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

Mai 2023

ABSTRACT

ADDING A THIRD GAUSSIAN 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 plays a crucial role in simulating atmospheric phenomena. It uses the sum of two probability density functions (pdfs) to represent a single grid layer. This 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 an innovative extension to the CLUBB model. We introduce a third normal pdf 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 maintaining computational tractability.

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. Additionally, it tries to describe certain asymptotic behavior of the trinormal pdf under various parameter settings.

Type dedication here.

Contents

List of Figures

List of Tables

List of Listings

List of Acronyms

cas computer algebra system. 13, 43

CLUBB Cloud Layers Unified By Binormals. ii, 1, 2, 4–6, 14, 27, 43, 44

lhs left hand side. 8

pde partial differential equation. 5

pdf probability density function. ii, 1, 3, 7, 8, 10, 13–17, 19, 20, 23–26, 28, 29, 40, 43

rhs right hand side. 6, 8, 9

List of Symbols

θ_l' Standardized liquid water potential temperature $(\theta_l' = \theta_l - \overline{\theta_l})$. 12

r_t' Standardized total water mixing ratio $(r_t' = r_t - \overline{r_t})$. 12

w' Standardized upward wind $(w' = w - \overline{w})$. 12, 19

Acknowledgments

Type acknowledgments here.

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). The motivation, details regarding the model's inputs and outputs, and a step-by-step approach for verifying the formulas are provided.

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 section details the introduction of the third normal pdf (section 3.1), the transformation of existing equations (section 4.3), and the derivation of key moments within the model (section 4.4 - section 4.5). Additionally, section 4.7 proposes a diagnostic approach to account for the skewness of heat and moisture, while section 4.8 introduces 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 analytic and numerical integration techniques, potentially 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

As being said in the chapter 1, we try to describe more possible shapes with adding a third pdf. To illustrate that, we plotted some of the shapes which are now possible but were not possible before. 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 illustrate 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 model this could look like figure 2.1.
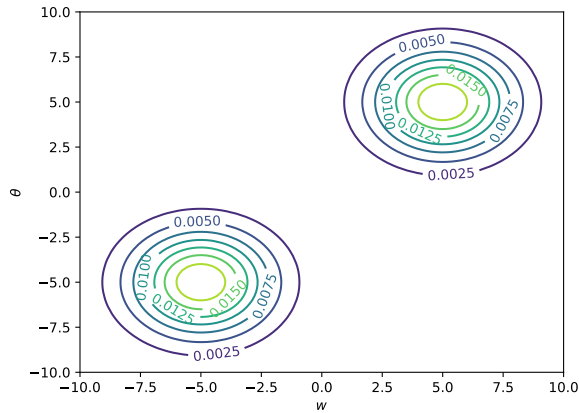
Figure 2.1: Binormal plot for two strong upward winds
With fixed variables: $w_1 = 5$, $w_2 = -5$, $\theta_{l1} = 5$, $\theta_{l2} = -5$, $\alpha = 0.5$, $\sigma_w = 2$, $\sigma_{\theta_{l1}} = 2$, $\sigma_{\theta_{l1}} = 2$.

However, this bimodal distribution (figure 2.1) doesn't accurately reflect reality. In nature, we would not expect such a sharp jump between the strong upward winds at w_1 and w_2. There would likely be some weaker upward winds 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.
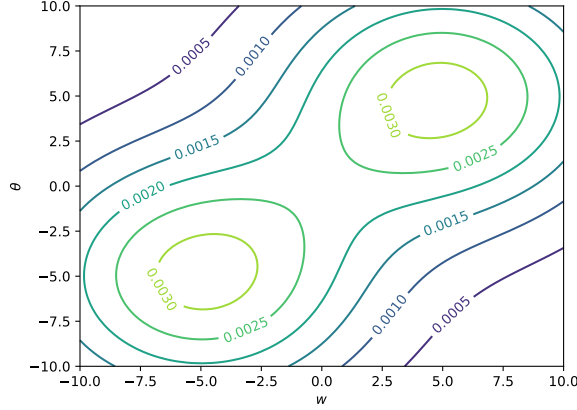
Figure 2.2: Binormal plot for two strong upward winds with increased standard deviations With fixed variables: $w_1 = 5$, $w_2 = -5$, $\theta_{l1} = 5$, $\theta_{l2} = -5$, $\alpha = 0.5$, $\sigma_w = 5$, $\sigma_{\theta_{l1}} = 5$, $\sigma_{\theta_{l1}} = 5$.

Seeing figure 2.2, the issue with having some values in the middle is slightly fixed 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, 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.
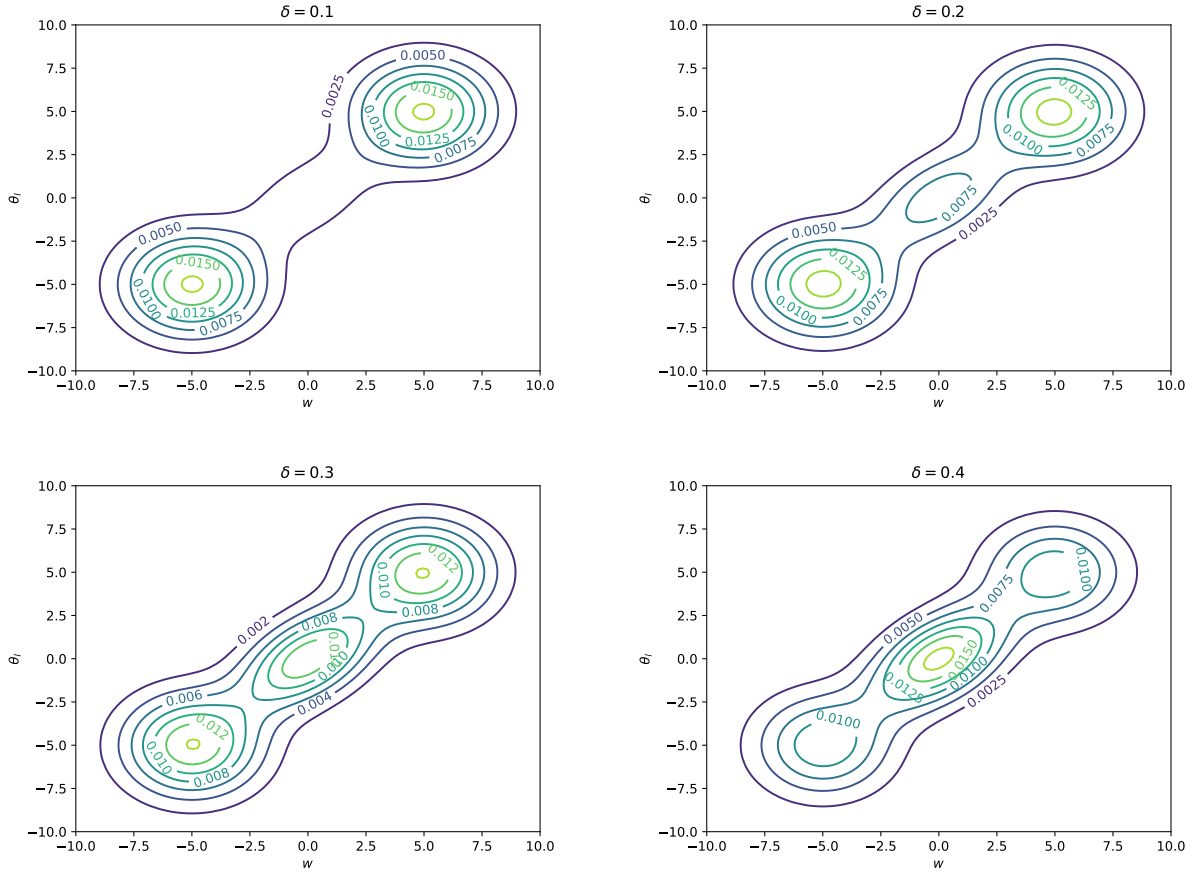
Figure 2.3: Trinormal plot for two strong upward winds with varying δ
With fixed variables: $w_1 = 5$, $w_2 = -5$, $\theta_{l1} = 5$, $\theta_{l2} = -5$, $\alpha = 0.5$, $\sigma_w = 2$, $\sigma_{\theta_{l1}} = 2$, $\sigma_{\theta_{l2}} = 2$, $\sigma_{w3} = 2$, $\sigma_{3\theta_l} = 2$, $\rho_{w\theta_l} = 0.5$.

2.2 Closing PDEs

The CLUBB model relies on a set of partial differential equations (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 ensures the model's mathematical integrity.

For instance, consider the following prognostic pde:[1]

---

[1]Vincent E. Larson. *CLUBB-SILHS: A parameterization of subgrid variability in the atmosphere.* 2022. arXiv: 1711.03675 [physics.ao-ph], p. 21.
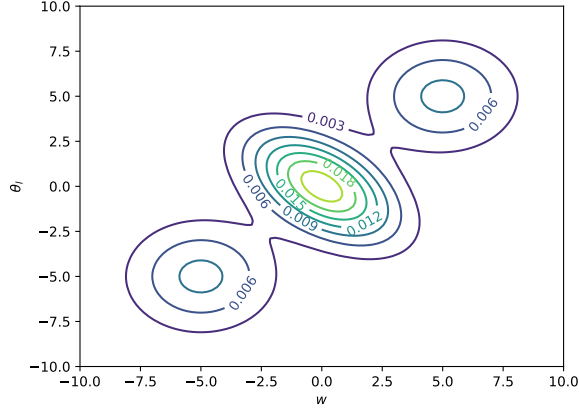
Figure 2.4: Trinormal plot for two strong upward winds with a third peak in the middle
With fixed variables: $w_1 = 5$, $w_2 = -5$, $\theta_{l1} = 5$, $\theta_{l2} = -5$, $\alpha = 0.5$, $\delta = 0.5$, $\sigma_w = 2$,
$\sigma_{\theta_{l1}} = 2$, $\sigma_{\theta_{l2}} = 2$, $\sigma_{w3} = 2$, $\sigma_{\theta_l 3} = 2$, $\rho_{w\theta_l} = 0.5$.

$$\frac{\partial \overline{w'r'_t}}{\partial t} = -\frac{\partial \overline{w'r'_t}}{\partial z} - \frac{1}{\rho_s}\frac{\partial \rho_s \overline{w'^2 r'_t}}{\partial z} - \overline{w'^2}\frac{\partial \overline{r'_t}}{\partial z} - \overline{w'r'_t}\frac{\partial \overline{w}}{\partial z} + \dots$$

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, even if they introduce slight limitations in capturing the full variability of the underlying atmospheric state.

Our focus here is on achieving closure for higher-order moments, such as the fourth moment of the vertical wind $(\overline{w'^4})$. This closure is achieved by expressing these higher-order moments in terms of readily calculable lower-order moments. This approach uses the relationships between moments within a probability distribution, enabling efficient model updates during

the time-stepping process.

2.3 Inputs and Outputs

While defining inputs and outputs can seem challenging at first glance, it's a crucial step towards understanding a system. In this context, the code provides us with a set of moment terms: $\overline{w}$, $\overline{w'^2}$, $\overline{w'^3}$, $\overline{\theta_l}$, $\overline{w'\theta_l'}$, $\overline{r_t}$, $\overline{w'r_t'}$, $\overline{\theta_l'^2}$, $\overline{r_t'^2}$, $\overline{r_t'\theta_l'}$. From these, we want to determine certain parameters which are describing the shape of the underlying pdf. Those parameters are standardized and some also normalized. So we try to solve for 13 unknowns, 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.

While the code seems to derive the desired moments from a set of parameters, we want to establish a formal mathematical foundation for these relationships. To achieve this, we will take a more traditional approach, working in the "forward" direction. This means we will:

1. *Define 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:

$$\overline{w} = \int_{\mathbb{R}} \int_{\mathbb{R}} \int_{\mathbb{R}} w \cdot P_{tmg} \, dw dr_t d\theta_l, \tag{2.3.1}$$

where $P_{tmg}{}^2$ is the pdf of the sum of all three normal distributions. Since some integrals are challenging to solve analytically, even with the help of SymPy, we are using the quadrature

---

[2]**T**rivariate **M**ixture of **G**aussians

7

method to calculate the integrals and choose arbitrary values for the inputs. All of this can be seen in section 5.2.

2.4 Steps for checking the formulas

We want to state a general approach for checking all those formulas and try to reference to the following steps in chapter 5 where there are actual examples stated. One way, or at least the way, which is used in this document is the following, where 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. Our formula for $\overline{w}$ (equation (4.4.1)) in terms of the pdf parameters can be checked.

3. Once the means are known, calculate the central variances, e.g. $\overline{w'^2} = \overline{(w - \overline{w})^2}$ by integration over the pdf. Our formula for $\overline{w'^2}$ (equation (4.4.3)) in terms of the pdf parameters can be checked.

4. Once the variances, e.g. $\overline{w'^2}$, are known, then *non-dimensional* pdf parameters such as λ_w 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, e.g. $\overline{w'^4}$ or $\overline{w'^2\theta'_l}$, by integration over the pdf. Again, we can check if lhs = rhs.

7. The integral $\overline{w'\theta_l'^2}$ is complicated. If it is written in terms of $\overline{\theta_l'^3}$ (equation (4.8.7)), then $\overline{\theta_l'^3}$ can be integrated (equation (4.4.9)) and substituted into the rhs of equation (4.8.7). If $\overline{w'\theta_l'^2}$ is written in terms of β (equation (4.8.9)), then we need to integrate to find $\overline{\theta_l'^3}$, back out β from equation (4.8.1), and finally substitute β into equation (4.8.7).

