandas as pd

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import sympy as sp
from IPython.display import display
from sympy import abc, oo, Rational

import checked_functions as c_f
import symbols as sym


# # This document aims to numerically check £\overline{w'^2\theta_l'}£

# ## Define the marginal distributions.
```

```
# In[2]:


display(sp.Eq(sym.G_1_w_theta, c_f.G_1_w_theta_density))


# In[3]:


display(sp.Eq(sym.G_2_w_theta, c_f.G_2_w_theta_density))


# In[4]:


display(sp.Eq(sym.G_3_w_theta, c_f.G_3_w_theta_density))


# In[5]:


display(sp.Eq(sym.G_w_theta, c_f.G_w_theta, evaluate=False))


# In[6]:


w_prime_2_theta_l_prime_bar = sp.Integral(
    (sp.abc.w - sym.w_bar) ** 2 * (sp.abc.theta - sym.theta_l_bar) * c_f.G_w_theta,
    [sp.abc.w, -oo, oo],
    [sp.abc.theta, -oo, oo])
display(sp.Eq(sym.w_prime_2_theta_prime_l_bar, w_prime_2_theta_l_prime_bar))


# In[7]:


w_prime_2_theta_l_prime_bar = (
    w_prime_2_theta_l_prime_bar.subs({
        sym.w_bar: c_f.w_bar(),
        sym.theta_l_bar: c_f.theta_l_bar()
    }))

display(sp.Eq(sym.w_prime_2_theta_prime_l_bar,
              w_prime_2_theta_l_prime_bar))


# The equation in the document is:

# In[8]:


display(sp.Eq(sym.w_prime_2_theta_prime_l_bar, c_f.w_prime_2_theta_l_prime_bar()))
```

```
# For this we still need £\tilde{\sigma_w}£:

# In[9]:


display(sp.Eq(sym.sigma_tilde_w, c_f.sigma_tilde_w()))


# And £\overline{w'^{2}}£:

# In[10]:


display(sp.Eq(sym.w_prime_2_bar, c_f.w_prime_2_bar()))


# And £\overline{w'^{3}}£:

# In[11]:


display(sp.Eq(sym.w_prime_3_bar, c_f.w_prime_3_bar()))


# And £\lambda_w£:

# In[12]:


display(sp.Eq(sym.lambda_w, c_f.lambda_w()))


# And £\lambda_{w\theta}£:

# In[13]:


display(sp.Eq(sym.lambda_w_theta, c_f.lambda_w_theta()))


# And £\overline{w'\theta_l'}£:

# In[14]:


display(sp.Eq(sym.w_prime_theta_l_prime_bar, c_f.w_prime_theta_l_prime_bar()))


# Putting those all together yields:

# In[15]:

```

```python

w_prime_2_theta_l_prime_bar_check_val = (
    c_f.w_prime_2_theta_l_prime_bar().subs({
        sym.sigma_tilde_w: c_f.sigma_tilde_w(),
        sym.lambda_w: c_f.lambda_w(),
        sym.w_prime_2_bar: c_f.w_prime_2_bar(),
        sym.w_prime_3_bar: c_f.w_prime_3_bar(),
        sym.lambda_w_theta: c_f.lambda_w_theta(),
        sym.w_prime_theta_l_prime_bar: c_f.w_prime_theta_l_prime_bar()
    }))

w_prime_2_theta_l_prime_bar_check_val = (
    w_prime_2_theta_l_prime_bar_check_val.subs({
        sym.w_prime_2_bar: c_f.w_prime_2_bar(),
        sym.w_bar: c_f.w_bar(),
        sym.theta_l_bar: c_f.theta_l_bar()
    }))

display(sp.Eq(sym.w_prime_2_theta_prime_l_bar,
              w_prime_2_theta_l_prime_bar_check_val))


# Since the integral is too difficult to be calculated analytically, at least with sympy, we try to put in some arbitrary numbers for the pdf parameters, to simplify the equations.

