The formal generation of models for scientific simulations [1]

Daniel Tang [2]

September, 2010

[1]Submitted in accordance with the requirements for the degree of Phd

[2]University of Leeds, School of Earth and Environment

The candidate confirms that the work submitted is his own and that appropriate credit had been given where reference has been made to the work of others.

I would like to thank Steven Dobbie for all his comments and suggestions over the last four years, for listening calmly to my long, crypto-mathematical rantings and for bravely reading through the visions and revisions that would become this thesis. I would also like to thank Nik Stott for providing much needed motivation and for his comments on the more important parts of this thesis. Thanks also to Jonathan Chrimes for supplying figures from the DYCOMS-II intercomparison study, and to Wayne and Jane at Northern Tea Power for supplying the necessary coffee. Thanks go to Zen Internet for the loan of a computer for the stratocumulus experiment and to the Natural Environment Research Council for funding this research under award number NER/S/A/2006/14148.

Contents

List of Figures

List of Tables

Abstract

It is now commonplace for complex physical systems such as the climate system to be studied indirectly via computer simulations. Often, the equations that govern the underlying physical system are known but detailed or high-resolution computer models of these equations ("governing models") are not practical because of limited computational resources; so the models are simplified or "parameterised". However, if the output of a simplified model is to lead to conclusions about a physical system, we must prove that these outputs reflect reality and are not merely artifacts of the simplifications. At present, simplifications are usually based on informal, ad-hoc methods making it difficult or impossible to provide such a proof rigorously. Here we introduce a set of formal methods for generating computer models. We present a newly developed computer program, "iGen", which syntactically analyses the computer code of a high-resolution, governing model and, without executing it, automatically produces a much faster, simplified model with provable bounds on error compared to the governing model. These bounds allow scientists to rigorously distinguish real world phenomena from artifact in subsequent numerical experiments using the simplified model. Using simple physical systems as examples, we illustrate that iGen produces simplified models that execute typically orders of magnitude faster than their governing models. Finally, iGen is used to generate a model of entrainment in marine stratocumulus. The resulting simplified model is appropriate for use as part of a parameterisation of marine stratocumulus in a Global Climate Model.

Preface

A note to the reader

Every day, scientists create scientific knowledge by modelling physical systems on computers. Scientists are well aware of the limitations and deficiencies of their models but they rarely stop and ask themselves "is this the best way of using a computer to do science?" This report describes a vision of how we ought to be using computers to do science.

As part of our approach, great importance will be put on the formality of the proposed methods, and all theoretical claims will be supported by formal proofs. These are presented at the the end of each relevant chapter rather than in the main text; thus allowing the reader to get to grips with the main ideas before getting embroiled with the details of the proofs.

After an introduction, the remaining chapters are written in a 'top-down' fashion, starting on the most abstract, theoretical level and gradually getting more concrete and practical. Chapter 2 is an important chapter but it introduces some quite abstract concepts. If the reader prefers a 'bottom-up' approach, starting with specifics and gradually generalising on these, they may wish to skim read this chapter at first and return to it later when the application of the concepts can be more clearly seen.

Chapter 1

Introduction

Sir Francis Bacon[1], declared in his Novum Organum (1620) that the purpose of science is to improve man's lot on Earth. If climatologists are to live up to this vision with respect to the humanitarian challenges of climate change then they must communicate justified beliefs about future climate to policymakers so that they can make informed policy decisions. If we are to take a decision theoretic viewpoint (see, e.g. Jeffrey, 1990) the information the policymakers need is contained in the probability distributions of certain sociologically poignant climate observables such as average surface temperatures or rainfalls under various emissions scenarios. Climatologists rely heavily on the use of climate models to generate beliefs about these observables. If a model simulates, say, a warmer world under a given emissions scenario then we are inclined to believe that the world would indeed be warmer if we were to make those emissions. However, a sceptic may question this deduction. Can we really be sure that the model is right? What reason does the scientist have to believe in the model projections? When the stakes are as high as they are in the case of climate change, it is quite reasonable to insist that there should be a good justification for believing in climate model results.

Suppose we had a climate model that not only produced climate projections but also produced bounds on the error and uncertainty in these projections. If these error/uncertainty bounds could be produced in such a way that we could justify them with a line of reasoning starting from a set of generally acceptable

[1]Lord keeper of The great seal and Baron of Verulam

assumptions then we could be sure that the model results were trustworthy to within these bounds. In this way we could satisfy any skeptic that our beliefs in future climate, as informed by the model's projections, are indeed justified and provide the policymakers with the information they need. In this article we will describe how climate models could be generated in order to make this possible.

The problem of simulating the climate can conveniently be split into two parts: firstly, there is the problem of understanding the physics of the climate system and describing it as a set of equations, the so called 'equations of motion'. Secondly, there is the problem of how to use these equations to answer pertinent questions about the climate (and indeed, to complete the hermenutic circle, to inform further experiment). The skeptic can be convinced in the equations of motion by appeal to the scientific method, one would hope. Although there is much to say about the scientific method and how it manages to justify beliefs (see, e.g., Chalmers, 1999), the second part of this problem is much less well understood and is the focus of our interest here.

Even if we are given the equations of motion of the climate system, it is difficult to answer practical questions about the climate because present day computers just aren't powerful enough to directly solve these equations and generate climate projections. So, climate models instead solve a simplified set of equations. These simplifications, called 'parameterisations', are not supported by the same weight of scientific evidence as the physical equations of motion and so the sceptic may ask what reason we have to believe that these simplifications do not affect the veracity of the model results. Let us look briefly at what arguments modellers have used to show that the simplifications in global climate models (GCM's) are justified. The fourth assessment report of the IPCC (IPCC, 2007) gives a good review of the literature on this subject, the arguments set out there are all informal arguments that follow one of the following formats:

1. The algorithm of the GCM is shown from first principles to be a faithful 'representation' of the physical processes described by the generally accepted physical laws.

2. The GCM is shown to agree with a set of observations and a demonstration given that faithfulness to the set of observations implies a faithful projection of the required climate observable.

3. The parts of the GCM are shown to more or less accurately agree with

a set of observations when run in isolation, and a demonstration is given that the faithfulness of the parts to the observations implies that the GCM will give a faithful projection of the required climate observable.

4. GCMs from different research groups are compared and shown to give comparable results.

5. Multi-model ensemble predictions have been shown to perform better than a single model, supposedly confirming the hypothesis that models are perturbations of reality.