8. To verify equation (4.8.19) for $\overline{w'r_t'\theta_l'}$, use equation (4.4.21).

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)

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

3.2 Multivariate Normal distribution

We say that a random vector $\boldsymbol{X}$ (r × r) is distributed according to a multivariate normal when it has the following joint density function:[1]

Definition 2 (pdf of a multivariate normal distribution)

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{r}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})\right), \boldsymbol{x} \in \mathbb{R}^r, \tag{3.2.1}$$

---

[1] 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, p. 59.

where

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_r \end{pmatrix} \in \mathbb{R}^r \tag{3.2.2}$$

is the mean vector, and

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

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

3.2.1 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:

$$\mathbb{E}[X^3] = \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mu_3}{\sigma^3} = \frac{\mathbb{E}[(X-\mu)^3]}{(\mathbb{E}[(X-\mu)^2])^{3/2}} = \frac{\kappa_3}{\kappa_2^{3/2}}, \tag{3.2.3}$$

$$\mathbb{E}[X^4] = \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{\mathbb{E}[(X-\mu)^4]}{(\mathbb{E}[(X-\mu)^2])^2} = \frac{\mu_4}{\sigma^4}. \tag{3.2.4}$$

3.3 Thermodynamic Scalars

We denote the thermodynamic scalars 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.[2] 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.

---

[2]Larson, *CLUBB-SILHS: A parameterization of subgrid variability in the atmosphere*, p. 10.

4 Formulas

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 Third Normal

We would like to define a third normal to the already existing two trivariate normals, which is right in the middle between those two. Using the following formula for a multivariate normal, we can just define the sum of our three normals by using a vector for the mean, as well as the covariance matrix. For the purpose of readability, 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 restriction in the code) and the covariance matrices as

$$\Sigma_1 = \begin{pmatrix} \sigma_w^2 & 0 & 0 \\ 0 & \sigma_{\theta_{l1}}^2 & \rho_{\theta_l r_t}\sigma_{\theta_l 3}\sigma_{r_t 3} \\ 0 & \rho_{\theta_l r_t}\sigma_{\theta_l 3}\sigma_{r_t 3} & \sigma_{r_{t1}}^2 \end{pmatrix}, \text{ and } \Sigma_2 = \begin{pmatrix} \sigma_w^2 & 0 & 0 \\ 0 & \sigma_{\theta_{l2}}^2 & \rho_{\theta_l r_t}\sigma_{\theta_l 3}\sigma_{r_t 3} \\ 0 & \rho_{\theta_l r_t}\sigma_{\theta_l 3}\sigma_{r_t 3} & \sigma_{r_{t2}}^2. \end{pmatrix}$$

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 by therefore also taking away some variability. 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

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

For our proposed mix 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. The advantage over just two normal pdfs is that we can now express more shapes of functions. We also define some additional relationships for this third normal distribution.

$$\lambda_w = \frac{\sigma_{w3}^2}{w'^2}, \quad \lambda_\theta = \frac{\sigma_{\theta_l 3}^2}{\theta_l'^2}, \quad \lambda_r = \frac{\sigma_{r_t 3}^2}{r_t'^2}, \qquad (4.1.2)$$

$$\lambda_{\theta r} = \frac{\rho_{\theta_l r_t}\sigma_{\theta_l 3}\sigma_{r_t 3}}{\overline{r_t'\theta_l'}}, \quad \lambda_{w\theta} = \frac{\rho_{w\theta_l}\sigma_{w3}\sigma_{\theta_l 3}}{\overline{w'\theta_l'}}, \quad \lambda_{wr} = \frac{\rho_{wr_t}\sigma_{w3}\sigma_{r_t 3}}{\overline{w'r_t'}}. \qquad (4.1.3)$$

Hence, we can rewrite Σ_3 as

$$\Sigma_3 = \begin{pmatrix} \sigma_{w3}^2 & \overline{w'\theta_l'}\cdot\lambda_{wr} & \overline{w'r_t'}\cdot\lambda_{wr} \\ \overline{w'\theta_l'}\cdot\lambda_{wr} & \sigma_{\theta_l 3}^2 & \overline{r_t'\theta_l'}\cdot\lambda_{\theta r} \\ \overline{w'r_t'}\cdot\lambda_{wr} & \overline{r_t'\theta_l'}\cdot\lambda_{\theta r} & \sigma_{r_t 3}^2 \end{pmatrix}. \qquad (4.1.4)$$

4.2 Normalized Variables

Since CLUBB is mostly using "normalized variables", we are going to list those, which are given in standard form.

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

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

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

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

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

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

4.3 Transformation of the equations

Having established the definition of the sum of normal distributions in section 4.1 to enhance mathematical tractability, we now delve into the transformation of equations from the existing binormal representation used in CLUBB-SILHS[1] to a trinormal one. While the formulas for the binormal case are well-defined (see CLUBB-SILHS[2]), the simplicity of the means and standard deviations employed in the trinormal representation suggests the existence of a specific transformation that holds true. This chapter will demonstrate that the following transformations, denoted by the subscript "dGn" for the binormal case, successfully achieve this conversion.

$$\overline{w'^2} \frac{1-\delta\lambda_w}{1-\delta} = \overline{w'^2}_{dGn}, \tag{4.3.1}$$

$$\overline{w'^3} \frac{1}{1-\delta} = \overline{w'^3}_{dGn}, \tag{4.3.2}$$

---

[1]Larson, *CLUBB-SILHS: A parameterization of subgrid variability in the atmosphere*.
[2]Ibid.

$$\frac{\overline{w'^3}}{\overline{w'^2}^{3/2}}\frac{(1-\delta)^{1/2}}{(1-\lambda_w\delta)^{3/2}} = \frac{\overline{w'^3}_{dGn}}{\overline{w'^2}_{dGn}^{3/2}}, \tag{4.3.3}$$

$$\overline{\theta_l'^2}\frac{1-\delta\lambda_\theta}{1-\delta} = \overline{\theta_l'^2}_{dGn}, \tag{4.3.4}$$

$$\overline{w'\theta_l'}\frac{1-\delta\lambda_{w\theta}}{1-\delta} = \overline{w'\theta_l'}_{dGn}, \tag{4.3.5}$$

$$\left(\overline{w'^4} - 3\delta\lambda_w^2\left(\overline{w'^2}\right)^2\right)\frac{1}{1-\delta} = \overline{w'^4}_{dGn} \tag{4.3.6}$$

$$\left(\frac{\overline{w'^4}}{(\overline{w'^2})^2} - 3\delta\lambda_w^2\right)\frac{1-\delta}{(1-\lambda_w\delta)^2} = \frac{\overline{w'^4}_{dGn}}{(\overline{w'^2}_{dGn})^2} \tag{4.3.7}$$

To get a sense of what those transformations mean and why they should work, we pick e.g. equation (4.3.1). If we substitute in the already defined formula for λ_w (equation (4.1.2)), we get

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

$$\overline{w'^2}(1-\delta\frac{\sigma_{w3}^2}{\overline{w'^2}}) = (1-\delta)\overline{w'^2}_{dGn}$$

$$\overline{w'^2} - \delta\sigma_{w3}^2 = (1-\delta)\overline{w'^2}_{dGn}$$

$$\overline{w'^2} = \overline{w'^2}_{dGn} - \delta\overline{w'^2}_{dGn} + \delta\sigma_{w3}^2$$

$$\overline{w'^2} = \overline{w'^2}_{dGn} - \delta\left(\overline{w'^2}_{dGn} - \sigma_{w3}^2\right)$$

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.

16

Also, if we look at equation (4.3.2), we see that there is no more λ_w present.

$$\overline{w'^3} = (1 - \delta)\overline{w'^3}_{dGn}$$

It makes sense graphically, that as delta 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 4.1. Table 4.1 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 (4.3.1).

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 evident 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 green plot (mixture) as δ approaches 1.
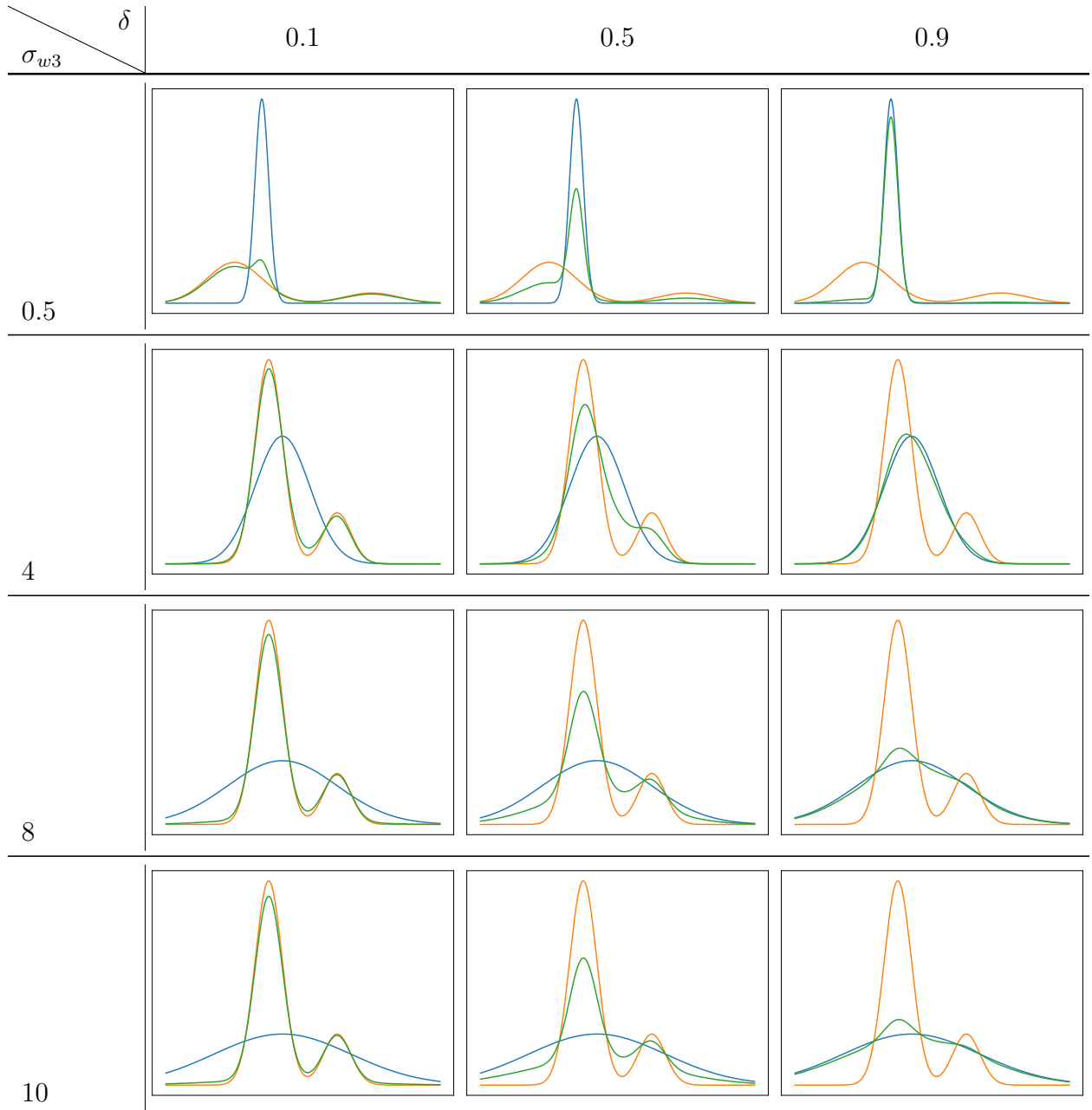
Table 4.1: 1D Plots for different δ and σ_{w3}

With fixed variables: $w_1 = 5$, $w_2 = -5$, $\alpha = 0.2$, $\sigma_w = 2$.

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.

18

4.4 Deduced Moments

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.

4.4.1 Lower Order Moments

4.4.1.1 Moments for w

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

$$\overline{w} = (1 - \delta)\alpha w_1 + (1 - \delta)(1 - \alpha)w_2 + \delta w_3, \tag{4.4.1}$$

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:

$$0 = \alpha \widehat{w}_1 + (1 - \alpha)\widehat{w}_2 \tag{4.4.2}$$

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:

$$\overline{w'^2} = (1 - \delta)\alpha[(w_1 - \overline{w})^2 + \sigma_w^2]$$
$$+ (1 - \delta)(1 - \alpha)[(w_2 - \overline{w})^2 + \sigma_w^2]$$
$$+ \delta\sigma_{w3}^2, \tag{4.4.3}$$

19

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

$$\overline{w'^2} = \mathbb{E}[w'^2] = \mathbb{E}[(w - \overline{w})^2] = \mathbb{E}[(w - \mathbb{E}[w])^2] = \mathbb{E}[w^2 - 2w\mathbb{E}[w] + \mathbb{E}[w]^2]$$

$$= \mathbb{E}[w^2] - 2\mathbb{E}[w\mathbb{E}[w]] + \mathbb{E}[\mathbb{E}[w]^2] = \mathbb{E}[w^2] - 2\mathbb{E}[w]\mathbb{E}[w] + \mathbb{E}[w]^2$$

$$= \mathbb{E}[w^2] - \mathbb{E}[w]^2 = \mathrm{Var}[w].$$

The *non-dimensional* relationship would then be:

$$\frac{1}{(1 - \tilde{\sigma}_w^2)} = \alpha \left(\widehat{w}_1^2 + \frac{\tilde{\sigma}_w^2}{(1 - \tilde{\sigma}_w)^2} \right) + (1 - \alpha) \left(\widehat{w}_2^2 + \frac{\tilde{\sigma}_w^2}{(1 - \tilde{\sigma}_w^2)} \right). \tag{4.4.4}$$

The third order moment is given as:

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

$$+ (1 - \delta)(1 - \alpha)[(w_2 - \overline{w})^3 + 3\sigma_w^2(w_2 - \overline{w})] \tag{4.4.5}$$