# We create a dataframe to get all possible permutations and therefore also all possible evaluations of the integrals.

# In[16]:


df = pd.DataFrame(
    product([0, 1],
            [-2, 2],
            [-1, 2],
            [0, 3],
            [Rational(1, 10)],
            [Rational(3, 10)],
            [Rational(4, 10)],
            [Rational(7, 10)],
            [Rational(6, 10)],
            [Rational(5, 10)],
            [Rational(1, 10), Rational(5, 10)],
            [Rational(5, 10)]),
    columns=[sym.w_1,
             sym.w_2,
             sym.theta_l_1,
             sym.theta_l_2,
             sym.sigma_theta_l_1,
             sym.sigma_theta_l_2,
             sym.sigma_lambda_theta_l,
             sym.sigma_w,
             sym.sigma_lambda_w,
```

```
            sp.abc.alpha,
            sp.abc.delta,
            sym.rho_w_theta_l])


# In[17]:


df['check_val'] = (
    df.apply(lambda x:
             w_prime_2_theta_l_prime_bar_check_val.subs({
                 sym.w_1: x[sym.w_1],
                 sym.w_2: x[sym.w_2],
                 sym.theta_l_1: x[sym.theta_l_1],
                 sym.theta_l_2: x[sym.theta_l_2],
                 sym.sigma_theta_l_1: x[sym.sigma_theta_l_1],
                 sym.sigma_theta_l_2: x[sym.sigma_theta_l_2],
                 sym.sigma_lambda_theta_l: x[sym.sigma_lambda_theta_l],
                 sym.sigma_w: x[sym.sigma_w],
                 sym.sigma_lambda_w: x[sym.sigma_lambda_w],
                 sp.abc.alpha: x[sp.abc.alpha],
                 sp.abc.delta: x[sp.abc.delta],
                 sym.rho_w_theta_l: x[sym.rho_w_theta_l]
             }), axis=1))


# Calculate the moment analytically:

# In[18]:


df['num_int'] = (
    df.apply(lambda x: Rational(w_prime_2_theta_l_prime_bar.subs({
        sym.w_1: x[sym.w_1],
        sym.w_2: x[sym.w_2],
        sym.theta_l_1: x[sym.theta_l_1],
        sym.theta_l_2: x[sym.theta_l_2],
        sym.sigma_theta_l_1: x[sym.sigma_theta_l_1],
        sym.sigma_theta_l_2: x[sym.sigma_theta_l_2],
        sym.sigma_lambda_theta_l: x[sym.sigma_lambda_theta_l],
        sym.sigma_w: x[sym.sigma_w],
        sym.sigma_lambda_w: x[sym.sigma_lambda_w],
        sp.abc.alpha: x[sp.abc.alpha],
        sp.abc.delta: x[sp.abc.delta],
        sym.rho_w_theta_l: x[sym.rho_w_theta_l]
    }).doit(conds='none', method='quad').evalf()), axis=1))


# In[19]:


df['diff'] = abs(df['check_val'] - df['num_int'])


```

```
# In[20]:


df['diff_num'] = abs(df['check_val'].astype(float) - df['num_int'].astype(float))


# In[21]:


display(df)