Beginning with the first argument, all non-trivial parts of a GCM are designed with some kind of reference to the underlying equations of motion and the parts are assembled with the intent of representing the whole climate system. However, as already mentioned, the use of parameterisations means that the model does not solve the physical equations. Here are some examples of the simplifications made in models:

False assumptions The physical justifications of many GCM components depend on assumptions that are known to be false, and no proof is given that such false assumptions do not adversely affect the performance of the components. Examples include the assumption that the sea has the viscosity of honey (Chassignet and Garrao, 2001) and cumulus clouds act like conical plumes (Gregory and Rowntree, 1990).

Unjustified generalisation Many GCM components contain 'tunable parameters' which are tuned to minimise the difference between the GCM output and some set of empirical measurements (e.g. Smith, 1990). However, no justification is given for supposing that this tuning will generalise to give good agreement with empirical measurements not used in the tuning process.

Missing processes Many physical processes which may have an effect on climate are not simulated by GCMs. For example, many GCM's do not include a carbon cycle; when this is included, many important processes in the cycle are omitted. This and many other missing processes are discussed in Lemoine (2010), for example.

Logical inconsistencies Different parts of a GCM often make conflicting assumptions, so it is not well defined what the GCM as a whole is simulating.

> For example, all GCMs split the atmosphere into a 'dynamical core' and a 'model physics'. The dynamical core treats, say, the temperature field as a sampled, bandwidth limited field, while the model physics treats the same field as a Reynolds averaged field. The two interpretations are incompatible. See theorem 1.2.1 for more detail.

No proof has been given that these simplifications do not adversely affect model output. Without this, the argument from physical representation is incomplete.

The second line of argument, that of showing that the GCM correctly predicts certain observations, is also problematic. The problem turns on the question "how many correct predictions must a model make before we should be confident that it is a good model of reality?" In theorem 1.2.2 I present an argument that shows, perhaps surprisingly, that in the absence of any other information, the amount of information a model must correctly predict about an observable, o, before it can be considered to be a good model of o is equal to the length of the shortest computer program that correctly models o (i.e. the Kolmogorov complexity of an infinite time series of the observable). It is doubtful that the amount of mutual information that has been shown to exist between GCM output and observations of climate exceeds the length of the shortest possible climate model. Take, for example, the global average surface temperature record over the last century, as presented in the IPCC fourth assessment report (IPCC, 2007). This gives, accounting for the error between modelled and observed values, only around 200 bits of mutual information. It seems doubtful that there exists a climate model that can be expressed in 200 bits of information, so this falls far short of the amount required.

The third approach, that of separately validating the parts, is perhaps the most realistic as it would be easier to validate a part against a large set of observations. However, surprisingly, there has been no formal proof to show how errors in the GCM parts affect errors in the GCM as a whole. Nor has there been any investigation into the effects of the contradictory definitions used in the different parts, as discussed in theorem 1.2.1. Without this, the argument is incomplete.

The fourth and fifth arguments require that the GCMs are perturbations of the real climate system and that the perturbations in the GCMs from different research groups give rise to noise that is not correlated between research groups. While GCM design does differ between research groups, a single common ap-

proximation would break this argument (since it may introduce an error across all GCMs). Indeed, there are a number of approximations that are common to all GCMs (consider only the common missing processes, as discussed above). Moreover, if the noise were uncorrelated we would expect to find that as we increase the number, n, of members in a multi-model ensemble, the error should tend to zero as $\frac{1}{\sqrt{n}}$. However, this is not found to be the case (Knutti, 2008), indicating that the errors in the models are not independent. See also Lemoine (2010) for more discussion on shared model biases.

On the strength of the above, the skeptic could argue that the inter-model ensemble average equilibrium climate sensitivity, as published by the IPCC (2007), is an *average of simplifications* that has no *provable* connection to the expectation value of climate sensitivity given our current state of knowledge. He could argue that the inter-model ensemble average spread is a *spread of disagreement* between models that has no *provable* connection to the standard deviation of the Bayesian PDF of climate sensitivity given our current state of knowledge. He could argue, therefore, that the figures on climate sensitivity we, as a scientific community, are reporting do not give the information necessary for policymakers to make decisions that are informed by scientific knowledge and so climate models are not fulfilling their primary reason for existence.

Clearly, in order to respond to these arguments, there is a need for much more rigorous techniques for creating models and parameterisations so that the sceptic can be convinced by climate model results and will finally go away. Those involved in the development of computer models are aware of the problem of parameterisation but the development of better parameterisations has been slow. Many models used today rely on parameterisations that were developed 30 to 40 years ago, not because these parameterisations are so accurate they needn't be changed but because the development of better parameterisations has been hampered by the sheer difficulty of the problem. In recent times practitioners have come to talk of the 'parameterisation deadlock' (Randall et.al., 2003): a perceived state of stagnation in the development of better parameterisations. As Khun (1962) has pointed out, this stagnation of disciplines has occurred repeatedly throughout the history of science, and the end of each stagnation period is marked by a paradigm shift or revolution.

In the remainder of this document I present a means of escape from the parameterisation deadlock. My approach is very different from previous attempts so I'll

present a new theoretical framework with which to think about modelling. I'll describe how this new framework has led to the development of a novel piece of software that we call 'iGen' which automatically generates models and parameterisations. Finally I'll describe how iGen has been applied to the previously unsolved problem of the parameterisation of cloud-top entrainment in marine stratocumulus.

Our first step on this journey will be to precisely define the "problem of parameterisation".

1.1 Posing the question

1.1.1 The conventional view of parameterisation

Modellers think of the physics of a computer model as being split into "large-scale", "resolved" processes and "small-scale", "unresolved" or "sub-grid" processes. To understand this distinction, consider the way the state of a physical system is represented in a computer. Certain physical processes, the unresolved ones, will occur on a scale much smaller than can be captured by the computer representation, while other processes, the resolved ones, will occur on a scale large enough to be well represented in the computer. Some unresolved processes can be left out of the model without affecting the veracity of the simulation. However, other unresolved processes have knock-on effects on the larger scale, and it is these effects that must be modelled in some way by the parameterisations.

Although these unresolved processes can often be modelled with some accuracy by a high resolution model, this is practical only if computer time is not an issue and the "small-scale" state of the system is known. However, a parameterisation must use relatively little computer time and only has access to the large-scale state of the system contained within the computer's representation. This leads us to the traditional definition of the problem of parameterisation: How do we quickly and accurately calculate the large-scale effects of unresolved processes, knowing only the large-scale state of the system?