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

$$\widehat{Sk}_w \equiv \frac{1}{(1 - \tilde{\sigma}_w^2)^{3/2}} \frac{\overline{w'^3}}{\left(\overline{w'^2} \right)^{3/2}} \frac{1}{\left(\frac{1 - \delta\lambda_w}{1 - \delta} \right)^{3/2}} \frac{1}{1 - \delta}$$

$$= \alpha \left(\widehat{w}_1^3 + 3\widehat{w}_1 \frac{\tilde{\sigma}_w^2}{(1 - \tilde{\sigma}_w^2)} \right) + (1 - \alpha) \left(\widehat{w}_2^3 + 3\widehat{w}_2 \frac{\tilde{\sigma}_w^2}{(1 - \tilde{\sigma}_w^2)} \right) \tag{4.4.6}$$

4.4.1.2 Moments for θ_l

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

$$0 = \alpha \tilde{\theta}_{l1} + (1 - \alpha) \tilde{\theta}_{l2} \tag{4.4.7}$$

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

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

$$+ (1-\delta)(1-\alpha)[(\theta_{l2} - \overline{\theta_l})^2 + \sigma_{\theta_{l2}}^2]$$

$$+ \delta\sigma_{\theta_l 3}^2, \tag{4.4.8}$$

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:

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

$$+ (1-\delta)(1-\alpha)[(\theta_{l2} - \overline{\theta_l})^3 + 3\sigma_{\theta_{l2}}^2(\theta_{l2} - \overline{\theta_l})] \tag{4.4.9}$$

Similarly to equation (4.4.6), we also list a moment which is more diagnosed than prognosed:

$$\widehat{Sk_{\theta_l}} \equiv \frac{\overline{\theta_l'^3}}{\left(\overline{\theta_l'^2}\right)^{3/2}} \left(\frac{1}{\frac{1-\delta\lambda_\theta}{1-\delta}}\right)^{3/2} \frac{1}{1-\delta}$$

$$= \alpha\left(\tilde{\theta}_{l1}^3 + 3\tilde{\theta}_{l1}\tilde{\sigma}_{\theta_l 1}^2\right) + (1-\alpha)\left(\tilde{\theta}_{l2}^3 + 3\tilde{\theta}_{l2}\tilde{\sigma}_{\theta_l 2}^2\right). \tag{4.4.10}$$

4.4.1.3 Moments for r_t

The relationships for r_t and $r_t'^2$ are given as follows

$$0 = \alpha\tilde{r}_{t1} + (1-\alpha)\tilde{r}_{t2}, \tag{4.4.11}$$

and

$$1 = \alpha\left(\tilde{r}_{t1}^2 + \tilde{\sigma}_{r_{t1}}^2\right) + (1-\alpha)\left(\tilde{r}_{t2}^2 + \tilde{\sigma}_{r_{t2}}^2\right). \tag{4.4.12}$$

21

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

$$\overline{r_t'^2} = (1-\delta)\alpha[(r_{t1} - \overline{r_t})^2 + \sigma_{r_{t1}}^2]$$
$$+ (1-\delta)(1-\alpha)[(r_{t2} - \overline{r_t})^2 + \sigma_{\theta_{l2}}^2]$$
$$+ \delta\sigma_{r_t3}^2, \tag{4.4.13}$$

and

$$\overline{r_t'^3} = (1-\delta)\alpha[(r_{t1} - \overline{r_t})^3 + 3\sigma_{r_{t1}}^2(r_{t1} - \overline{r_t})]$$
$$+ (1-\delta)(1-\alpha)[(r_{t2} - \overline{r_t})^3 + 3\sigma_{r_{t2}}^2(r_{t2} - \overline{r_t})] \tag{4.4.14}$$

4.4.1.4 Mixed Moments

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

$$\overline{w'\theta_l'} = (1-\delta)\alpha[(w_1 - \overline{w})(\theta_{l1} - \overline{\theta_l})]$$
$$+ (1-\delta)(1-\alpha)[(w_2 - \overline{w})(\theta_{l2} - \overline{\theta_l})]$$
$$+ \delta\lambda_{w\theta}\overline{w'\theta_l'}, \tag{4.4.15}$$

$$\overline{w'r_t'} = (1-\delta)\alpha[(w_1 - \overline{w})(r_{t1} - \overline{r_t})]$$
$$+ (1-\delta)(1-\alpha)[(w_2 - \overline{w})(r_{t2} - \overline{r_t})]$$
$$+ \delta\lambda_{wr}\overline{w'r_t'}, \tag{4.4.16}$$

and

$$\overline{r_t'\theta_l'} = (1-\delta)\alpha[(r_{t1} - \overline{r_t})(\theta_{l1} - \overline{\theta_l}) + r_{r_t\theta_l}\sigma_{r_{t1}}\sigma_{\theta_{l1}}]$$
$$+ (1-\delta)(1-\alpha)[(r_{t2} - \overline{r_t})(\theta_{l2} - \overline{\theta_l}) + r_{r_t\theta_l}\sigma_{r_{t2}}\sigma_{\theta_{l2}}]$$
$$+ \delta\lambda_{r\theta}\overline{r_t'\theta_l'}. \tag{4.4.17}$$

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

$$\widehat{c}_{w\theta_l} \equiv \frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}} \frac{\overline{w'\theta_l'}}{\sqrt{\overline{w'^2}}\sqrt{\overline{\theta_l'^2}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_w}{1-\delta}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_\theta}{1-\delta}}} \frac{1-\delta\lambda_{w\theta}}{1-\delta}$$

$$= \alpha\widehat{w}_1\tilde{\theta}_{l1} + (1-\alpha)\widehat{w}_2\tilde{\theta}_{l2}, \tag{4.4.18}$$

$$\widehat{c}_{wr_t} \equiv \frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}} \frac{\overline{w'r_t'}}{\sqrt{\overline{w'^2}}\sqrt{\overline{r_t'^2}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_w}{1-\delta}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_r}{1-\delta}}} \frac{1-\delta\lambda_{wr}}{1-\delta}$$

$$= \alpha\widehat{w}_1\tilde{r}_{t1} + (1-\alpha)\widehat{w}_2\tilde{r}_{t2}, \tag{4.4.19}$$

and

$$\widehat{c}_{r_t\theta_l} \equiv \frac{\overline{r_t'\theta_l'}}{\sqrt{\overline{r_t'^2}}\sqrt{\overline{\theta_l'^2}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_q}{1-\delta}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_\theta}{1-\delta}}} \frac{1-\delta\lambda_{\theta r}}{1-\delta}$$

$$= \alpha\left(\tilde{r}_{t1}\tilde{\theta}_{l1} + r_{r_t\theta_l}\tilde{\sigma}_{q_{t1}}\tilde{\sigma}_{\theta_{l1}}\right) + (1-\alpha)\left(\tilde{r}_{t2}\tilde{\theta}_{l2} + r_{r_t\theta_l}\tilde{\sigma}_{r_{t2}}\tilde{\sigma}_{\theta_{l2}}\right), \tag{4.4.20}$$

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

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

$$\overline{w'r_t'\theta_l'} = (1-\delta)\alpha(w_1 - \overline{w})[(r_{t1} - \overline{r_t})(\theta_{l1} - \overline{\theta_l}) + r_{r_t\theta_l}\sigma_{r_{t1}}\sigma_{\theta_{l1}}]$$

$$+ (1-\delta)(1-\alpha)(w_2 - \overline{w})[(r_{t2} - \overline{r_t})(\theta_{l2} - \overline{\theta_l}) + r_{r_t\theta_l}\sigma_{r_{t2}}\sigma_{\theta_{l2}}] \tag{4.4.21}$$

4.5 Finding pdf parameters in terms of moments

We now select a particular member of the normal mixture family by mapping the prognosed moments to the pdf parameters. In other words, we invert equation (4.4.2) - equation (4.4.20) in order to find the set of pdf parameters that guarantees that the resulting pdf has moments that correspond to the prognosed ones. The inversion is non-trivial because the equations are non-linear in the pdf parameters. However, the pdf (equation (4.1.1)) is simple enough to permit an analytic solution.

The solution procedure is as follows.

1. Solve for the pdf parameters α, $\widehat{w}_1$, and $\widehat{w}_2$ from the moment equations for $\overline{w}$ (equation (4.4.2)), $\overline{w'^2}$ (equation (4.4.4)), $\overline{w'^3}$ (equation (4.4.6)):

$$\alpha = \frac{1}{2}\left[1 - \widehat{Sk}_w \sqrt{\frac{1}{4 + \widehat{Sk}_w^2}}\right], \tag{4.5.1}$$

$$\widehat{w}_1 = \sqrt{\frac{1-\alpha}{\alpha}}, \tag{4.5.2}$$

$$\widehat{w}_2 = -\sqrt{\frac{\alpha}{1-\alpha}}. \tag{4.5.3}$$

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

2. Equation (4.5.1) implies that $\widehat{Sk}_w$ is determined solely by α:

$$\widehat{Sk}_w = \frac{1 - 2\alpha}{\sqrt{\alpha(1-\alpha)}}. \tag{4.5.4}$$

3. We can obtain $\tilde{\theta}_{l1}$ and $\tilde{\theta}_{l2}$ from equation (4.4.7) for $\overline{\theta_l}$, and equation (4.4.18) for $\overline{w'\theta_l'}$:

$$\tilde{\theta}_{l1} = -\frac{\widehat{c}_{w\theta_l}}{\widehat{w}_2}, \tag{4.5.5}$$

$$\tilde{\theta}_{l2} = -\frac{\widehat{c}_{w\theta_l}}{\widehat{w}_1}. \tag{4.5.6}$$

4. The widths of the Normals, $\tilde{\sigma}_{\theta_l 1}$ and $\tilde{\sigma}_{\theta_l 2}$, are determined by satisfying equation (4.4.8) for $\overline{\theta_l'^2}$, and equation (4.4.9) for $\overline{\theta_l'^3}$:

$$\tilde{\sigma}_{\theta_l 1}^2 = \left(1 - \widehat{c}_{w\theta_l}^2\right) + \left(\sqrt{\frac{1-\alpha}{\alpha}}\right)\frac{1}{3\widehat{c}_{w\theta_l}}\left(\widehat{Sk}_{\theta_l} - \widehat{c}_{w\theta_l}^3 \widehat{Sk}_w\right), \tag{4.5.7}$$

$$\tilde{\sigma}_{\theta_l 2}^2 = \left(1 - \widehat{c}_{w\theta_l}^2\right) - \left(\sqrt{\frac{\alpha}{1-\alpha}}\right)\frac{1}{3\widehat{c}_{w\theta_l}}\left(\widehat{Sk}_{\theta_l} - \widehat{c}_{w\theta_l}^3 \widehat{Sk}_w\right). \tag{4.5.8}$$

Here Sk_{θ_l} is the skewness of θ_l. It must be provided either by a prognostic equation or by a diagnostic equation such as equation (4.4.10) below.

5. Equations for $\tilde{r}_{t1}$, $\tilde{r}_{t2}$, $\tilde{\sigma}^2_{r_t1}$, and $\tilde{\sigma}^2_{r_t2}$ are found by expressions identical to equation (4.5.5), equation (4.5.6), equation (4.5.7), and equation (4.5.8), except that θ_l is replaced everywhere by r_t.

6. Finally, from equation (4.4.17) for $\overline{r'_t \theta'_l}$ we find

$$r_{r_t \theta_l} = \frac{\widehat{c}_{r_t \theta_l} - \widehat{c}_{wr_t}\widehat{c}_{w\theta_l}}{\alpha \tilde{\sigma}_{r_t1}\tilde{\sigma}_{\theta_l1} + (1-\alpha)\tilde{\sigma}_{r_t2}\tilde{\sigma}_{\theta_l2}}. \tag{4.5.9}$$

Here $r_{r_t \theta_l}$ is the in-between normal correlation and $c_{r_t \theta_l}$ is the total correlation.

4.6 Higher-order moments in terms of pdf parameters

Once the pdf parameters have been specified, all higher-order moments can be calculated by integration over the pdf. The needed formulas are:

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

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

$$+ \frac{1}{(1-\tilde{\sigma}_w^2)^2}\frac{(1-\delta)}{(1-\delta\lambda_w)^2}\delta 3\lambda_w^2, \tag{4.6.1}$$

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

$$+ (1-\alpha)\left[\widehat{w}_2^2 + \frac{\tilde{\sigma}_w^2}{(1-\tilde{\sigma}_w^2)}\right]\tilde{\theta}_{l2}, \tag{4.6.2}$$

$$\frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}}\frac{(1-\delta)^{1/2}}{(1-\delta\lambda_w)^{1/2}(1-\delta\lambda_\theta)}\frac{\overline{w'\theta'^2_l}}{\left(\overline{w'^2}\right)^{1/2}\overline{\theta'^2_l}} = \alpha\widehat{w}_1\left(\tilde{\theta}_{l1}^2 + \tilde{\sigma}_{\theta_l1}^2\right) + (1-\alpha)\widehat{w}_2\left(\tilde{\theta}_{l2}^2 + \tilde{\sigma}_{\theta_l2}^2\right),$$

$$\tag{4.6.3}$$

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

$$= \alpha\widehat{w}_1 \left(\tilde{r}_{t1}\tilde{\theta}_{l1} + r_{r_t\theta_l}\tilde{\sigma}_{r_{t1}}\tilde{\sigma}_{\theta_{l1}} \right) + (1-\alpha)\widehat{w}_2 \left(\tilde{r}_{t2}\tilde{\theta}_{l2} + r_{r_t\theta_l}\tilde{\sigma}_{r_{t2}}\tilde{\sigma}_{\theta_{l2}} \right) \tag{4.6.4}$$

The equations for $\overline{w'^2 r_t'}$ and $\overline{w'r_t'^2}$ are analogous to equation (4.6.3) and equation (4.6.4).