# In[22]:


import numpy as np

print('The mean error between the rhs and the lhs is:', np.mean(df['diff_num']))

```

Listing B.4: w_prime_2_theta_l_prime_bar.py

B.6 w_prime_theta_l_prime_2_bar.py

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


from itertools import product

import pandas as pd

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import sympy as sp
from IPython.display import display
from sympy import abc, oo, Rational, init_printing

import checked_functions as c_f
import symbols as sym
init_printing()


# # This document aims to numerically check £\overline{w'\theta_l'^2}£

# ## Define the marginal distributions.
```

```python
# In[2]:


display(sp.Eq(sym.G_1_w_theta, c_f.G_1_w_theta_density))


# In[3]:


display(sp.Eq(sym.G_2_w_theta, c_f.G_2_w_theta_density))


# In[4]:


display(sp.Eq(sym.G_3_w_theta, c_f.G_3_w_theta_density))


# In[5]:


display(sp.Eq(sym.G_w_theta, c_f.G_w_theta, evaluate=False))


# In[6]:


w_prime_theta_l_2_prime_bar = sp.Integral(
    (sp.abc.w - sym.w_bar) * (sp.abc.theta - sym.theta_l_bar) ** 2 * c_f.G_w_theta,
    [sp.abc.w, -oo, oo],
    [sp.abc.theta, -oo, oo])

display(sp.Eq(sym.w_prime_theta_prime_l_2_bar, w_prime_theta_l_2_prime_bar))


# In[7]:


w_prime_theta_l_2_prime_bar = w_prime_theta_l_2_prime_bar.subs({
    sym.w_bar: c_f.w_bar(),
    sym.theta_l_bar: c_f.theta_l_bar()
})

display(sp.Eq(sym.w_prime_theta_prime_l_2_bar, w_prime_theta_l_2_prime_bar))


# The equation in the document is:

# In[8]:


display(sp.Eq(sym.w_prime_theta_prime_l_2_bar, c_f.w_prime_theta_l_prime_2_bar()))

```

```python
# For this we still need £\tilde{\sigma_w}£:

# In[9]:


display(sp.Eq(sym.sigma_tilde_w, c_f.sigma_tilde_w()))


# And £\overline{w'^{2}}£:

# In[10]:


display(sp.Eq(sym.w_prime_2_bar, c_f.w_prime_2_bar()))


# And £\overline{w'^{3}}£:

# In[11]:


display(sp.Eq(sym.w_prime_3_bar, c_f.w_prime_3_bar()))


# And £\overline{w'\theta'_l}£:

# In[12]:


display(sp.Eq(sym.w_prime_theta_l_prime_bar, c_f.w_prime_theta_l_prime_bar()))


# And £\lambda_w£:

# In[13]:


display(sp.Eq(sym.lambda_w, c_f.lambda_w()))


# And £\lambda_{w\theta}£:

# In[14]:


display(sp.Eq(sym.lambda_w_theta, c_f.lambda_w_theta()))


# And £\overline{\theta_l'^3}£:

# In[15]:


```

```
display(sp.Eq(sym.theta_l_prime_3_bar, c_f.theta_l_prime_3_bar()))


# Putting those all together yields:

# In[16]:


w_prime_theta_l_prime_2_bar_check_val = c_f.w_prime_theta_l_prime_2_bar().subs({
    sym.theta_l_prime_3_bar: c_f.theta_l_prime_3_bar(),
    sym.sigma_tilde_w: c_f.sigma_tilde_w(),
    sym.lambda_w: c_f.lambda_w(),
    sym.w_prime_2_bar: c_f.w_prime_2_bar(),
    sym.w_prime_3_bar: c_f.w_prime_3_bar(),
    sym.lambda_w_theta: c_f.lambda_w_theta(),
    sym.w_prime_theta_l_prime_bar: c_f.w_prime_theta_l_prime_bar()
})

w_prime_theta_l_prime_2_bar_check_val = w_prime_theta_l_prime_2_bar_check_val.subs({
    sym.w_prime_2_bar: c_f.w_prime_2_bar(),
    sym.lambda_w: c_f.lambda_w(),
    sym.w_bar: c_f.w_bar(),
    sym.theta_l_bar: c_f.theta_l_bar()
})

display(sp.Eq(sym.w_prime_theta_prime_l_2_bar, w_prime_theta_l_prime_2_bar_check_val))


# Since the integral is too difficult to be calculated analytically, at least with sympy, we try to put in some arbitrary numbers for the pdf parameters, to simplify the equations.

# We create a dataframe to get all possible permutations and therefore also all possible evaluations of the integrals.

# In[17]:


df = pd.DataFrame(
    product([0, 1],
            [-2, 2],
            [-1, 2],
            [0, 3],
            [Rational(1, 10)],
            [Rational(3, 10)],
            [Rational(4, 10)],
            [Rational(7, 10)],
            [Rational(6, 10)],
            [Rational(5, 10)],
            [Rational(1, 10), Rational(5, 10)],
            [Rational(5, 10)]),
    columns=[sym.w_1,
             sym.w_2,
             sym.theta_l_1,
```

```python
            sym.theta_l_2,
            sym.sigma_theta_l_1,
            sym.sigma_theta_l_2,
            sym.sigma_theta_l_3,
            sym.sigma_w,
            sym.sigma_w_3,
            sp.abc.alpha,
            sp.abc.delta,
            sym.rho_w_theta_l])


# In[18]:


df['check_val'] = (
    df.apply(lambda x:
            w_prime_theta_l_prime_2_bar_check_val.subs({
                sym.w_1: x[sym.w_1],
                sym.w_2: x[sym.w_2],
                sym.theta_l_1: x[sym.theta_l_1],
                sym.theta_l_2: x[sym.theta_l_2],
                sym.sigma_theta_l_1: x[sym.sigma_theta_l_1],
                sym.sigma_theta_l_2: x[sym.sigma_theta_l_2],
                sym.sigma_theta_l_3: x[sym.sigma_theta_l_3],
                sym.sigma_w: x[sym.sigma_w],
                sym.sigma_w_3: x[sym.sigma_w_3],
                sp.abc.alpha: x[sp.abc.alpha],
                sp.abc.delta: x[sp.abc.delta],
                sym.rho_w_theta_l: x[sym.rho_w_theta_l]
            }).evalf(), axis=1))


# Calculate the moment analytically:

# In[19]:


df['num_int'] = (
    df.apply(lambda x: w_prime_theta_l_2_prime_bar.subs({
        sym.w_1: x[sym.w_1],
        sym.w_2: x[sym.w_2],
        sym.theta_l_1: x[sym.theta_l_1],
        sym.theta_l_2: x[sym.theta_l_2],
        sym.sigma_theta_l_1: x[sym.sigma_theta_l_1],
        sym.sigma_theta_l_2: x[sym.sigma_theta_l_2],
        sym.sigma_theta_l_3: x[sym.sigma_theta_l_3],
        sym.sigma_w: x[sym.sigma_w],
        sym.sigma_w_3: x[sym.sigma_w_3],
        sp.abc.alpha: x[sp.abc.alpha],
        sp.abc.delta: x[sp.abc.delta],
        sym.rho_w_theta_l: x[sym.rho_w_theta_l]
    }).doit(conds='none', method='quad').evalf(), axis=1))
```

```python
# In[20]:


df['diff'] = abs(df['check_val'] - df['num_int'])


# In[21]:


df['diff_num'] = abs(df['check_val'].astype(float) - df['num_int'].astype(float))


# In[22]:


display(df)


# In[23]:


import numpy as np

print('The mean error between the rhs and the lhs is:', np.mean(df['diff_num']))