At first glance this looks like a well defined problem, but upon closer inspection it can be seen that it leaves the problem rather ill defined. For a start,

how exactly should small and large scale processes be distinguished? For example, if a 99km cumulus cloud straddles two 100km gridboxes, is this sub-grid or large-scale? This is more than just pedantry: many (probably all) GCMs in use today contain multiple parameterisations that make this distinction in mutually contradictory ways, leading to model errors (see theorem 1.2.1). How do we define a "process"? It is clear that in any volume of atmosphere there is a great deal of interaction between different sub-grid processes. What properties must a process have in order to make it meaningfully separable from the other processes with which it interacts? Inattention to this detail has led, for example, to the problem of cloud-overlap in GCMs (Collins, 2001). What should a parameterisation do when the parameterised value is under-determined by the large scale state? It could be, and often is, the case that the variable to be parameterised cannot be precisely determined from the information available in the large-scale variables. Recent parameterisations of convection, for example, have been taking a stochastic approach (e.g. Buizza et.al. 1999), but this begs the question "what is a stochastic parameterisation, and how does it differ from a non-stochastic parameterisation?" Most importantly of all, however, this definition does not give us any idea of how a model that uses a parameterisation can be used to justify the beliefs that it entails. These are just a few of the many questions that ought to be answered, or dissolved, if we are to properly define the problem.

1.1.2 Re-posing the question

Since the conventional view of parameterisation is at best incomplete, we would like to make a new definition that is complete while ensuring that uncertainty and error in the parameterised model can be quantified. It is a truism that a good parameterisation is one that, when used in a particular model, makes a fast, accurate model. It is also generally appreciated that while a given parameterisation may give accurate results in one model, it may give less accurate results in another. So the accuracy of a parameterisation cannot be quantified in isolation from its host model, it must be considered in the context of the model in which it resides if we are to quantify the error it introduces. It is towards the model, then, that we first look to see how a parameterisation should be designed.

Our first step, perhaps surprisingly, is to throw out the differential calculus. We choose not to express the equations of motion of a dynamical system as a set of differential equations, but rather as a computer program considered in a special way: Using existing numerical techniques we can easily write a high resolution, 'resolve-everything' model whose output tends to a solution of any set of equations of motion in the limit of infinite processor time and memory. To formalise this limit, consider any procedural computer programming language and add a variable, δ, which has an arbitrarily small value. δ can be used in the program to calculate, for example, grid spacing or time step. In the limit of infinite computer resources and precision, the output of the resolve-everything model tends to a solution to the differential equations as δ tends to zero.

So, any dynamical system can be trivially expressed as a computer program, in the limit, given its equations of motion. From now on, then, we will consider a computer program, let's call it f, to be the dynamical system of interest, as given to us by our physical understanding. This formalism ensures that problems are well defined by imposing 'distinguished limits' (Klein, 2008) and will be particularly convenient for our development when we get to chapter 3.

However, f doesn't immediately tell us what we want to know. As already mentioned, we would like to know about the probabilities of making particular observations of the climate system. Let's say we write a program, o, that simulates the observation; that is, it calculates the result of making the observation for a given state of the system. So, if the start state of the system is ψ, then the value of a future observation would be $o\,(f(\psi))$. However, we don't know the start state of the system. Our knowledge of the current state of the climate system will generally consist of certain measurements from satellites, balloons etc. but no matter how many measurements we make, it is quite impossible to pin down the start state to a single state that we can feed directly into f; we simply cannot get at every detail of every turbulent eddy, for example. Instead, we must settle for a large-scale, coarse description which could be satisfied by a number of detailed, small-scale states. So, to calculate the probability of making observation o, we'd have to do a Monte-Carlo simulation on $o\,(f(\psi_n))$ for each of these small-scale states, $\psi_0 ... \psi_N$. Collating the results of this Monte-Carlo simulation gives us the probability distributions we're after. This stochastic formulation of the problem of climate modelling can be traced back at least as far as Epstein (1969). Since any probability distribution can be described as

a set of moments we can consider the information we're after to be contained within a set of moments of the observable. In addition, since we can consider a square, or any other power, of an observable to also be an observable we will, without loss of generality, consider the information we're after to be contained within the first moments of a number of observables (or of a single observable, if we allow an observable to be a vector). If we write another computer program, μ, that performs a Monte-Carlo simulation on a model for a set of input states, and returns the first moment of the output then the information we're after could be calculated by $\mu(o(f), B)$, where B is our knowledge of the start state of the system.

This is a skeptic-proof model. If we had a computer to run it on, we could say of any climate projections it made: "well, if you believe in the equations of motion and in the measurements of the boundary conditions, and you want to be consistent in your beliefs, then you must also believe in the model's projection". So the output of this model defines the value we'd like to compute. In practice, as δ gets smaller, memory requirements increase, execution time increases and the model would execute far too slowly to be of any use.

Hasselmann (1976) was perhaps the first to suggest a way of formally simplifying this model by separating "weather disturbances" from climate variables using the Fokker-Planck equation, this was later developed by Arnold (2001). However, this technique has not yet led to a practical way of calculating the required observables. Another strategy is to use Bayes linear analysis. In this approach μ is simplified so that the full Bayesian probability is not computed, but a simpler function is computed instead and an argument given that this simpler function is a good model of our knowledge of the probability. This is the approach taken by Goldstein and Rougier (2009) and Oakley and O'Hagan (2002) for example. The problem with this approach is that Bayes linear inference is not as good a model of inference as the full Bayesian inference. The hard-line skeptic could argue that this type of inference is not good enough.

Our approach here is to retain the full Bayesian inference as our reference program and find an alternative program that executes much faster, but can be proven to have bounded error with respect to this resolve-everything model. After being shown the proof that the error is bounded, the skeptic should believe in the output of this faster model, to within the bounds on error. So the problem of parameterisation ultimately reduces to the problem of how to find

this faster model, and how to prove that its error is bounded. This problem is set down more formally in Definition 1.2.1.

1.2 Mathematical addenda

Theorem 1.2.1. *Parameterisations commonly used in models imply contradictory interpretations of the distinction between 'large' and 'small' scale, and this leads to model error.*

Proof. Consider, for example, the modelling of a conserved quantity, q, according to some velocity field v according to:

$$\frac{dq}{dt} = -\nabla \cdot (vq) \ .$$

Turbulence parameterisations depend on the splitting of v and q into a large scale field (denoted by an over-bar) and a perturbation (denoted by a prime) such that

$$v = \bar{v} + v'$$

and

$$q = \bar{q} + q'$$

substituting this into the conservation equation gives

$$\frac{d(\bar{q} + q')}{dt} = -\nabla \cdot ((\bar{v} + v')(\bar{q} + q')) \ . \tag{1.1}$$

Let the over-bar operation be Reynolds averaging:

$$\bar{F}(x) = \frac{1}{v} \int_{x - \frac{\Delta x}{2}}^{x + \frac{\Delta x}{2}} F(x')dx'$$

where x, x' and Δx are vectors, the integration is understood to be over the hypercube centred on x, and v is the volume of the hypercube. If we now Reynolds average equation 1.1 and remove zero terms, we are left with:

$$\frac{d\bar{q}}{dt} = -\nabla \cdot (\bar{v}\bar{q}) - \nabla \cdot \overline{v'q'} \ . \tag{1.2}$$

It is very common practice to interpret the first term in this equation as the large scale advection term, and the second term as the small-scale, turbulent flux of q.