4.7 A diagnostic ansatz for the skewness of heat and moisture

We cannot close the system of equations until we specify the skewness of θ_l, Sk_{θ_l}, that appears in equation (4.5.7) for $\tilde{\sigma}_{\theta_l 1}$, and equation (4.5.8) for $\tilde{\sigma}_{\theta_l 2}$. Likewise, we need to specify Sk_{r_t}. We could prognose these scalar skewnesses, but this would involve additional computational expense, storage, and complexity. In some cases, the extra complexity may be worthwhile. However, here we instead propose the following diagnostic formula

$$\widehat{Sk}_{\theta_l} = \widehat{Sk}_w \widehat{c}_{w\theta_l} \left[\beta + (1-\beta)\widehat{c}_{w\theta_l}^2 \right], \tag{4.7.1}$$

and a similar formula for $\widehat{Sk}_{r_t}$. The parameter β is dimensionless. We also solve for β because we are going to need the equation later on to show that other equations are true.

$$\implies \beta = \frac{\frac{\widehat{Sk}_{\theta_l}}{\widehat{Sk}_w \widehat{c}_{w\theta_l}} - \widehat{c}_{w\theta_l}^2}{1 - \widehat{c}_{w\theta_l}^2} \tag{4.7.2}$$

Equation (4.7.1) is physically plausible but limited at the same time. The formula states that Sk_{θ_l} is proportional to Sk_w, the skewness of w. An increase in β leads to an increase in $|Sk_{\theta_l}|$, which, in turn, leads to a pdf with a longer θ_l-tail. Sk_{θ_l} and Sk_w have the same sign when w and θ_l are positively correlated; Sk_{θ_l} and Sk_w have opposite sign when w and θ_l are negatively correlated. In those large eddy simulations, this is usually but not always

true. Sk_{θ_l} vanishes when either Sk_w or $c_{w\theta_l}$ vanishes; clearly this need not be true in nature. $|Sk_{\theta_l}|$ can be either smaller or larger than $|Sk_w|$, depending on the values of $\tilde{\sigma}_w^2$, $c_{w\theta_l}$, and β.

If we assume our ansatz (equation (4.4.10)) for $\widehat{Sk}_{\theta_l}$, then the θ_l-widths of the first (equation (4.5.7)) and second normal (equation (4.5.7)) reduce to

$$\tilde{\sigma}_{\theta_l 1}^2 = \frac{\left(1 - \widehat{c}_{w\theta_l}^2\right)}{\alpha} \left[\frac{1}{3}\beta + \alpha \left(1 - \frac{2}{3}\beta\right)\right], \tag{4.7.3}$$

and

$$\tilde{\sigma}_{\theta_l 2}^2 = \frac{\left(1 - \widehat{c}_{w\theta_l}^2\right)}{1 - \alpha} \left\{1 - \left[\frac{1}{3}\beta + \alpha \left(1 - \frac{2}{3}\beta\right)\right]\right\}. \tag{4.7.4}$$

Substituting equation (4.7.3), equation (4.7.4), and their r_t counterparts into the expression for $r_{r_t \theta_l}$ equation (4.5.9) yields the simplified form:

$$r_{r_t \theta_l} = \frac{c_{r_t \theta_l} - \widehat{c}_{wr_t} \widehat{c}_{w\theta_l}}{\left(1 - \widehat{c}_{wr_t}^2\right)^{1/2} \left(1 - \widehat{c}_{w\theta_l}^2\right)^{1/2}}. \tag{4.7.5}$$

Here, $r_{r_t \theta_l}$ is the correlation of r_t and θ_l in-between the Normals and $c_{r_t \theta_l}$ is the total correlation.

CLUBB chose a specific formula for the w-"width" of the individual normals, which is the following:

$$\tilde{\sigma}_w^2 = \gamma \left[1 - \max(c_{w\theta_l}^2, c_{wr_t}^2)\right]. \tag{4.7.6}$$

This makes $\tilde{\sigma}_w^2$ depend on $c_{wr_t}^2$ and $c_{w\theta_l}^2$. Here $0 \leq \gamma < 1$ is a dimensionless constant. This formula helps ensure that when c_{wr_t} or $c_{w\theta_l}$ becomes large in magnitude, $0 \leq \widehat{c}_{w\theta_l}^2, \widehat{c}_{wr_t}^2 < 1$ and hence $\tilde{\sigma}_{r_t 1,2}^2$, $\tilde{\sigma}_{\theta_l 1,2}^2$, and $r_{r_t \theta_l}$ remain realistic.

4.8 Proposed closures for third- and fourth-order moments based on the mixture of trivariate normals

This section lists formulas for the higher-order moments that are needed for closure in the parameterization of: $\overline{w'^4}$, $\overline{w'^2\theta'_l}$, $\overline{w'\theta'^2_l}$, and $\overline{w'r'_t\theta'_l}$. These are obtained by substituting the expressions for the pdf parameters (equation (4.5.1) - equation (4.5.9)) into the equations for the higher-order moments (equation (4.6.1) - equation (4.6.4)). Formulas for $\overline{w'^2r'_t}$ and $\overline{w'r'^2_t}$, which are also needed, are identical to those for $\overline{w'^2\theta'_l}$ and $\overline{w'\theta'^2_l}$ except that r_t replaces θ_l everywhere.

First, we list the equation for θ'^3_l, which is obtained by dimensionalizing equation (4.7.1):

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

(4.8.1)

While the scalar third moments are not essential for solving the prognostic equations directly, they still play a role in shaping cloud properties. This is because cumulus clouds tend to form at the edges, or "tails", of the pdf for a specific variable. Typically, as the relative "width" of the normal distribution in w (represented by $\tilde{\sigma}^2_w = \sigma^2_w/\overline{w'^2}$) increases, the value of $\overline{\theta'^3_l}$ also grows (see equation (4.8.1)) for details). In simpler terms, a larger $\overline{\theta'^3_l}$ corresponds to a broader w-marginal pdf, which deviates further from a double delta function (a function with two spikes at zero).

$\overline{w'^4}$ does not depend on the thermodynamic scalar moments; therefore, it does not depend on β. Substituting equation (4.5.2) and equation (4.5.2) into equation (4.6.1) and using

equation (4.4.6), we find

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

$$+ \widehat{Sk}_w^2 + \frac{1}{(1-\tilde{\sigma}_w^2)^2}\frac{(1-\delta)}{(1-\delta\lambda_w)^2}\delta 3\lambda_w^2, \tag{4.8.2}$$

and also

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

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

$$+ \delta 3\lambda_w^2\left(\overline{w'^2}\right)^2. \tag{4.8.3}$$

Similar to $\overline{w'^4}$, $\overline{w'^2\theta_l'}$ does not depend on β for our particular pdf family. Substituting equation (4.5.2) - equation (4.5.6) into equation (4.6.2), we find

$$\frac{1}{(1-\tilde{\sigma}_w^2)}\frac{(1-\delta)^{1/2}}{(1-\delta\lambda_w)(1-\delta\lambda_\theta)^{1/2}}\frac{\overline{w'^2\theta_l'}}{\overline{w'^2}\left(\overline{\theta_l'^2}\right)^{1/2}} = \widehat{c}_{w\theta_l}\widehat{Sk}_w, \tag{4.8.4}$$

and

$$\overline{w'^2\theta_l'} = \frac{1}{(1-\tilde{\sigma}_w^2)}\frac{1-\delta\lambda_{w\theta}}{1-\delta\lambda_w}\frac{\overline{w'^3}}{\overline{w'^2}}\overline{w'\theta_l'}. \tag{4.8.5}$$

$\overline{w'\theta_l'^2}$ depends explicitly on Sk_{θ_l}. Substituting equation (4.5.2) - equation (4.5.8) into equation (4.6.4) yields

$$\frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}}\frac{(1-\delta)^{1/2}}{(1-\delta\lambda_w)^{1/2}(1-\delta\lambda_\theta)}\frac{\overline{w'\theta_l'^2}}{\left(\overline{w'^2}\right)^{1/2}\overline{\theta_l'^2}} = \frac{2}{3}\widehat{c}_{w\theta_l}^2\widehat{Sk}_w + \frac{1}{3}\frac{\widehat{Sk}_{\theta_l}}{\widehat{c}_{w\theta_l}}, \tag{4.8.6}$$

and

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

$$+ \frac{1}{3}\frac{(1-\delta\lambda_w)}{(1-\delta\lambda_{w\theta})}\left(1-\tilde{\sigma}_w^2\right)\frac{\overline{w'^2}}{\overline{w'\theta_l'}}\frac{\overline{\theta_l'^3}}{\overline{w'\theta_l'}}. \tag{4.8.7}$$

The formula has a problem because $\overline{w'\theta_l'}$ is in the denominator. As $\overline{w'\theta_l'}$ gets closer to zero, the formula becomes infinitely large, which is called a singularity. This can cause issues if we use the formula directly with real-world measurements of $\overline{\theta_l'^3}$ and $\overline{w'\theta_l'}$. The resulting diagnosis of $\overline{w'\theta_l'^2}$ would be very sensitive to small changes in the measurements and might not be reliable (noisy). We can fix this singularity by either

- substitute in the ansatz for Sk_{θ_l} (equation (4.4.10)) into the original formula (equation (4.6.3)),

- or, equivalently, substitute equation (4.8.1) for $\overline{\theta_l'^3}$. This is possible because equation (4.8.1) shows that $\overline{\theta_l'^3}$ is proportional to $\overline{w'\theta_l'}$.

Both approaches effectively remove the singularity from the formula. Therefore, we find:

$$\frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}}\frac{(1-\delta)^{1/2}}{(1-\delta\lambda_w)^{1/2}(1-\delta\lambda_\theta)}\frac{\overline{w'\theta_l'^2}}{\left(\overline{w'^2}\right)^{1/2}\overline{\theta_l'^2}} = \widehat{Sk}_w\left[\frac{1}{3}\beta + \left(1-\frac{1}{3}\beta\right)\widehat{c}_{w\theta_l}^2\right], \qquad (4.8.8)$$

and

$$\overline{w'\theta_l'^2} = \frac{1}{(1-\tilde{\sigma}_w^2)}\frac{(1-\delta\lambda_\theta)}{(1-\delta\lambda_w)}\frac{\overline{w'^3}}{\overline{w'^2}}\left[\frac{1}{3}\beta\overline{\theta_l'^2} + \frac{\left(1-\frac{1}{3}\beta\right)}{(1-\tilde{\sigma}_w^2)}\frac{(1-\delta\lambda_{w\theta})^2}{(1-\delta\lambda_w)(1-\delta\lambda_\theta)}\frac{\left(\overline{w'\theta_l'}\right)^2}{\overline{w'^2}}\right]. \qquad (4.8.9)$$

Finally, substituting equation (4.5.2) - equation (4.5.9) into equation (4.6.4) yields the following formula for the turbulent flux of $\overline{r_t'\theta_l'}$, $\overline{w'r_t'\theta_l'}$:

$$\frac{(1-\delta)^{1/2}}{(1-\delta\lambda_w)^{1/2}(1-\delta\lambda_\theta)^{1/2}(1-\delta\lambda_{r_t})^{1/2}}\frac{\overline{w'r_t'\theta_l'}}{(1-\tilde{\sigma}_w^2)^{1/2}\left(\overline{w'^2}\right)^{1/2}\left(\overline{r_t'^2}\right)^{1/2}\left(\overline{\theta_l'^2}\right)^{1/2}}$$
$$= \widehat{c}_{wr_t}\widehat{c}_{w\theta_l}\widehat{Sk}_w + E(w,q_t,\theta_l)\frac{1}{2}\widehat{Sk}_w\left(c_{q_t\theta_l} - \widehat{c}_{wr_t}\widehat{c}_{w\theta_l}\right), \qquad (4.8.10)$$

and

$$\overline{w'r_t'\theta_l'} = \frac{\frac{1}{2}E}{(1-\tilde{\sigma}_w^2)}\frac{(1-\delta\lambda_{\theta q})}{(1-\delta\lambda_w)}\overline{r_t'\theta_l'}\frac{\overline{w'^3}}{\overline{w'^2}}$$
$$+ \frac{1-\frac{1}{2}E}{(1-\tilde{\sigma}_w^2)^2}\frac{(1-\delta\lambda_{wq})(1-\delta\lambda_{w\theta})}{(1-\delta\lambda_w)^2}\overline{w'r_t'}\;\overline{w'\theta_l'}\frac{\overline{w'^3}}{\left(\overline{w'^2}\right)^2}. \qquad (4.8.11)$$

30

The function $E(w, r_t, \theta_l)$ is

$$E = \frac{1 - \frac{1}{2}\frac{2\alpha}{1-2\alpha}\xi}{1 + \frac{1}{2}\xi}, \tag{4.8.12}$$

where

$$1 + \xi = \frac{1-\alpha}{\alpha}\frac{\tilde{\sigma}_{r_{t2}}}{\tilde{\sigma}_{r_{t1}}}\frac{\tilde{\sigma}_{\theta_{l2}}}{\tilde{\sigma}_{\theta_{l1}}} = \left(\frac{A_{r_t} - B_{r_t}}{-A_{r_t} - B_{r_t}}\right)^{1/2}\left(\frac{A_{\theta_l} - B_{\theta_l}}{-A_{\theta_l} - B_{\theta_l}}\right)^{1/2}, \tag{4.8.13}$$

and

$$A_{\theta_l} = Sk_{\theta_l} - \frac{3}{2}\widehat{c}_{w\theta_l}\widehat{Sk}_w + \frac{1}{2}\widehat{c}^3_{w\theta_l}\widehat{Sk}_w, \tag{4.8.14}$$

$$B_{\theta_l} = \frac{3}{2}\left(4 + \widehat{Sk}^2_w\right)^{1/2}\widehat{c}_{w\theta_l}\left(1 - \widehat{c}^2_{w\theta_l}\right), \tag{4.8.15}$$

and A_{r_t} and B_{r_t} are analogous. We now list two cases in which the expression for E simplifies. First, if

$$\frac{\tilde{\sigma}_{r_{t2}}}{\tilde{\sigma}_{r_{t1}}}\frac{\tilde{\sigma}_{\theta_{l2}}}{\tilde{\sigma}_{\theta_{l1}}} = 1$$

then $E = 0$. This would occur, for instance, if the "widths" of the first and second normal were equal to each other for both r_t and θ_l, that is, if $\tilde{\sigma}_{r_{t2}} = \tilde{\sigma}_{r_{t1}}$ and $\tilde{\sigma}_{\theta_{l2}} = \tilde{\sigma}_{\theta_{l1}}$. Second, if we use the diagnostic ansatz (equation (4.4.10)) for the scalar skewnesses, then

$$\xi = \frac{1 - 2\zeta}{\zeta}, \tag{4.8.16}$$

where

$$\zeta = \alpha + \frac{1}{3}\beta\left(1 - 2\alpha\right). \tag{4.8.17}$$

Then we find

$$E = \frac{2}{3}\beta, \tag{4.8.18}$$

and finally

$$\overline{w'r'_t\theta'_l} = \frac{\frac{1}{3}\beta}{(1 - \tilde{\sigma}^2_w)}\frac{(1 - \delta\lambda_{\theta r})}{(1 - \delta\lambda_w)}\overline{r'_t\theta'_l}\frac{\overline{w'^3}}{\overline{w'^2}}$$
$$+ \frac{1 - \frac{1}{3}\beta}{(1 - \tilde{\sigma}^2_w)^2}\frac{(1 - \delta\lambda_{wr})(1 - \delta\lambda_{w\theta})}{(1 - \delta\lambda_w)^2}\overline{w'r'_t}\,\overline{w'\theta'_l}\frac{\overline{w'^3}}{\left(\overline{w'^2}\right)^2}. \tag{4.8.19}$$

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[1] – 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 2.4). 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.