```

Listing B.5: w_prime_theta_l_prime_2_bar.py

B.7 w_prime_r_t_prime_theta_l_prime_bar.py

```python
#!/usr/bin/env python
# coding: utf-8

# In[2]:


from itertools import product

import pandas as pd

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import sympy as sp
from IPython.display import display
from sympy import abc, oo, Rational, init_printing

import checked_functions as c_f
import symbols as sym
```

```
init_printing()


# # This document aims to numerically check £\overline{w'r_t'\theta_l'}£

# ## Define the normal distributions.

# In[3]:


display(sp.Eq(sp.symbols('\\mu_1'), c_f.mu_1_w_theta_l_r_t, evaluate=False))
display(sp.Eq(sp.symbols('\\Sigma_1'), c_f.Sigma_1_w_theta_l_r_t, evaluate=False))
#display(sp.Eq(sym.G_1_w_theta_l_r_t, c_f.G_1_w_theta_l_r_t_density))


# In[4]:


display(sp.Eq(sp.symbols('\\mu_2'), c_f.mu_2_w_theta_l_r_t, evaluate=False))
display(sp.Eq(sp.symbols('\\Sigma_2'), c_f.Sigma_2_w_theta_l_r_t, evaluate=False))
#display(sp.Eq(sym.G_2_w_theta_l_r_t, c_f.G_2_w_theta_l_r_t_density))


# In[5]:


display(sp.Eq(sp.symbols('\\mu_3'), c_f.mu_3_w_theta_l_r_t, evaluate=False))
display(sp.Eq(sp.symbols('\\Sigma_3'), c_f.Sigma_3_w_theta_l_r_t, evaluate=False))
#display(sp.Eq(sym.G_3_w_theta_l_r_t, c_f.G_3_w_theta_l_r_t_density))


# In[6]:


display(sp.Eq(sym.G_w_theta_l_r_t, c_f.G_w_theta_l_r_t))


# In[7]:


w_prime_r_t_prime_theta_l_prime_bar = sp.Integral(
    (sp.abc.w - sym.w_bar) * (sym.r_t - sym.r_t_bar) * (sp.abc.theta - sym.theta_l_bar)
    ↪  * c_f.G_w_theta_l_r_t,
    [sp.abc.w, -oo, oo],
    [sym.theta_l, -oo, oo],
    [sym.r_t, -oo, oo])
display(sp.Eq(sym.w_prime_r_t_prime_theta_l_prime_bar,
↪  w_prime_r_t_prime_theta_l_prime_bar))


# In[8]:


```

```python
w_prime_r_t_prime_theta_l_prime_bar = (
    w_prime_r_t_prime_theta_l_prime_bar.subs({
        sym.w_bar: c_f.w_bar(),
        sym.r_t_bar: c_f.r_t_bar(),
        sym.theta_l_bar: c_f.theta_l_bar()
    }))
display(sp.Eq(sym.w_prime_r_t_prime_theta_l_prime_bar, w_prime_r_t_prime_theta_l_prime_bar))


# The equation in the document is:

# In[9]:


display(sp.Eq(sym.w_prime_r_t_prime_theta_l_prime_bar, c_f.w_prime_r_t_prime_theta_l_prime_bar()))


# Since the integral is too difficult to be calculated analytically, at least with sympy, we try to put in some arbitrary numbers for the pdf parameters, to simplify the equations.

# We also use the method `nquad(..)` from `sympy` to get a numerical evaluation of the 3d integral.

# We create a dataframe to get all possible permutations and therefore also all possible evaluations of the integrals.

# In[10]:


df = pd.DataFrame(
    product([3, 1],
            [-2],
            [-1, 3],
            [2],
            [1, 4],
            [2],
            [1.1],
            [1.3],
            [1.4],
            [1.7],
            [1.2],
            [1.5],
            [1.9],
            [1.6],
            [.55],
            [.8],
            [.65],
            [.45],
            [.35],
            [.5]),
    columns=[sym.w_1,
```

```python
            sym.w_2,
            sym.theta_l_1,
            sym.theta_l_2,
            sym.r_t_1,
            sym.r_t_2,
            sym.sigma_theta_l_1,
            sym.sigma_theta_l_2,
            sym.sigma_theta_l_3,
            sym.sigma_w,
            sym.sigma_r_t_1,
            sym.sigma_r_t_2,
            sym.sigma_r_t_3,
            sym.sigma_w_3,
            sp.abc.alpha,
            sp.abc.delta,
            sym.rho_w_theta_l,
            sym.rho_w_r_t,
            sym.rho_theta_l_r_t,
            sym.r_r_t_theta_l])


# In[11]:


w_prime_r_t_prime_theta_l_prime_bar_check_sym_val = (
    c_f.w_prime_r_t_prime_theta_l_prime_bar().subs({
        sym.w_bar: c_f.w_bar(),
        sym.r_t_bar: c_f.r_t_bar(),
        sym.theta_l_bar: c_f.theta_l_bar()
    }))


# In[12]:


df['check_val'] = (
    df.apply(lambda x: Rational(c_f.w_prime_r_t_prime_theta_l_prime_bar().subs({
        sym.w_bar: c_f.w_bar(),
        sym.theta_l_bar: c_f.theta_l_bar(),
        sym.r_t_bar: c_f.r_t_bar(),
        sym.w_1: x[sym.w_1],
        sym.w_2: x[sym.w_2],
        sym.theta_l_1: x[sym.theta_l_1],
        sym.theta_l_2: x[sym.theta_l_2],
        sym.r_t_1: x[sym.r_t_1],
        sym.r_t_2: x[sym.r_t_2],
        sym.sigma_theta_l_1: x[sym.sigma_theta_l_1],
        sym.sigma_theta_l_2: x[sym.sigma_theta_l_2],
        sym.sigma_r_t_1: x[sym.sigma_r_t_1],
        sym.sigma_r_t_2: x[sym.sigma_r_t_2],
        sp.abc.alpha: x[sp.abc.alpha],
        sp.abc.delta: x[sp.abc.delta],
        sym.r_r_t_theta_l: x[sym.r_r_t_theta_l]
    }).evalf()), axis=1))
```

```python
# Calculate the moment numerically:

# In[13]:


import scipy

df['num_int'] = df.apply(lambda x: scipy.integrate.nquad(
    sp.lambdify(
        [sp.abc.w, sym.r_t, sym.theta_l],
        (((sp.abc.w - c_f.w_bar()) *
          (sym.r_t - c_f.r_t_bar()) *
          (sym.theta_l - c_f.theta_l_bar()) *
          c_f.G_w_theta_l_r_t))
        .subs({
            sym.w_bar: c_f.w_bar(),
            sym.r_t_bar: c_f.r_t_bar(),
            sym.theta_l_bar: c_f.theta_l_bar(),
            sym.w_1: x[sym.w_1],
            sym.w_2: x[sym.w_2],
            sym.theta_l_1: x[sym.theta_l_1],
            sym.theta_l_2: x[sym.theta_l_2],
            sym.r_t_1: x[sym.r_t_1],
            sym.r_t_2: x[sym.r_t_2],
            sym.sigma_w: x[sym.sigma_w],
            sym.sigma_w_3: x[sym.sigma_w_3],
            sym.sigma_theta_l_1: x[sym.sigma_theta_l_1],
            sym.sigma_theta_l_2: x[sym.sigma_theta_l_2],
            sym.sigma_theta_l_3: x[sym.sigma_theta_l_3],
            sym.sigma_r_t_1: x[sym.sigma_r_t_1],
            sym.sigma_r_t_2: x[sym.sigma_r_t_2],
            sym.sigma_r_t_3: x[sym.sigma_r_t_3],
            sp.abc.alpha: x[sp.abc.alpha],
            sp.abc.delta: x[sp.abc.delta],
            sym.rho_w_theta_l: x[sym.rho_w_theta_l],
            sym.rho_w_r_t: x[sym.rho_w_r_t],
            sym.rho_theta_l_r_t: x[sym.rho_theta_l_r_t],
            sym.r_r_t_theta_l: x[sym.r_r_t_theta_l]
        })),
    ranges=[[-30, 30], [-30, 30], [-30, 30]])[0],
                         axis=1)


# In[14]:


df['diff'] = abs(df['check_val'] - df['num_int'])


# In[15]:


```

```python
df['diff_num'] = abs(df['check_val'].astype(float) - df['num_int'])


# In[16]:


display(df)


# In[17]:


import numpy as np

print('The mean error between the rhs and the lhs is:', np.mean(df['diff_num']))

```

Listing B.6: w_prime_r_t_prime_theta_l_prime_bar.py