Accordingly, the advection scheme calculates the first term and the turbulence parameterisation calculates the second term from the large-scale fields.

However, if we let $\bar{F}^i(x)$ be the field F after Reynolds averaging in all directions except x_i, then the Reynolds averaging operation can be expressed in the form

$$\bar{F}(x) = \int_{x_i - \frac{\Delta x_i}{2}}^{x_i + \frac{\Delta x_i}{2}} \bar{F}^i(x_i') dx_i' \ .$$

The large-scale advection term in equation 1.2 can now be expressed as

$$\begin{aligned} \frac{d\bar{F}(x)}{dx_i} &= \frac{d}{dx_i} \left(\int_{x_i - \frac{\Delta x_i}{2}}^{x_i + \frac{\Delta x_i}{2}} \bar{F}^i(x_i') dx_i' \right) \\ &= \bar{F}^i \left(x_i + \frac{\Delta x_i}{2} \right) - \bar{F}^i \left(x_i - \frac{\Delta x_i}{2} \right) \end{aligned} \tag{1.3}$$

where $x = \bar{v}\bar{q}$ and summation over i is implied. However, in a computer model, the large-scale fields, $\bar{q}$ and $\bar{v}$, are explicitly represented as gridded data or spectrally as the lowest n modes of some expansion. Numerical advection schemes use finite differences or analytic differentiation of the spectral modes to calculate the advection term in equation 1.2. However, this type of differentiation relies on the assumption that the field does not contain frequencies above the Nyquist frequency of the grid, or above the highest mode in the spectral expansion. Reynolds averaging does not conform to this assumption. In the case of differentiation by finite differences, for example, the differential can be expressed as

$$\frac{d\bar{F}(x)}{dx_i} = \bar{F} \left(x + \frac{\Delta x_i}{2} \right) - \bar{F} \left(x - \frac{\Delta x_i}{2} \right)$$

subtracting equation 1.3 shows that the error in the finite difference advection is given by

$$\epsilon = \bar{F} \left(x + \frac{\Delta x_i}{2} \right) - \bar{F}^i \left(x_i + \frac{\Delta x_i}{2} \right) - \bar{F} \left(x - \frac{\Delta x_i}{2} \right) + \bar{F}^i \left(x_i - \frac{\Delta x_i}{2} \right)$$

there is no reason to suppose that this error is small.

In effect, the advection scheme assumes that the large and small scales are defined as the frequencies below and above the Nyquist frequency respectively, while the turbulence scheme assumes the large and small scales are defined as the Reynolds averaged field and the perturbation[2]. If we let $L(x)$ denote a low-pass brick wall filter on a field x with cutoff at the Nyquist frequency, and

[2]This problem was also noted, in a different form, by Arakawa (2004) and Palmer (2006)

$H(x)$ be the corresponding high-pass filter then the error introduced by these contradictory assumptions can be expressed as

$$\epsilon = \nabla \cdot ((L(\bar{v}) + H(\bar{v}))(L(\bar{q}) + H(\bar{q}))) - \nabla \cdot (L(\bar{v})L(\bar{q}))$$

$$= \nabla \cdot ((L(\bar{v})H(\bar{q}) + H(\bar{v})L(\bar{q}) + H(\bar{v})H(\bar{q})) \ .$$

As long as the exact solution contains frequencies just above the Nyquist frequency, there is no guarantee that this error is small. □

Appeal is sometimes made to a 'spectral gap' at the mesoscale which would imply that ϵ could be made small by careful choice of grid spacing. However, the existence of any such spectral gap has been fairly robustly refuted (e.g. Nastrom 1984, Nastrom and Gage, 1985).

As a corollary, results reported by Koshyk and Hamilton (2001) show that the simulated spectral density of kinetic energy in the Geophysical Fluid Dynamics Laboratory global model becomes unrealistically high as the Nyquist frequency is approached, showing systematic errors in the dissipation by the model physics, as would be expected in light of this proof.

It would not be easy to solve this problem by substituting a turbulence parameterisation that made the same assumptions as the advection scheme since if we pass $\nabla \cdot ((\bar{v} + v')(\bar{q} + q'))$ through a low pass filter, we do not get a clean separation of small and large scales, as we do with Reynolds averaging. None of the terms reduce to zero and we are left with $\epsilon = L(\nabla \cdot (v'q') + \nabla \cdot (\bar{v}q') + \nabla \cdot (v'\bar{q}))$. Worse than this, with a low pass filter we lose locality; an unresolved feature in one gridbox can instantaneously affect the filtered field in another gridbox (This is due to the non-local nature of filtering, rather than any physical transfer of energy). The physical interpretation of this formal non-locality of parameterisation is explored in Palmer (2001).

Theorem 1.2.2. *Given two Turing machines, one (representing reality) runs program R and the other (the model) runs program M. We observe that both machines output d as the first n bits of their output. In order for us to have any confidence in the assertion that R and M compute the same function, n must be of the same order as the Kolmogorov complexity of R.*

Proof. Without loss of generality we consider computer programs executed by a prefix Turing machine (see, for example, Floyd and Biegel, 1994 for a definition

of Turing machines). Since we know nothing about the states of the program tapes R and M, by the principle of insufficient reason we assign equal probabilities to the possible states of the tapes. The probability that a tape will begin with program p, then, is simply $2^{-l(p)}$, where $l(p)$ is the length, in bits, of program p. The probability that a machine will have an output that begins with d, then, is given by

$$m(x) = \sum_{\{p:\pi(T(p),l(d))=d\}} 2^{-l(p)}$$

where $T(p)$ is the output of a prefix Turing machine on program p and $\pi(x, n)$ returns the first n bits of x.