---

[1]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`.

Listing 5.1: Import statements

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

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

Listing 5.2: Defining symbols

```python
sigma_w = Symbol('\sigma_w')
w_1 = Symbol('w_1')
w_2 = Symbol('w_2')
w_bar = Symbol('\overline{w}')
sigma_lambda_w = Symbol('\sigma_{\lambda w}')
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

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

```python
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))
```

Figure 5.1: Output of listing 5.4

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

Listing 5.5: Calculating and printing the integral

```python
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

$$\overline{w'^2} = -\overline{w}^2\delta + \overline{w}^2 + 2\overline{w}\alpha\delta w_1 - 2\overline{w}\alpha\delta w_2 - 2\overline{w}\alpha w_1 + 2\overline{w}\alpha w_2 + 2\overline{w}\delta w_2 - 2\overline{w}w_2$$
$$-\sigma_w^2\delta + \sigma_w^2 + \sigma_{\lambda w}^2\delta - \alpha\delta w_1^2 + \alpha\delta w_2^2 + \alpha w_1^2 - \alpha w_2^2 - \delta w_2^2 + w_2^2,$$

the actual function we want to check. We can now compare figure 5.2 to the given equation. To do this, we first need to define the equation for equation (4.4.3) in listing 5.6. We can

Listing 5.6: Python function for the second order moment

```python
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_lambda_w=sigma_lambda_w):
    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_lambda_w ** 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

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

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 2.4). As in section 5.2, 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

```
sigma_lambda_theta_l = Symbol('\sigma_{\lambda\\theta_l}')
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}}')
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_lambda_w ** 2,
        rho_w_theta_l * sigma_lambda_w * sigma_lambda_theta_l],
        [rho_w_theta_l * sigma_lambda_w * sigma_lambda_theta_l,
        sigma_lambda_theta_l ** 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 (4.4.15), equation (4.4.5), equation (4.4.3), equation (4.2.3), and equation (4.1.2). We do not list the functions to those equations here, because they are defined the same way like the other equations are defined as functions.

Instead, since we cannot compute the integral analytically, we can create a `dataframe` using `pandas`.[2] The columns for this dataframe are going to be all the inputs we have. To get

---

[2]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`.

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, we are using the integral which has been specified earlier and adding the parameter `method='quad'` 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 pack-

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))
```

age NumPy[3]) 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.

---

[3]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.

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 $\delta \in [0,1)$. The restriction to δ vs the one to α 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 (4.4.6) have a $1 - \delta$ in the denominator.

- In section 4.3, we saw, how the transformation between the sum of two normal distributions and the sum of three normal distributions are working. From those transformations, we see that we want

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

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.

For realizability, it turns out that we want $-1 < \widehat{c}_{w\theta_l}, \widehat{c}_{wr_t}, \widehat{c}_{r_t\theta_l} < 1$. This is for instance:

$$-1 < \widehat{c}_{w\theta_l} < 1 \iff (\widehat{c}_{w\theta_l})^2 < 1$$

$$\iff \left(\frac{1}{(1-\tilde{\sigma}_w^2)^{1/2}} \frac{\overline{w'r_t'}}{\sqrt{\overline{w'^2}}\sqrt{\overline{r_t'^2}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_w}{1-\delta}}} \frac{1}{\sqrt{\frac{1-\delta\lambda_r}{1-\delta}}} \frac{1-\delta\lambda_{wr}}{1-\delta} \right)^2 < 1$$

$$\iff \frac{1}{(1-\tilde{\sigma}_w^2)} \frac{(\overline{w'r_t'})^2}{\overline{w'^2}\,\overline{r_t'^2}} \frac{1-\delta}{1-\delta\lambda_w} \frac{1-\delta}{1-\delta\lambda_r} \frac{(1-\delta\lambda_{wr})^2}{(1-\delta)^2} < 1$$

$$\iff \frac{(1-\delta\lambda_{wr})^2(\overline{w'r_t'})^2}{(1-\delta\lambda_w)(1-\delta\lambda_r)\overline{w'^2}\,\overline{r_t'^2}} < (1-\tilde{\sigma}_w^2)$$

$$\iff \frac{(1-\delta\lambda_{wr})^2(\overline{w'r_t'})^2}{(1-\delta\lambda_w)(1-\delta\lambda_r)\overline{w'^2}\,\overline{r_t'^2}} < 1 - \left(\frac{\sigma_w^2(1-\delta)}{\overline{w'^2}(1-\delta\lambda_w)} \right)$$

$$\iff \frac{(1-\delta\lambda_{wr})^2(\overline{w'r_t'})^2}{(1-\delta\lambda_w)(1-\delta\lambda_r)\overline{w'^2}\,\overline{r_t'^2}} + \frac{\sigma_w^2(1-\delta)}{\overline{w'^2}(1-\delta\lambda_w)} < 1$$

$$\iff \frac{(1-\delta\lambda_{wr})^2(\overline{w'r_t'})^2 + \sigma_w^2(1-\delta)(1-\delta\lambda_r)\overline{r_t'^2}}{(1-\delta\lambda_w)(1-\delta\lambda_r)\overline{w'^2}\,\overline{r_t'^2}} < 1$$

7 Summary

8 Outlook

We have seen that adding a third normal distribution right in the middle of the two defined (simplified) normal distributions does not change the "closed" formulas too much, neither it makes it too complicated. So 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

Harris, Charles R. 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.

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

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

McKinney, Wes. "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.

Meurer, Aaron 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.

A Code

A.1 User Guide

A.1.1 Accessing the code

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

A.1.2 Key files and their purposes

- checked_functions.py: This file houses the definitions of all functions used for checking integrals.

- symbols.py: This file contains definitions of all symbols employed for computations with SymPy, a powerful library for symbolic mathematics.

A.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()`).

A.1.4 Displaying equations effectively

1. Import the `display` function from `IPython.display`.

2. Use `display(..)` to present statements or even equations visually.

A.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.

A.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_lambda_w = Symbol('\\tilde{\sigma}_{\lambda_w}')
sigma_tilde_w = Symbol('\\tilde{\sigma}_w')
sigma_lambda_w = Symbol('\sigma_{\lambda_w}')
sigma_lambda_theta_l = Symbol('\sigma_{\lambda\theta_l}')
sigma_lambda_r_t = Symbol('\sigma_{{\lambda}r_t}')

# 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}')
```

47

```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 = symbols(
    '\overline{r_t} \overline{r_t\'^2}'
)

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}}'
)

# 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_prime_r_t_prime_bar = Symbol('\overline{\\theta_l\'r_t\'}')

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_hat_w = Symbol('\widehat{Sk}_w')
sk_theta_l_hat = Symbol('\widehat{Sk}_{\\theta_l}')
c_w_theta_l_hat = Symbol('\hat{c}_{w\\theta_l}')
G_3 = Symbol('G_3')
G_3_hat = Symbol('\hat{G_3}')
```

```
95   G_w, G_1_w, G_2_w, G_3_w = symbols(
96       'G_{w}(w) G_{1w}(w) G_{2w}(w) G_{3w}(w)'
97   )
98
99   G_w_1_2 = Symbol('G_{w_{12}}')
100
101   G_w_theta = Symbol('G_{w\\theta}(w,\\theta)')
102   G_1_w_theta = Symbol('G_{1w\\theta}(w,\\theta)')
103   G_2_w_theta = Symbol('G_{2w\\theta}(w,\\theta)')
104   G_3_w_theta = Symbol('G_{3w\\theta}(w,\\theta)')
105
106   G_theta_r = Symbol('G_{\\theta r}(\\theta, r)')
107   G_1_theta_r = Symbol('G_{1\\theta r}(\\theta, r)')
108   G_2_theta_r = Symbol('G_{2\\theta r}(\\theta, r)')
109   G_3_theta_r = Symbol('G_{3\\theta r}(\\theta, r)')
110
111   G_w_theta_l_r_t = Symbol('G_{w{\\theta_l}{r_t}}(w,{\\theta_l},{r_t})')
112   G_1_w_theta_l_r_t = Symbol('G_{1w{\\theta_l}{r_t}}(w,{\\theta_l},{r_t})')
113   G_2_w_theta_l_r_t = Symbol('G_{2w{\\theta_l}{r_t}}(w,{\\theta_l},{r_t})')
114   G_3_w_theta_l_r_t = Symbol('G_{3w{\\theta_l}{r_t}}(w,{\\theta_l},{r_t})')
115
116   G_theta, G_1_theta, G_2_theta, G_3_theta = symbols(
117       'G_{\\theta}(\\theta) G_{1\\theta}(\\theta) G_{2\\theta}(\\theta)
         ↪  G_{3\\theta}(\\theta)'
118   )
119
120   rho_w_theta_l = Symbol('\\rho_{w\\theta_l}')
121   rho_w_r_t = Symbol('\\rho_{wr_t}')
122   rho_theta_l_r_t = Symbol('\\rho_{\\theta_lr_t}')
```

Listing A.1: symbols.py

A.3 checked_functions.py

```
1   import sympy as sp
2   from sympy import Rational
3   from sympy.stats import Normal, density
4
5   import symbols as sym
6
7
8   # w equations
9   # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10
11  def w_bar(alpha=sp.abc.alpha, delta=sp.abc.delta, w_1=sym.w_1, w_2=sym.w_2):
12      return ((1 - delta) * alpha * w_1
13              + (1 - delta) * (1 - alpha) * w_2
14              + delta * (alpha * w_1 + (1 - alpha) * w_2))
15
```

```python
16
17  def w_prime_2_bar(delta=sp.abc.delta, alpha=sp.abc.alpha, w_1=sym.w_1, w_2=sym.w_2,
18                    w_bar=sym.w_bar, sigma_w=sym.sigma_w,
                      ↪  sigma_lambda_w=sym.sigma_lambda_w):
19      return (((1 - delta) * alpha * ((w_1 - w_bar) ** 2 + sigma_w ** 2)) +
20              ((1 - delta) * (1 - alpha) * ((w_2 - w_bar) ** 2 + sigma_w ** 2)) +
21              (delta * sigma_lambda_w ** 2))
22
23
24  def w_prime_3_bar(delta=sp.abc.delta, alpha=sp.abc.alpha, w_1=sym.w_1, w_2=sym.w_2,
25                    w_bar=sym.w_bar, sigma_w=sym.sigma_w):
26      return (((1 - delta) * alpha * ((w_1 - w_bar) ** 3 +
27                                      3 * sigma_w ** 2 * (w_1 - w_bar))) +
28              ((1 - delta) * (1 - alpha) * ((w_2 - w_bar) ** 3 +
29                                            3 * sigma_w ** 2 * (w_2 - w_bar))))
30
31
32  def w_prime_4_bar(w_prime_2_bar=sym.w_prime_2_bar,
33                    w_prime_3_bar=sym.w_prime_3_bar,
34                    delta=sp.abc.delta,
35                    sigma_tilde_w=sym.sigma_tilde_w,
36                    sigma_lambda_w=sym.sigma_lambda_w):
37      return (w_prime_2_bar ** 2 *
38              ((1 - delta * (sigma_lambda_w ** 2 / w_prime_2_bar)) ** 2 / (1 - delta)) *
39              (3 * sigma_tilde_w ** 4 +
40               6 * (1 - sigma_tilde_w ** 2) *
41               sigma_tilde_w ** 2 +
42               (1 - sigma_tilde_w ** 2) ** 2) +
43              ((1 / (1 - sigma_tilde_w ** 2)) *
44               (1 / (1 - delta * (sigma_lambda_w ** 2 / w_prime_2_bar))) *
45               (w_prime_3_bar ** 2 / w_prime_2_bar)) +
46              (delta * 3 * sigma_lambda_w ** 4))
47
48
49  # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50
51  # theta_l equations
52  # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
53
54  def theta_l_bar(alpha=sp.abc.alpha, delta=sp.abc.delta,
55                  theta_l_1=sym.theta_l_1, theta_l_2=sym.theta_l_2):
56      return ((1 - delta) * alpha * theta_l_1
57              + (1 - delta) * (1 - alpha) * theta_l_2
58              + delta * (alpha * theta_l_1 + (1 - alpha) * theta_l_2))
59
60
61  def theta_l_prime_2_bar(delta=sp.abc.delta,
62                          alpha=sp.abc.alpha,
63                          theta_l_1=sym.theta_l_1,
64                          theta_l_2=sym.theta_l_2,
65                          theta_l_bar=sym.theta_l_bar,
66                          sigma_theta_l_1=sym.sigma_theta_l_1,
67                          sigma_theta_l_2=sym.sigma_theta_l_2,
68                          sigma_lambda_theta_l=sym.sigma_lambda_theta_l):
```

```python
69        return (((1 - delta) * alpha * ((theta_l_1 - theta_l_bar) ** 2 + sigma_theta_l_1 **
  ↪  2)) +
70                ((1 - delta) * (1 - alpha) *
71                 ((theta_l_2 - theta_l_bar) ** 2 + sigma_theta_l_2 ** 2)) +
72                (delta * sigma_lambda_theta_l ** 2))
73
74
75  def theta_l_prime_3_bar(delta=sp.abc.delta,
76                          alpha=sp.abc.alpha,
77                          theta_l_1=sym.theta_l_1,
78                          theta_l_2=sym.theta_l_2,
79                          theta_l_bar=sym.theta_l_bar,
80                          sigma_theta_l_1=sym.sigma_theta_l_1,
81                          sigma_theta_l_2=sym.sigma_theta_l_2):
82        return (((1 - delta) * alpha *
83                ((theta_l_1 - theta_l_bar) ** 3 +
84                 3 * sigma_theta_l_1 ** 2 * (theta_l_1 - theta_l_bar))) +
85                ((1 - delta) * (1 - alpha) *
86                ((theta_l_2 - theta_l_bar) ** 3 +
87                 3 * sigma_theta_l_2 ** 2 * (theta_l_2 - theta_l_bar))))
88
89
90  # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
91
92  # r_t equations
93  # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
94
95  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):
96        return ((1 - delta) * alpha * r_t_1
97                + (1 - delta) * (1 - alpha) * r_t_2
98                + delta * (alpha * r_t_1 + (1 - alpha) * r_t_2))
99
100
101  def r_t_prime_2_bar(delta=sp.abc.delta,
102                      alpha=sp.abc.alpha,
103                      r_t_1=sym.r_t_1,
104                      r_t_2=sym.r_t_2,
105                      r_t_bar=sym.r_t_bar,
106                      sigma_r_t_1=sym.sigma_r_t_1,
107                      sigma_r_t_2=sym.sigma_r_t_2,
108                      sigma_lambda_r_t=sym.sigma_lambda_r_t):
109        return (((1 - delta) * alpha * ((r_t_1 - r_t_bar) ** 2 + sigma_r_t_1 ** 2)) +
110                ((1 - delta) * (1 - alpha) * ((r_t_2 - r_t_bar) ** 2 + sigma_r_t_2 ** 2)) +
111                (delta * sigma_lambda_r_t ** 2))
112
113
114  # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
115
116  # Mixed equations
117  # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
118
119  def w_prime_theta_l_prime_bar(delta=sp.abc.delta,
120                                alpha=sp.abc.alpha,
121                                w_1=sym.w_1,
```

```
122                                w_2=sym.w_2,
123                                w_bar=sym.w_bar,
124                                theta_l_1=sym.theta_l_1,
125                                theta_l_2=sym.theta_l_2,
126                                theta_l_bar=sym.theta_l_bar,
127                                cov_lambda_w_theta=sym.rho_w_theta_l * sym.sigma_lambda_w
                                ↪   *
128                                        sym.sigma_lambda_theta_l):
129     return ((((1 - delta) * alpha * ((w_1 - w_bar) * (theta_l_1 - theta_l_bar))) +
130             ((1 - delta) * (1 - alpha) * ((w_2 - w_bar) * (theta_l_2 - theta_l_bar)))
131             + delta * cov_lambda_w_theta)
132
133
134 def w_prime_r_t_prime_bar(delta=sp.abc.delta,
135                           alpha=sp.abc.alpha,
136                           w_1=sym.w_1,
137                           w_2=sym.w_2,
138                           w_bar=sym.w_bar,
139                           r_t_1=sym.r_t_1,
140                           r_t_2=sym.r_t_2,
141                           r_t_bar=sym.r_t_bar,
142                           cov_lambda_w_r=sym.rho_w_r_t * sym.sigma_lambda_w *
143                                   sym.sigma_lambda_r_t):
144     return ((((1 - delta) * alpha * ((w_1 - w_bar) * (r_t_1 - r_t_bar))) +
145             ((1 - delta) * (1 - alpha) * ((w_2 - w_bar) * (r_t_2 - r_t_bar)))
146             + delta * cov_lambda_w_r)
147
148
149 def w_prime_2_theta_l_prime_bar(sigma_tilde_w=sym.sigma_tilde_w,
150                                 delta=sp.abc.delta,
151                                 lambda_w_theta=sym.lambda_w_theta,
152                                 lambda_w=sym.lambda_w,
153                                 w_prime_3_bar=sym.w_prime_3_bar,
154                                 w_prime_2_bar=sym.w_prime_2_bar,
155                                 w_prime_theta_l_prime=sym.w_prime_theta_l_prime_bar):
156     return ((1 / (1 - sigma_tilde_w ** 2)) *
157             ((1 - delta * lambda_w_theta) / (1 - delta * lambda_w)) *
158             (w_prime_3_bar / w_prime_2_bar) *
159             w_prime_theta_l_prime)
160
161
162 def w_prime_theta_l_prime_2_bar(delta=sp.abc.delta,
163                                 lambda_w_theta=sym.lambda_w_theta,
164                                 lambda_w=sym.lambda_w,
165                                 sigma_tilde_w=sym.sigma_tilde_w,
166                                 w_prime_3_bar=sym.w_prime_3_bar,
167                                 w_prime_2_bar=sym.w_prime_2_bar,
168                                 w_prime_theta_l_prime_bar=sym.w_prime_theta_l_prime_bar,
169                                 theta_l_prime_3_bar=sym.theta_l_prime_3_bar):
170     return (Rational(2, 3) *
171             ((1 - delta * lambda_w_theta) ** 2 / (1 - delta * lambda_w) ** 2) *
172             (1 / (1 - sigma_tilde_w ** 2) ** 2) *
173             (w_prime_3_bar / w_prime_2_bar ** 2) *
174             w_prime_theta_l_prime_bar ** 2 +
```

```
175                Rational(1, 3) *
176                ((1 - delta * lambda_w) / (1 - delta * lambda_w_theta)) *
177                (1 - sigma_tilde_w ** 2) *
178                ((w_prime_2_bar * theta_l_prime_3_bar) / w_prime_theta_l_prime_bar))
179
180
181    def w_prime_theta_l_prime_2_bar_beta(
182            sigma_tilde_w=sym.sigma_tilde_w,
183            delta=sp.abc.delta,
184            lambda_theta=sym.lambda_theta,
185            lambda_w=sym.lambda_w,
186            w_prime_3_bar=sym.w_prime_3_bar,
187            w_prime_2_bar=sym.w_prime_2_bar,
188            beta=sp.abc.beta,
189            theta_l_prime_2_bar=sym.theta_l_prime_2_bar,
190            lambda_w_theta=sym.lambda_w_theta,
191            w_prime_theta_l_prime_bar=sym.w_prime_theta_l_prime_bar):
192        from sympy import Rational
193        return (
194                (1 / (1 - sigma_tilde_w ** 2)) *
195                ((1 - delta * lambda_theta) / (1 - delta * lambda_w)) *
196                (w_prime_3_bar / w_prime_2_bar) *
197                (
198                        Rational(1, 3) * beta * theta_l_prime_2_bar +
199                        (
200                                ((1 - Rational(1, 3) * beta) / (1 - sigma_tilde_w ** 2)) *
201                                ((1 - delta * lambda_w_theta) ** 2 /
202                                 ((1 - delta * lambda_w) * (1 - delta * lambda_theta))) *
203                                (w_prime_theta_l_prime_bar ** 2 / w_prime_2_bar)
204                        )
205                )
206        )
207
208
209    def w_prime_r_t_prime_theta_l_prime_bar(delta=sp.abc.delta,
210                                             alpha=sp.abc.alpha,
211                                             w_1=sym.w_1,
212                                             w_2=sym.w_2,
213                                             w_bar=sym.w_bar,
214                                             r_t_1=sym.r_t_1,
215                                             r_t_2=sym.r_t_2,
216                                             r_t_bar=sym.r_t_bar,
217                                             theta_l_1=sym.theta_l_1,
218                                             theta_l_2=sym.theta_l_2,
219                                             theta_l_bar=sym.theta_l_bar,
220                                             r_r_t_theta_l=sym.r_r_t_theta_l,
221                                             sigma_r_t_1=sym.sigma_r_t_1,
222                                             sigma_r_t_2=sym.sigma_r_t_2,
223                                             sigma_theta_l_1=sym.sigma_theta_l_1,
224                                             sigma_theta_l_2=sym.sigma_theta_l_2):
225        return (((1 - delta) * alpha * (w_1 - w_bar) *
226                 (
227                        (r_t_1 - r_t_bar) * (theta_l_1 - theta_l_bar) +
228                        r_r_t_theta_l * sigma_r_t_1 * sigma_theta_l_1
```

```
229                 )) +
230                 ((1 - delta) * (1 - alpha) * (w_2 - w_bar) *
231                  (
232                          (r_t_2 - r_t_bar) * (theta_l_2 - theta_l_bar) +
233                          r_r_t_theta_l * sigma_r_t_2 * sigma_theta_l_2
234                 )))


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,
        theta_l_prime_r_t_prime_bar=sym.theta_l_prime_r_t_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)) *
                    theta_l_prime_r_t_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 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_lambda_theta_l *
                                                    sym.sigma_lambda_r_t):
    return ((1 - delta) * alpha * (
```

```python
                     (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)


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

# 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_lambda_theta_l)
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_lambda_w)
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)
```