From Bayes's theorem, the probability that $T(R) = T(M)$, conditioned on evidence d is given by

$$P\left(T(R) = T(M)|d\right) = \frac{m\left(T(R)\right)}{m(d)} \ .$$

It is clear that

$$m(d) > 2^{(-l(d)-O(1))}$$

since we can always write a program that simply copies d from the program tape. The coding theorem (see Li and Vitanyi, 1997, p.253) shows that

$$m\left(T(R)\right) < 2^{(-K(T(R))-O(1))} \ .$$

Where $K(x)$ is the Kolmogorov complexity of x. So

$$P(T(R) = T(M)|d) < \frac{2^{(-K(T(R))-O(1))}}{2^{(-l(d)-O(1))}} \ .$$

This means that, as long as $l(d)$ is smaller than the Kolmogorov complexity of $T(R)$, the hypothesis that $T(M) = T(R)$ is not well confirmed by the evidence d. □

This is a short, relatively simple proof but contains many subtleties that take some time to get to grips with. Anyone with an interest in the foundations of human knowledge is encouraged to spend some time truly understanding this proof, as the investment will be repaid. The proof makes use of the intriguing concept of Kolmogorov complexity (see, for example, Li and Vitanyi, 1997) and has profound epistemological implications, effectively giving an answer to the problem of induction (see, for example, Dancy, 1985) and laying down a

foundation on which the Scientific method can be built (see, for example, Mayo, 1996 for a Bayesian approach to scientific knowledge or Chalmers, 1999, for a light, general introduction to the problem of scientific knowledge). Solomonoff (2008) has recently presented a similar approach to inductive reasoning as we have presented here.

Definition 1.2.1. ***The problem posed***

We suppose that we are given a set of equations of motion of the climate system describing the orbit of a vector Ω*:*

$$\frac{\partial \Omega}{\partial t} = D(\Omega) \tag{1.4}$$

for all t*, subject to*

$$F(\Omega) = \emptyset \tag{1.5}$$

where D *is a function that includes differential operators in space,* F *represents satisfaction of any conservation laws or generalised observations, and* $\emptyset$ *is the zero vector. These equations may contain stochastic terms which represent any incompleteness of our physical understanding.*

Let $\xi(\Omega, \delta, t, s)$ *be a computer program where* Ω *is a state of the climate system,* s *is a seed for a pseudo-random number generator and the result of executing* ξ*,* T_ξ*, is the integration of* Ω *over* t *simulated seconds, such that if we let* $\Omega(t) = \lim_{\delta \to 0}(T_{\xi(\Omega_0, \delta, t, s)})$ *then* $\Omega(t)$ *satisfies equations 1.4 and 1.5 for all* $t > 0$*. (Given equations 1.4 and 1.5 it is possible to write a program* ξ *using standard numerical techniques and a 'resolve everything' strategy.)*

Let $o(\Omega)$ *be the result of an observation of the climate system in state* Ω *and let* B *be a multi-set of climate system states which represents our knowledge of the boundary conditions. Our aim is to calculate the* n^{th} *moment*

$$\mu_n(B, t) = \sum_{\Omega_0 \in B, s \in S} \frac{o\left(\lim_{\delta \to 0}(T_{\xi(\Omega_0, \delta, t, s)})\right)^n}{|B||S|} \ .$$

Without loss of generality we can consider only the first moment, $n = 1$*, since higher moments can be calculated from the first moment of a modified observable* $o'(x) = o(x)^n$*. Clearly, it is possible to write a program to calculate* μ_1 *by executing* ξ *once for each member of* B *and summing the results. Call this*

program ϕ. If we define a metric on functions $|T_\phi - T_\psi|$ and an upper limit ϵ, then we can define the set of programs

$$\Psi = \{\psi | \epsilon > |T_\phi - T_\psi|\}$$

and the problem is to find a member of Ψ that executes much faster than ϕ.

Chapter 2

Splitting models into modules

In the last chapter we defined our ultimate aim to be that of finding a good, low-resolution approximation of a high resolution model and we showed how to construct the high resolution model. We now look at how the construction of the low-resolution model can be made easier by splitting it into modules or sub-tasks in various ways, thus reducing a very hard problem into a number of easier problems. For example, we will probably construct the model so that a simulation consists of multiple, sequential executions of a single timestep function. Also, within each timestep we may, for example, have one module simulating the sea and another simulating the atmosphere. The atmospheric model may in turn be split into modules dealing separately with convection, chemistry, advection etc.

If we are to use these modularisation strategies, however, we must ensure that the errors and uncertainties of each module are correctly accounted for in the final output. Our strategy for doing this is as follows: The user decides on a way of splitting the model into modules by defining the physical meaning of each module. From this we derive a set of ideal modules that minimise the error in the final result over the expected lifetime of the model. Note that in general there will not be a way of making the error zero since the modularisation itself can be a source of error. We term this the "intrinsic" error. In addition to this error, each module is required to report a bound on its error compared to the

ideal module. This error will come from whatever approximations that have been used to increase the speed of execution of the module so we call this the "approximation error".

To calculate the moments of the outputs of the whole model, the output of each module should be fed into a 'supervisor' program that combines the intrinsic and approximation errors into a specification that describes the probability distribution of the output. The supervisor program then chooses a state at random from this distribution and uses it as output of the module. The whole model should be executed multiple times in a Monte-Carlo simulation, the results of which will give us probabilistic information about our knowledge of the moments of the observables.

We now deconstruct a typical real world model to see which modularisation strategies are commonly used and show how to derive the ideal module and what intrinsic error each modularisation strategy introduces. Surprisingly, there is no formal theory of how this can be done. However, the mathematical concepts described in Lasota and Mackey (1985) and those forming the subject of Random dynamical systems (e.g. Bhattacharya and Majumdar (2007) and Arnold (2002)) give a good start towards this. I have modified these concepts to apply to our current problem and taken some ideas from Cousot and Cousot (1977) to create a new theory which I call abstraction theory which can be used to reason about high and low resolution models. This theory is formally presented in section 2.3 but in the following section we explain the main results of the theory in less formal language.

2.1 Abstraction theory: An informal introduction

The most obvious difference between our resolve-everything model and a real world model is that the real world model will have a much lower resolution. This loss of resolution may be a source of error, so we must clarify how error is generated by loss of resolution. Before we can begin to compare the output of a low-res model to that of a high-res model, however, we must have some way to relate high-res states to low-res states. As we have seen, the input and output of a high-res model, f, consists of a description of every minute detail of the

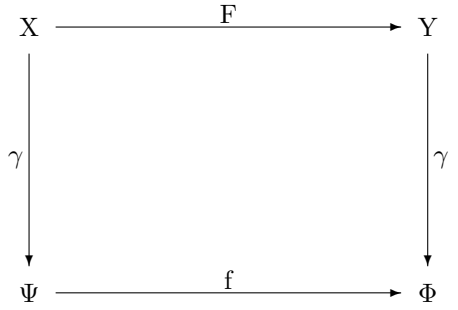

Figure 2.1: Graphical representation of the relation between a low resolution model, F, and a high resolution model f. Ψ and Φ are high resolution states, while X and Y are low resolution states. γ is the extra program that converts low to high resolution states.

state of the system. A low resolution model, F, on the other hand, would only require the large scale details of the system, neglecting the small scale detail. We compare these different resolutions by considering an extra computer program, γ, that takes a low-res state as input and turns it into a high-res state (definition 2.3.3). This requires a filling in of the small scale details using a random number generator. This program represents the conditional probability of finding the system in some high-res state, given that we know that it's in some low-res state.

If we now attach one copy of γ to the input of the high-res model and another copy to the output of the low-res model (giving $f(\gamma(X))$ and $\gamma(F(X))$ which, for simplicity, we write $f\gamma$ and γF) we have two stochastic programs, both of which take a low-res input and return a high-res output (see figure 2.1). It is evident that if γF has the same output probability distribution as $f\gamma$ for all low-res inputs (i.e. $\gamma F = f\gamma$), then the two models will always compute the same moments of any observable, as long as our knowledge of the boundary conditions can be expressed in terms of the low-res states. What's more, if γF is equal to $f\gamma$ for a single timestep then they will remain equal for any number of timesteps (theorem 2.3.1). So, if we can prove that $\gamma F = f\gamma$ over one timestep of some low-res model F then we have proved that F is error free, despite its low resolution.

Unfortunately, if we are to take the prognostic variables of a GCM as our low-res state, then no low-res model exists that has this property (theorem 2.3.6). However, for any size of low-res state, there *is* a set of prognostic variables for which there does exist a low-res model that has this property (Theorem 2.3.7). This can and should be interpreted as a call for a new set of prognostic variables for GCMs, this is left as an exercise for the reader.

In the meantime, however, we must find a way of dealing with GCM prognostic variables. We proceed by perturbing the high-resolution system so that there does exist an exact low-resolution model for the variables we're interested in. The perturbation is chosen very carefully so as to ensure that the error in any observable of the resulting low-res model (compared to the un-perturbed high-res model) is zero when averaged over the lifetime of the model (see section 2.3.4). This has the satisfying consequence that as we increase the expected lifetime of the model, the attractor of the low-res model tends towards the attractor of the un-perturbed high-res model (theorem 2.3.10).

This low-res model can be written in the following way: Begin by writing a program that converts high-res states into low-res ones (i.e. the opposite of γ) call it α. This is generally quite easy as it simply involves throwing away some of the information in the high-res state. For example, we may average over some volume or spectrally filter some field. The other converter program, γ, can be constructed from α. We show in section 2.3.5 how this should be done but essentially for any low-res input, γ should output a high-res output with a probability equal to the conditional probability of finding the system in this state, given that we know it's in the low-res state. Given these two programs, α and γ, we can construct a single timestep of our low-res model by taking the high-res model and "wrapping" it in these two extra programs so that γ is bolted on to its input and α to its output (i.e. $\alpha f \gamma$). So, when designing a low-res model we would like to make its timestep function a good approximation of $\alpha f \gamma$.

It may at first seem a bit pointless to construct a low-res model that requires the execution of a high-res model but, superparameterisation aside, we aren't proposing that this program should be used in a GCM. Instead, it serves as a constructive definition of what a low-res model should be doing; a reference against which we can measure our approximations and parameterisations.

2.2 Splitting programs into modules

We've now reduced the problem to the consideration of a single timestep of a low-res model, but the timestep of a model will itself consist of many subprocesses and modules. A parameterisation scheme, for example, will generally run alongside other parameterisation schemes, it may only update some of the output variables, or may not have any direct contact with the output, instead sending some kind of flux to another module, for example. In addition, a parameterisation scheme may be executed many times in one timestep, perhaps once for each gridbox. All this modularisation may also be a source of error which must be quantified. Once this is done, we can go on to calculate what function each module should be trying to calculate in order to minimise error in the timestep function, and so ultimately in the whole model.

We begin by considering the splitting of the timestep into two sequential operations on a state. For example, perhaps the convection scheme first calculates a convective mass flux which is added to the state, then this updated state is used to calculate, say, detrainment of convective mass flux into layer cloud. If the split has some well defined physical meaning, then these processes can be calculated by the high-res model and put together in the same way, i.e. f, can be split into two operations, g and h, such that $f = g(h)$. However, by insisting on a split we are, in effect, insisting that the model has a low-resolution representation at some point in the middle of the timestep, meaning that $f' = g(\alpha\gamma h)$. We have shown (theorem 2.3.3) that in this case if we let $G = \alpha_g g \gamma_g$ and $H = \alpha_h h \gamma_h$, the model $F = GH$ minimises error.

We next consider the splitting of the model into gridboxes, where the information available to each gridbox contains only a limited amount of local information and the timestep is calculated by separately updating each gridbox. To do this we imagine the high-res model to be split into separate volumes, each representing one gridbox. In this case, even if we minimise the error in each gridbox, the act of splitting into gridboxes may introduce further error. It turns out that extra error will result unless, in the high-res model, there is no correlation between the states of the different gridbox volumes after one timestep, given the start state of the whole system (theorem 2.3.4). However, the low-res gridbox should still calculate the function $\alpha g \gamma$ where g is the high resolution model over the volume of the gridbox in order to minimise error. The same

argument can be used to split a single gridbox into different physical schemes such as advection, precipitation, convection etc.

Finally, some parameterisation schemes make use of diagnostic variables. In this case, the high resolution model can be split into two functions $f = g(\psi, h(\psi))$ and we can minimise error, rather predictably, by letting $H = \alpha h \gamma$, $G = \alpha g \gamma$ and $F = G(X, H(X))$.

2.3 Abstraction theory: Mathematical development

We now formally develop Abstraction theory. Abstraction theory helps us reason about the links between low resolution models and their high resolution counterparts. Let p be a high resolution model and U be a two-tape universal Turing machine and let $\phi = \langle s_{t+\Delta t}, \emptyset \rangle = U_p(\langle s_t, b_t \rangle)$ be the function calculated by p, where s_t is the atmospheric state at time t and b_t are the boundary conditions at time t and $\emptyset$ is the null boundary condition. Similarly, let P be a low resolution model and $\Phi = \langle S_{t+\Delta t}, \emptyset \rangle = U_P(\langle S_t, B_t \rangle)$ be the function computed by P. We suppose, without loss of generality, that the tape of the Turing machine is arbitrarily long but finite.