55

```python
332  G_3_w_theta = Normal(name='G_3_w_theta', mean=sp.Matrix([sym.w_bar, sym.theta_l_bar]),
333                        std=sp.Matrix([
334                            [sym.sigma_lambda_w ** 2,
335                             sym.rho_w_theta_l * sym.sigma_lambda_w *
                              ↪  sym.sigma_lambda_theta_l],
336                            [sym.rho_w_theta_l * sym.sigma_lambda_w *
                              ↪  sym.sigma_lambda_theta_l,
337                             sym.sigma_lambda_theta_l ** 2]
338                        ]))
339  G_3_w_theta_density = sp.simplify(density(G_3_w_theta)(sp.abc.w, sp.abc.theta))
340
341  G_w_theta = ((1 - sp.abc.delta) * sp.abc.alpha * G_1_w_theta_density +
342              (1 - sp.abc.delta) * (1 - sp.abc.alpha) * G_2_w_theta_density +
343              sp.abc.delta * G_3_w_theta_density)
344
345  # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
346
347  mu_1_theta_l_r_t = sp.Matrix([sym.theta_l_1, sym.r_t_1])
348  Sigma_1_theta_l_r_t = sp.Matrix([[sym.sigma_theta_l_1 ** 2,
349                                    sym.r_r_t_theta_l * sym.sigma_theta_l_1 *
                                     ↪  sym.sigma_r_t_1],
350                                   [sym.r_r_t_theta_l * sym.sigma_theta_l_1 *
                                     ↪  sym.sigma_r_t_1,
351                                    sym.sigma_r_t_1 ** 2]])
352
353  G_1_theta_l_r_t = Normal(name='G_1_theta_l_r_t',
354                           mean=mu_1_theta_l_r_t,
355                           std=Sigma_1_theta_l_r_t)
356
357  G_1_theta_l_r_t_density = density(G_1_theta_l_r_t)(sym.theta_l, sym.r_t)
358
359  mu_2_theta_l_r_t = sp.Matrix([sym.theta_l_2, sym.r_t_2])
360  Sigma_2_theta_l_r_t = sp.Matrix(
361      [[sym.sigma_theta_l_2 ** 2,
362        sym.r_r_t_theta_l * sym.sigma_theta_l_2 * sym.sigma_r_t_2],
363       [sym.r_r_t_theta_l * sym.sigma_theta_l_2 * sym.sigma_r_t_2,
364        sym.sigma_r_t_2 ** 2]])
365
366  G_2_theta_l_r_t = Normal(name='G_2_theta_l_r_t',
367                           mean=mu_2_theta_l_r_t,
368                           std=Sigma_2_theta_l_r_t)
369
370  G_2_theta_l_r_t_density = density(G_2_theta_l_r_t)(sym.theta_l, sym.r_t)
371
372  mu_3_theta_l_r_t = sp.Matrix([sym.theta_l_bar, sym.r_t_bar])
373  Sigma_3_theta_l_r_t = sp.Matrix(
374      [[sym.sigma_lambda_theta_l ** 2,
375        sym.rho_theta_l_r_t * sym.sigma_lambda_theta_l * sym.sigma_lambda_r_t],
376       [
377            sym.rho_theta_l_r_t * sym.sigma_lambda_theta_l * sym.sigma_lambda_r_t,
378            sym.sigma_lambda_r_t ** 2]])
379
380  G_3_theta_l_r_t = Normal(name='G_3_theta_l_r_t',
381                           mean=mu_3_theta_l_r_t,
```

```
382                              std=Sigma_2_theta_l_r_t)
383   G_3_theta_l_r_t_density = density(G_3_theta_l_r_t)(sym.theta_l, sym.r_t)
384
385   G_theta_l_r_t = ((1 - sp.abc.delta) * sp.abc.alpha * G_1_theta_l_r_t_density +
386                    (1 - sp.abc.delta) * (1 - sp.abc.alpha) * G_2_theta_l_r_t_density +
387                    sp.abc.delta * G_3_theta_l_r_t_density)
388
389   # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
390
391   mu_1_w_theta_l_r_t = sp.Matrix([sym.w_1, sym.theta_l_1, sym.r_t_1])
392   Sigma_1_w_theta_l_r_t = sp.Matrix(
393       [[sym.sigma_w ** 2,
394         0,
395         0],
396        [0,
397         sym.sigma_theta_l_1 ** 2,
398         sym.r_r_t_theta_l * sym.sigma_theta_l_1 * sym.sigma_r_t_1],
399        [0,
400         sym.r_r_t_theta_l * sym.sigma_theta_l_1 * sym.sigma_r_t_1,
401         sym.sigma_r_t_1 ** 2]])
402
403   G_1_w_theta_l_r_t = Normal(name='G_1_w_theta_l_r_t',
404                              mean=mu_1_w_theta_l_r_t,
405                              std=Sigma_1_w_theta_l_r_t)
406
407   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)
408
409   mu_2_w_theta_l_r_t = sp.Matrix([sym.w_2, sym.theta_l_2, sym.r_t_2])
410   Sigma_2_w_theta_l_r_t = sp.Matrix([[sym.sigma_w ** 2,
411                                       0,
412                                       0],
413                                      [0,
414                                       sym.sigma_theta_l_2 ** 2,
415                                       sym.r_r_t_theta_l * sym.sigma_theta_l_2 *
416                                    ↪  sym.sigma_r_t_2],
                                        [0,
417                                       sym.r_r_t_theta_l * sym.sigma_theta_l_2 *
                                     ↪  sym.sigma_r_t_2,
418                                       sym.sigma_r_t_2 ** 2]])
419
420   G_2_w_theta_l_r_t = Normal(name='G_2_w_theta_l_r_t',
421                              mean=mu_2_w_theta_l_r_t,
422                              std=Sigma_2_w_theta_l_r_t)
423
424   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)
425
426   mu_3_w_theta_l_r_t = sp.Matrix([sym.w_bar, sym.theta_l_bar, sym.r_t_bar])
427   Sigma_3_w_theta_l_r_t = sp.Matrix(
428       [[sym.sigma_lambda_w ** 2,
429         sym.rho_w_theta_l * sym.sigma_lambda_w * sym.sigma_lambda_theta_l,
430         sym.rho_w_r_t * sym.sigma_lambda_theta_l * sym.sigma_lambda_r_t],
431        [
432            sym.rho_w_theta_l * sym.sigma_lambda_w * sym.sigma_lambda_theta_l,
433            sym.sigma_lambda_theta_l ** 2,
```

```python
                sym.rho_theta_l_r_t * sym.sigma_lambda_w * sym.sigma_lambda_r_t],
            [sym.rho_w_r_t * sym.sigma_lambda_theta_l * sym.sigma_lambda_r_t,
             sym.rho_theta_l_r_t * sym.sigma_lambda_w * sym.sigma_lambda_r_t,
             sym.sigma_lambda_r_t ** 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)
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))))


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

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

def lambda_w(sigma_lambda_w=sym.sigma_lambda_w, w_prime_2_bar=sym.w_prime_2_bar):
    return sigma_lambda_w ** 2 / w_prime_2_bar


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


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


def lambda_w_theta(cov_lambda_w_theta=
                   sym.rho_w_theta_l * sym.sigma_lambda_w * sym.sigma_lambda_theta_l,
                   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_lambda_w * sym.sigma_lambda_r_t,
```

```python
                      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_lambda_r_t *
                   ↪  sym.sigma_lambda_theta_l,
                   r_t_prime_theta_l_prime_bar=sym.theta_l_prime_r_t_prime_bar):
    return cov_lambda_r_theta / r_t_prime_theta_l_prime_bar


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

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

def sk_theta_l_hat_beta(sk_hat_w=sym.sk_hat_w, c_hat_w_theta_l=sym.c_w_theta_l_hat,
                        beta=sp.abc.beta):
    return sk_hat_w * 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_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)))


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)) *
```

```
540            (w_prime_theta_l_prime_bar / ((sqrt(w_prime_2_bar)) *
               ↪  sqrt(theta_l_prime_2_bar))) *
541            (1 / sqrt((1 - delta * lambda_w) / (1 - delta))) *
542            (1 / sqrt((1 - delta * lambda_theta) / (1 - delta))) *
543            ((1 - delta * lambda_w_theta) / (1 - delta)))
544
545
546  # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
547
548  # sk
549  # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
550
551  def beta(c_w_theta_l_hat=sym.c_w_theta_l_hat,
552           sk_w_hat=sym.sk_hat_w,
553           sk_theta_l_hat=sym.sk_theta_l_hat):
554      return ((((c_w_theta_l_hat ** 3 * sk_w_hat) - sk_theta_l_hat) /
555              (c_w_theta_l_hat * sk_w_hat * (c_w_theta_l_hat ** 2 - 1))))
556
557  # -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Listing A.2: checked_functions.py

A.4 w_prime_4_bar.py

```
1   #!/usr/bin/env python
2   # coding: utf-8
3
4   # In[2]:
5
6
7   import sympy as sp
8   from IPython.display import display
9   from sympy import abc, oo
10
11  import checked_functions as c_f
12  import symbols as sym
13
14  # # This document aims to analytically check £\overline{w'^4}£
15
16  # ## Define the marginal distributions with those parameters.
17
18  # In[2]:
19
20
21  display(sp.Eq(sym.G_1_w, c_f.G_1_theta_l_density))
22
23  # In[3]:
24
25
```

```
26  display(sp.Eq(sym.G_2_w, c_f.G_2_theta_l_density))
27
28  # In[4]:
29
30
31  display(sp.Eq(sym.G_3_w, c_f.G_3_theta_l_density))
32
33  # In[5]:
34
35
36  display(sp.Eq(sym.G_w, c_f.G_w))
37
38  # Calculate the moment analytically:
39
40  # In[6]:
41
42
43  w_prime_4_bar_int = sp.Integral((sp.abc.w - sym.w_bar) ** 4 * c_f.G_w, [sp.abc.w, -oo,
    ↪  oo])
44  display(sp.Eq(sym.w_prime_4_bar, w_prime_4_bar_int))
45
46  # In[7]:
47
48
49  w_prime_4_bar_int_val = w_prime_4_bar_int.doit(conds='none').simplify()
50  display(sp.Eq(sym.w_prime_4_bar, w_prime_4_bar_int_val))
51
52  # The equation in the document is:
53
54  # In[8]:
55
56
57  display(sp.Eq(sym.w_prime_4_bar, c_f.w_prime_4_bar()))
58
59  # where
60
61  # In[9]:
62
63
64  display(sp.Eq(sym.sigma_tilde_w, c_f.sigma_tilde_w()))
65
66  # and
67
68  # In[10]:
69
70
71  display(sp.Eq(sym.w_prime_2_bar, c_f.w_prime_2_bar()))
72
73  # and
74
75  # In[11]:
76
77
78  display(sp.Eq(sym.w_prime_3_bar, c_f.w_prime_2_bar()))
```

```
79
80   # So,
81
82   # In[12]:
83
84
85   lambda_w_val = c_f.lambda_w().subs({
86       sym.w_prime_2_bar: c_f.w_prime_2_bar()
87   })
88   display(sp.Eq(sym.lambda_w, lambda_w_val))
89
90   # In[13]:
91
92
93   w_prime_4_bar_check_val = c_f.w_prime_4_bar().subs({
94       sym.w_prime_2_bar: c_f.w_prime_2_bar(),
95       sym.w_prime_3_bar: c_f.w_prime_3_bar(),
96       sym.sigma_tilde_w: c_f.sigma_tilde_w().subs({
97           sym.w_prime_2_bar: c_f.w_prime_2_bar(),
98           sym.lambda_w: c_f.lambda_w()
99       })
100  })
101
102  display(sp.Eq(sym.w_prime_4_bar, w_prime_4_bar_check_val))
103
104  # In[14]:
105
106
107  display(sp.Eq(w_prime_4_bar_int_val, w_prime_4_bar_check_val, evaluate=True)
108          .subs({sym.w_bar: c_f.w_bar()}).simplify())
```

Listing A.3: w_prime_4_bar.py

A.5 w_prime_2_theta_l_prime_bar.py

```
1    #!/usr/bin/env python
2    # coding: utf-8
3
4    # In[2]:
5
6
7    import sympy as sp
8    from IPython.display import display
9    from sympy import abc, oo
10
11   import checked_functions as c_f
12   import symbols as sym
13
14   # # This document aims to analytically check £\overline{w'^4}£
```

```python
15
16   # ## Define the marginal distributions with those parameters.
17
18   # In[2]:
19
20
21   display(sp.Eq(sym.G_1_w, c_f.G_1_theta_l_density))
22
23   # In[3]:
24
25
26   display(sp.Eq(sym.G_2_w, c_f.G_2_theta_l_density))
27
28   # In[4]:
29
30
31   display(sp.Eq(sym.G_3_w, c_f.G_3_theta_l_density))
32
33   # In[5]:
34
35
36   display(sp.Eq(sym.G_w, c_f.G_w))
37
38   # Calculate the moment analytically:
39
40   # In[6]:
41
42
43   w_prime_4_bar_int = sp.Integral((sp.abc.w - sym.w_bar) ** 4 * c_f.G_w, [sp.abc.w, -oo,
     ↪  oo])
44   display(sp.Eq(sym.w_prime_4_bar, w_prime_4_bar_int))
45
46   # In[7]:
47
48
49   w_prime_4_bar_int_val = w_prime_4_bar_int.doit(conds='none').simplify()
50   display(sp.Eq(sym.w_prime_4_bar, w_prime_4_bar_int_val))
51
52   # The equation in the document is:
53
54   # In[8]:
55
56
57   display(sp.Eq(sym.w_prime_4_bar, c_f.w_prime_4_bar()))
58
59   # where
60
61   # In[9]:
62
63
64   display(sp.Eq(sym.sigma_tilde_w, c_f.sigma_tilde_w()))
65
66   # and
67
```

63

```
68   # In[10]:
69
70
71   display(sp.Eq(sym.w_prime_2_bar, c_f.w_prime_2_bar()))
72
73   # and
74
75   # In[11]:
76
77
78   display(sp.Eq(sym.w_prime_3_bar, c_f.w_prime_2_bar()))
79
80   # So,
81
82   # In[12]:
83
84
85   lambda_w_val = c_f.lambda_w().subs({
86       sym.w_prime_2_bar: c_f.w_prime_2_bar()
87   })
88   display(sp.Eq(sym.lambda_w, lambda_w_val))
89
90   # In[13]:
91
92
93   w_prime_4_bar_check_val = c_f.w_prime_4_bar().subs({
94       sym.w_prime_2_bar: c_f.w_prime_2_bar(),
95       sym.w_prime_3_bar: c_f.w_prime_3_bar(),
96       sym.sigma_tilde_w: c_f.sigma_tilde_w().subs({
97           sym.w_prime_2_bar: c_f.w_prime_2_bar(),
98           sym.lambda_w: c_f.lambda_w()
99       })
100  })
101
102  display(sp.Eq(sym.w_prime_4_bar, w_prime_4_bar_check_val))
103
104  # In[14]:
105
106
107  display(sp.Eq(w_prime_4_bar_int_val, w_prime_4_bar_check_val, evaluate=True)
108          .subs({sym.w_bar: c_f.w_bar()}).simplify())
```

Listing A.4: w_prime_2_theta_l_prime_bar.py

A.6 w_prime_theta_l_prime_2_bar.py

```
1   #!/usr/bin/env python
2   # coding: utf-8
3
```

```python
# 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'\theta_l'^2}£

# ## 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_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]:
```

65

```
58   w_prime_theta_l_2_prime_bar = w_prime_theta_l_2_prime_bar.subs({
59       sym.w_bar: c_f.w_bar(),
60       sym.theta_l_bar: c_f.theta_l_bar()
61   })
62
63   display(sp.Eq(sym.w_prime_theta_prime_l_2_bar, w_prime_theta_l_2_prime_bar))
64
65   # The equation in the document is:
66
67   # In[8]:
68
69
70   display(sp.Eq(sym.w_prime_theta_prime_l_2_bar, c_f.w_prime_theta_l_prime_2_bar()))
71
72   # For this we still need £\tilde{\sigma_w}£:
73
74   # In[9]:
75
76
77   display(sp.Eq(sym.sigma_tilde_w, c_f.sigma_tilde_w()))
78
79   # And £\overline{w'^{2}}£:
80
81   # In[10]:
82
83
84   display(sp.Eq(sym.w_prime_2_bar, c_f.w_prime_2_bar()))
85
86   # And £\overline{w'^{3}}£:
87
88   # In[11]:
89
90
91   display(sp.Eq(sym.w_prime_3_bar, c_f.w_prime_3_bar()))
92
93   # And £\overline{w'\theta'_l}£:
94
95   # In[12]:
96
97
98   display(sp.Eq(sym.w_prime_theta_l_prime_bar, c_f.w_prime_theta_l_prime_bar()))
99
100  # And £\lambda_w£:
101
102  # In[13]:
103
104
105  display(sp.Eq(sym.lambda_w, c_f.lambda_w()))
106
107  # And £\lambda_{w\theta}£:
108
109  # In[14]:
110
111
```

```python
112  display(sp.Eq(sym.lambda_w_theta, c_f.lambda_w_theta()))
113
114  # And £\overline{\theta_l'^3}£:
115
116  # In[15]:
117
118
119  display(sp.Eq(sym.theta_l_prime_3_bar, c_f.theta_l_prime_3_bar()))
120
121  # Putting those all together yields:
122
123  # In[16]:
124
125
126  w_prime_theta_l_prime_2_bar_check_val = c_f.w_prime_theta_l_prime_2_bar().subs({
127      sym.theta_l_prime_3_bar: c_f.theta_l_prime_3_bar(),
128      sym.sigma_tilde_w: c_f.sigma_tilde_w(),
129      sym.lambda_w: c_f.lambda_w(),
130      sym.w_prime_2_bar: c_f.w_prime_2_bar(),
131      sym.w_prime_3_bar: c_f.w_prime_3_bar(),
132      sym.lambda_w_theta: c_f.lambda_w_theta(),
133      sym.w_prime_theta_l_prime_bar: c_f.w_prime_theta_l_prime_bar()
134  })
135
136  w_prime_theta_l_prime_2_bar_check_val = w_prime_theta_l_prime_2_bar_check_val.subs({
137      sym.w_prime_2_bar: c_f.w_prime_2_bar(),
138      sym.lambda_w: c_f.lambda_w(),
139      sym.w_bar: c_f.w_bar(),
140      sym.theta_l_bar: c_f.theta_l_bar()
141  })
142
143  display(sp.Eq(sym.w_prime_theta_prime_l_2_bar, w_prime_theta_l_prime_2_bar_check_val))
144
145  # 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.
146
147  # We create a dataframe to get all possible permutations and therefore also all possible
     ↪   evaluations of the integrals.
148
149  # In[17]:
150
151
152  df = pd.DataFrame(
153      product([0, 1],
154              [-2, 2],
155              [-1, 2],
156              [0, 3],
157              [Rational(1, 10)],
158              [Rational(3, 10)],
159              [Rational(4, 10)],
160              [Rational(7, 10)],
161              [Rational(6, 10)],
162              [Rational(5, 10)],
```

```
163                  [Rational(1, 10), Rational(5, 10)],
164                  [Rational(5, 10)]),
165          columns=[sym.w_1,
166                   sym.w_2,
167                   sym.theta_l_1,
168                   sym.theta_l_2,
169                   sym.sigma_theta_l_1,
170                   sym.sigma_theta_l_2,
171                   sym.sigma_lambda_theta_l,
172                   sym.sigma_w,
173                   sym.sigma_lambda_w,
174                   sp.abc.alpha,
175                   sp.abc.delta,
176                   sym.rho_w_theta_l])
177
178      # In[18]:
179
180
181      df['check_val'] = (
182          df.apply(lambda x:
183                  w_prime_theta_l_prime_2_bar_check_val.subs({
184                      sym.w_1: x[sym.w_1],
185                      sym.w_2: x[sym.w_2],
186                      sym.theta_l_1: x[sym.theta_l_1],
187                      sym.theta_l_2: x[sym.theta_l_2],
188                      sym.sigma_theta_l_1: x[sym.sigma_theta_l_1],
189                      sym.sigma_theta_l_2: x[sym.sigma_theta_l_2],
190                      sym.sigma_lambda_theta_l: x[sym.sigma_lambda_theta_l],
191                      sym.sigma_w: x[sym.sigma_w],
192                      sym.sigma_lambda_w: x[sym.sigma_lambda_w],
193                      sp.abc.alpha: x[sp.abc.alpha],
194                      sp.abc.delta: x[sp.abc.delta],
195                      sym.rho_w_theta_l: x[sym.rho_w_theta_l]
196                  }).evalf(), axis=1))
197
198      # Calculate the moment analytically:
199
200      # In[19]:
201
202
203      df['num_int'] = (
204          df.apply(lambda x: w_prime_theta_l_2_prime_bar.subs({
205              sym.w_1: x[sym.w_1],
206              sym.w_2: x[sym.w_2],
207              sym.theta_l_1: x[sym.theta_l_1],
208              sym.theta_l_2: x[sym.theta_l_2],
209              sym.sigma_theta_l_1: x[sym.sigma_theta_l_1],
210              sym.sigma_theta_l_2: x[sym.sigma_theta_l_2],
211              sym.sigma_lambda_theta_l: x[sym.sigma_lambda_theta_l],
212              sym.sigma_w: x[sym.sigma_w],
213              sym.sigma_lambda_w: x[sym.sigma_lambda_w],
214              sp.abc.alpha: x[sp.abc.alpha],
215              sp.abc.delta: x[sp.abc.delta],
216              sym.rho_w_theta_l: x[sym.rho_w_theta_l]
```

```
217        }).doit(conds='none', method='quad').evalf(), axis=1))
218
219    # In[20]:
220
221
222    df['diff'] = abs(df['check_val'] - df['num_int'])
223
224    # In[21]:
225
226
227    df['diff_num'] = abs(df['check_val'].astype(float) - df['num_int'].astype(float))
228
229    # In[22]:
230
231
232    display(df)
233
234    # In[23]:
235
236
237    import numpy as np
238
239    print('The mean error between the rhs and the lhs is:', np.mean(df['diff_num']))
```

Listing A.5: w_prime_theta_l_prime_2_bar.py

A.7 w_prime_r_t_prime_theta_l_prime_bar.py

```
1    #!/usr/bin/env python
2    # coding: utf-8
3
4    # In[1]:
5
6
7    from itertools import product
8
9    import pandas as pd
10
11   pd.set_option('display.max_columns', None)
12   pd.set_option('display.max_rows', None)
13
14   import sympy as sp
15   from IPython.display import display
16   from sympy import abc, oo, Rational
17
18   import checked_functions as c_f
19   import symbols as sym
20
21   # # This document aims to numerically check £\overline{w'r_t'\theta_l'}£
```

```
22
23    # ## Define the normal distributions.
24
25    # In[2]:
26
27
28    display(sp.Eq(sp.symbols('\mu_1'), c_f.mu_1_w_theta_l_r_t, evaluate=False))
29    display(sp.Eq(sp.symbols('\Sigma_1'), c_f.Sigma_1_w_theta_l_r_t, evaluate=False))
30    # display(sp.Eq(sym.G_1_w_theta_l_r_t, c_f.G_1_w_theta_l_r_t_density))
31
32
33    # In[3]:
34
35
36    display(sp.Eq(sp.symbols('\mu_2'), c_f.mu_2_w_theta_l_r_t, evaluate=False))
37    display(sp.Eq(sp.symbols('\Sigma_2'), c_f.Sigma_2_w_theta_l_r_t, evaluate=False))
38    # display(sp.Eq(sym.G_2_w_theta_l_r_t, c_f.G_2_w_theta_l_r_t_density))
39
40
41    # In[4]:
42
43
44    display(sp.Eq(sp.symbols('\mu_3'), c_f.mu_3_w_theta_l_r_t, evaluate=False))
45    display(sp.Eq(sp.symbols('\Sigma_3'), c_f.Sigma_3_w_theta_l_r_t, evaluate=False))
46    # display(sp.Eq(sym.G_3_w_theta_l_r_t, c_f.G_3_w_theta_l_r_t_density))
47
48
49    # In[5]:
50
51
52    display(sp.Eq(sym.G_w_theta_l_r_t, c_f.G_w_theta_l_r_t))
53
54    # In[6]:
55
56
57    w_prime_r_t_prime_theta_l_prime_bar = sp.Integral(
58        (sp.abc.w - sym.w_bar) * (sym.r_t - sym.r_t_bar) * (
59                    sp.abc.theta - sym.theta_l_bar) * c_f.G_w_theta_l_r_t,
60        [sp.abc.w, -oo, oo],
61        [sym.theta_l, -oo, oo],
62        [sym.r_t, -oo, oo])
63    display(sp.Eq(sym.w_prime_r_t_prime_theta_l_prime_bar,
      ↪  w_prime_r_t_prime_theta_l_prime_bar))
64
65    # In[7]:
66
67
68    w_prime_r_t_prime_theta_l_prime_bar = (
69        w_prime_r_t_prime_theta_l_prime_bar.subs({
70            sym.w_bar: c_f.w_bar(),
71            sym.r_t_bar: c_f.r_t_bar(),
72            sym.theta_l_bar: c_f.theta_l_bar()
73        }))
```

```
74  display(sp.Eq(sym.w_prime_r_t_prime_theta_l_prime_bar,
    ↪   w_prime_r_t_prime_theta_l_prime_bar))

75

76  # The equation in the document is:

77

78  # In[8]:

79


80

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

82

83  # 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.

84

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

86

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

88

89  # In[9]:

90

91

92  df = pd.DataFrame(
93      product([3, 1],
94              [-2],
95              [-1, 3],
96              [2],
97              [1, 4],
98              [2],
99              [1.1],
100             [1.3],
101             [1.4],
102             [1.7],
103             [1.2],
104             [1.5],
105             [1.9],
106             [1.6],
107             [.55],
108             [.8],
109             [.65],
110             [.45],
111             [.35],
112             [.5]),
113     columns=[sym.w_1,
114             sym.w_2,
115             sym.theta_l_1,
116             sym.theta_l_2,
117             sym.r_t_1,
118             sym.r_t_2,
119             sym.sigma_theta_l_1,
120             sym.sigma_theta_l_2,
121             sym.sigma_lambda_theta_l,
```

```
122              sym.sigma_w,
123              sym.sigma_r_t_1,
124              sym.sigma_r_t_2,
125              sym.sigma_lambda_r_t,
126              sym.sigma_lambda_w,
127              sp.abc.alpha,
128              sp.abc.delta,
129              sym.rho_w_theta_l,
130              sym.rho_w_r_t,
131              sym.rho_theta_l_r_t,
132              sym.r_r_t_theta_l])
133
134  # In[10]:
135
136
137  w_prime_r_t_prime_theta_l_prime_bar_check_sym_val = (
138      c_f.w_prime_r_t_prime_theta_l_prime_bar().subs({
139          sym.w_bar: c_f.w_bar(),
140          sym.r_t_bar: c_f.r_t_bar(),
141          sym.theta_l_bar: c_f.theta_l_bar()
142      }))
143
144  # In[11]:
145
146
147  df['check_val'] = (
148      df.apply(lambda x: Rational(c_f.w_prime_r_t_prime_theta_l_prime_bar().subs({
149          sym.w_bar: c_f.w_bar(),
150          sym.theta_l_bar: c_f.theta_l_bar(),
151          sym.r_t_bar: c_f.r_t_bar(),
152          sym.w_1: x[sym.w_1],
153          sym.w_2: x[sym.w_2],
154          sym.theta_l_1: x[sym.theta_l_1],
155          sym.theta_l_2: x[sym.theta_l_2],
156          sym.r_t_1: x[sym.r_t_1],
157          sym.r_t_2: x[sym.r_t_2],
158          sym.sigma_theta_l_1: x[sym.sigma_theta_l_1],
159          sym.sigma_theta_l_2: x[sym.sigma_theta_l_2],
160          sym.sigma_r_t_1: x[sym.sigma_r_t_1],
161          sym.sigma_r_t_2: x[sym.sigma_r_t_2],
162          sp.abc.alpha: x[sp.abc.alpha],
163          sp.abc.delta: x[sp.abc.delta],
164          sym.r_r_t_theta_l: x[sym.r_r_t_theta_l]
165      }).evalf()), axis=1))
166
167  # Calculate the moment numerically:
168
169  # In[12]:
170
171
172  import scipy
173
174  df['num_int'] = df.apply(lambda x: scipy.integrate.nquad(
175      sp.lambdify(
```

```
176              [sp.abc.w, sym.r_t, sym.theta_l],
177              (((sp.abc.w - c_f.w_bar()) *
178                (sym.r_t - c_f.r_t_bar()) *
179                (sym.theta_l - c_f.theta_l_bar()) *
180                c_f.G_w_theta_l_r_t))
181              .subs({
182                  sym.w_bar: c_f.w_bar(),
183                  sym.r_t_bar: c_f.r_t_bar(),
184                  sym.theta_l_bar: c_f.theta_l_bar(),
185                  sym.w_1: x[sym.w_1],
186                  sym.w_2: x[sym.w_2],
187                  sym.theta_l_1: x[sym.theta_l_1],
188                  sym.theta_l_2: x[sym.theta_l_2],
189                  sym.r_t_1: x[sym.r_t_1],
190                  sym.r_t_2: x[sym.r_t_2],
191                  sym.sigma_w: x[sym.sigma_w],
192                  sym.sigma_theta_l_1: x[sym.sigma_theta_l_1],
193                  sym.sigma_theta_l_2: x[sym.sigma_theta_l_2],
194                  sym.sigma_lambda_theta_l: x[sym.sigma_lambda_theta_l],
195                  sym.sigma_lambda_w: x[sym.sigma_lambda_w],
196                  sym.sigma_lambda_r_t: x[sym.sigma_lambda_r_t],
197                  sym.sigma_r_t_1: x[sym.sigma_r_t_1],
198                  sym.sigma_r_t_2: x[sym.sigma_r_t_2],
199                  sp.abc.alpha: x[sp.abc.alpha],
200                  sp.abc.delta: x[sp.abc.delta],
201                  sym.rho_w_theta_l: x[sym.rho_w_theta_l],
202                  sym.rho_w_r_t: x[sym.rho_w_r_t],
203                  sym.rho_theta_l_r_t: x[sym.rho_theta_l_r_t],
204                  sym.r_r_t_theta_l: x[sym.r_r_t_theta_l]
205              })),
206          ranges=[[-30, 30], [-30, 30], [-30, 30]])[0],
207                          axis=1)
208
209  # In[13]:
210
211
212  df['diff'] = abs(df['check_val'] - df['num_int'])
213
214  # In[14]:
215
216
217  df['diff_num'] = abs(df['check_val'].astype(float) - df['num_int'])
218
219  # In[15]:
220
221
222  display(df)
223
224  # In[16]:
225
226
227  import numpy as np
228
```

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

Listing A.6: w_prime_r_t_prime_theta_l_prime_bar.py