Definition 2.3.1. *In the context of abstraction theory we will refer to the high resolution model as the* ***concrete model****, and the low resolution model as the* ***abstract model****.*

Definition 2.3.2. *A* ***stochastic state vector*** *of a set, X, is a vector whose dimension is equal to the cardinality of X and whose elements are real numbers in the interval $[0, 1]$. The set of all stochastic state vectors of a set X will be written $\vec{\wp}(X)$.*

Stochastic state vectors over the states of the concrete and abstract models will be referred to as ***concrete*** *and* ***abstract*** *stochastic state vectors respectively.*

If the sum of elements of a stochastic state vector of a set X is 1, it can be interpreted as a probability distribution over the members of X. Representing this as a vector, rather than a function, will turn out to be very convenient, as we shall see. In the remainder of this section we will use index notation V^x to denote the x^{th} element of the vector V, and the Einstein summation notation to

represent linear operators on vectors, for example if O is a matrix whose $(x, y)^{th}$ element is O^y_x then

$$O^y_x V^x \equiv \sum_x O^y_x V^x \ .$$

Summation is only implied over pairs of upper and lower indices, so $V^x W^x$ is not summed over. To aid notation, concrete stochastic state vectors will be named using upper-case Greek letters, and their indices will be given lower-case Greek letters. Abstract stochastic state vectors will be named using upper-case Latin letters, and their indices will be given lower-case Latin letters.

The key idea of abstraction is that the abstract model is intended to be a representation of the concrete model and each (non-stochastic) state of the abstract model must have a fixed *meaning*, in terms of the states of the concrete model. This meaning is given mathematical rigour by defining it as a probability distribution over the states of the concrete model. Once the meanings of the abstract states are defined, the relation between the behaviour of the abstract model and the concrete model follows. In full generality, this *meaning* should be given by a function from abstract stochastic state vectors to concrete stochastic state vectors.

Definition 2.3.3. *A **semantics**, γ^ψ_x is a matrix that transforms abstract stochastic state vectors to concrete stochastic state vectors. The semantics must be probability preserving, i.e. for all x*

$$\sum_\psi \gamma^\psi_x = 1$$

and each element must be in the range $[0 : 1]$.

Definition 2.3.4. *Any function, $f : \Psi \to \Phi$, defines a **power matrix**, $\wp(f) : \vec{\wp}(\Psi) \to \vec{\wp}(\Phi)$, where $\vec{\wp}(\Psi)$ and $\vec{\wp}(\Phi)$ are the sets of stochastic state vectors on Ψ and Φ respectively, such that*

$$\wp(f)^\phi_\psi = \begin{cases} 1 & \text{if } \phi = f(\psi) \\ 0 & \text{otherwise.} \end{cases}$$

This definition is extended to functions that return random variables as

$$\wp(f)^\phi_\psi = P(f(\psi) = \phi) \ .$$

So, for example, the power matrix of our concrete model, $\wp(\phi)$, would be the timestep function on concrete stochastic state vectors. In this way, the power matrix of a timestep function is the discrete equivalent of the Perron-Forbenius operator (see, e.g. Lasota and Mackey, 1985). We will refer to the power matrix of the concrete model as the stochastic concrete model.

Definition 2.3.5. *Given a stochastic concrete model f and a semantics γ, a function F is said to be an **abstraction** of f with respect to γ if and only if $\gamma_y^\phi F_x^y = f_\psi^\phi \gamma_x^\psi$.*

This definition really just embodies what we mean by *meaning*. Suppose we have a high resolution computer model f' and a low resolution model F'. If the meaning of an abstract (non-stochastic) state x is the probability distribution of concrete states given by γ_x, and F' is to have the same meaning as f', we expect the meaning of $F'(x)$ (i.e. $\gamma_{F'(x)}$) to be the probability distribution, γ_x, after it has been passed through the concrete power matrix $\wp(f')$.

Abstractions have the important property that after any number of timesteps, the meaning of their state is exactly the probability distribution that we would get if we were to run an ensemble of concrete simulations. More formally:

Theorem 2.3.1. *Given a stochastic concrete model, f, its abstraction, F, and a semantics, γ, then $\gamma F^n = f^n \gamma$ for all n, where by F^n we mean n iterations of F (i.e. matrix multiplication, n times).*

Proof. By definition

$$\gamma F = f\gamma$$

Suppose now that for some n

$$\gamma F^n = f^n \gamma \ . \tag{2.1}$$

In this case

$$\gamma F^{(n+1)} = f^n \gamma F$$

but from the definition of abstraction

$$f^n \gamma F = f^{(n+1)} \gamma$$

so

$$\gamma F^{(n+1)} = f^{(n+1)} \gamma \ .$$

From the definition of abstraction, equation 2.1 is true for $n = 1$ so by recurrence for any n

$$\gamma F^n = f^n \gamma \ .$$

□

Definition 2.3.6. *An **abstraction operator** α_ψ^x is a transform from concrete stochastic state vectors to abstract stochastic state vectors. It is the inverse of γ such that $\alpha_\psi^y \gamma_x^\psi = I_x^y$, where I_x^y is the identity matrix, all elements α_ψ^y are in the range $[0:1]$ and α is probability preserving, i.e. for all ψ*

$$\sum_y \alpha_\psi^y = 1 \ .$$

If the semantics γ is a partition of ψ (i.e. for any given ψ, γ_x^ψ is non zero for only one value of x) then there is a unique probability preserving inverse of γ. In this case we call the abstraction operator a **partitioning abstraction operator** and γ is a **partitioning semantics**.

Theorem 2.3.2. *There is a unique abstraction operator of a partitioning semantics γ, defined as*

$$\alpha_\psi^x = \begin{cases} 1 & \text{if } \gamma_x^\psi > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Proof. By definition

$$\alpha_\psi^y \gamma_x^\psi = I_x^y$$

so for any x and ϕ such that $\gamma_x^\phi \neq 0$ then $\alpha_\phi^y = 0$ if $y \neq x$ since neither α or γ have any negative elements.

From probability preservation, for any given ϕ: $\sum_y \alpha_\phi^y = 1$. so it follows that if $\gamma_x^\phi \neq 0$ then $\alpha_x^\phi = 1$, so

$$\alpha_\psi^x = \begin{cases} 1 & \text{if } \gamma_x^\psi \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

□

Given α, since

$$\gamma F = f \gamma$$

then

$$\alpha \gamma F = \alpha f \gamma$$

so, since

$$\alpha\gamma = I$$

then

$$F = \alpha f \gamma$$

and we have a constructive definition of F.

In full generality, the semantics may not be partitioning. However, we will consider only partitioning semantics here. This does not affect the theory's relevance to existing GCMs, which all have prognostic variables with partitioning semantics (i.e. given a microstate of a gridbox, its large scale state is defined). However, there is a very strong case for abandoning partitioning prognostic variables in favour of, for example, ranges of the thermodynamic quantities. This would lead to models that provide provably correct results and obviate the need for ensemble runs. This would certainly lead to more persuasive model results and may also lead to more computationally efficient models. An investigation into this is left for future work.

2.3.1 Abstraction of composite functions

A composite function is a function $f(\psi)$ that can be expressed as a composition of functions $g(h(\psi))$. In terms of operators on stochastic state vectors, composition becomes matrix multiplication $f^{\phi}_{\psi} = g^{\phi}_{\xi} h^{\xi}_{\psi}$. Composition is important because it is the basis of time-stepping.

Theorem 2.3.3. *If we have a composite function $f^{\phi}_{\psi} = g^{\phi}_{\xi} h^{\xi}_{\psi}$ and $G = \alpha g \gamma$ is an abstraction of g and $H = \alpha h \gamma$ is an abstraction of h then GH is an abstraction of f for all inputs as long as $\gamma\alpha = I$*

Proof. By substitution

$$GH = \alpha g \gamma \alpha h \gamma$$

but since $\gamma\alpha = I$ then

$$GH = \alpha g h \gamma$$

but $f = gh$ so

$$GH = \alpha f \gamma$$

□

The requirement $\gamma\alpha = I$ is not generally satisfiable. However, if we are averaging over a large number of time steps, then we have the weaker requirement that $\Psi = \alpha\gamma\Psi$ for the sum of states.

2.3.2 Abstraction of compound dynamic systems

In a GCM, each variable in the state belongs to a gridbox and each gridbox is considered as a separate subsystem which communicates with the rest of the system via its boundaries. The usefulness of this is to 'divide and conquer' the problem; it is hoped that it is easier to solve the equations of motion of the subsystems and join their respective results, than to tackle the whole problem in one go. In this section we consider this in its most general form as a splitting of a system into two parts (and by induction into any number of parts) and consider what must be assumed for this to be formally possible.

For any dynamic system, $x^{t+\Delta t} = F(x^t)$, we can arbitrarily split this system into two subsystems

$$a^{t+\Delta t} = g(a^t, b^t)$$

$$b^{t+\Delta t} = h(b^t, a^t)$$

such that $F(T(a,b)) = T(g(a,b), h(a,b))$ for some one-one mapping $x^t = T(a^t, b^t)$. a can be considered to be the state of subsystem g and b to be its boundary conditions; conversely for h.

However, this raises the question of how to construct abstractions of these subsystems so as to ensure that the whole abstract system is an abstraction of the whole concrete system. Put more formally we have a concrete system $f_{ab}^{a'b'} = g_{ab}^{a'} h_{ab}^{b'}$ and would like to construct a stochastic abstract system $G_{AB}^{A'} H_{AB}^{B'}$ which is an abstraction of f. From this, it can be seen that this is only possible if, for the abstraction of f, A' and B' are uncorrelated, given AB. This is not guaranteed, and is a property of the prognostic variables that we must ensure is fulfilled. Some thought shows that this property is not fulfilled by the prognostic variables of the current generation of GCMs. For example, gravity waves which are not captured by the prognostic variables can propagate between gridboxes and trigger convection, or the spatial distribution of convection within a gridbox can affect convection in a neighbouring gridbox by self-organisation without being reflected in the large scale variables.

Theorem 2.3.4. *Given a compound concrete model $f_{ab}^{a'b'} = g_{ab}^{a'} h_{ab}^{b'}$, a semantics γ_{AB}^{ab} and abstraction operators $\alpha_{a'}^{A'}$ and $\beta_{b'}^{B'}$, then an abstract system $G_{AB}^{A'} H_{AB}^{B'}$ is an abstraction of f if G is an abstraction of g, H is an abstraction of h and, for the abstraction of f, A' and B' are uncorrelated, given AB.*

Proof. Let

$$\kappa_{xy}^{A'} = \alpha_{a'}^{A'} \wp(g)_{xy}^{a'}$$

and

$$\lambda_{xy}^{B'} = \beta_{b'}^{B'} \wp(h)_{xy}^{b'} \ .$$

Using the notation $(c)_x = c$ for all x. Let

$$(\Delta\kappa_{xy}^{A'})_{XY} = (\kappa_{xy}^{A'})_{XY} - (\kappa_{ij}^{A'} \gamma_{XY}^{ij})_{xy} \tag{2.2}$$

and

$$(\Delta\lambda_{xy}^{A'})_{XY} = (\lambda_{xy}^{A'})_{XY} - (\lambda_{ij}^{A'} \gamma_{XY}^{ij})_{xy} \ .$$

Multiplying through by γ gives

$$(\Delta\kappa_{xy}^{A'})_{XY} \gamma_{XY}^{xy} = (\kappa_{xy}^{A'})_{XY} \gamma_{XY}^{xy} - (\kappa_{ij}^{A'} \gamma_{XY}^{ij})_{xy} \gamma_{XY}^{xy}$$

but because γ gives conditional probabilities, $(k_{XY})_{xy} \gamma_{XY}^{xy} = k_{XY}$ so from probability conservation

$$(\Delta\kappa_{xy}^{A'})_{XY} \gamma_{XY}^{xy} = \kappa_{xy}^{A'} \gamma_{XY}^{xy} - \kappa_{ij}^{A'} \gamma_{XY}^{ij} = 0$$

similarly for $\Delta\lambda$.

Let F be the stochastic abstraction of the whole concrete system

$$F_{AB}^{A'B'} = \alpha_{a'}^{A'} \beta_{b'}^{B'} \wp(g)_{ab}^{a'} \wp(h)_{ab}^{b'} \gamma_{AB}^{ab} \ .$$

Substituting equation 2.2 into F gives

$$F_{AB}^{A'B'} = \left((\kappa_{ij}^{A'} \gamma_{AB}^{ij})_{ab} + (\Delta\kappa_{ab}^{A'})_{AB} \right) \left((\lambda_{ij}^{A'} \gamma_{AB}^{ij})_{ab} + (\Delta\lambda_{ab}^{A'})_{AB} \right) \gamma_{AB}^{ab} \ .$$

Multiplying out and removing zero terms leaves

$$F_{AB}^{A'B'} = \kappa_{ij}^{A'} \gamma_{AB}^{ij} \lambda_{kl}^{B'} \gamma_{AB}^{kl} + \Delta\kappa_{xy}^{A'} \Delta\lambda_{xy}^{B'} \gamma_{AB}^{xy}$$

but by definition

$$G_{AB}^{A'} H_{AB}^{B'} = \kappa_{ij}^{A'} \gamma_{AB}^{ij} \lambda_{kl}^{B'} \gamma_{AB}^{kl}$$

the last two equations are equal when the last term in the first equation is zero, but equating this term to zero is equivalent to stating that A' and B' are uncorrelated, given AB. □

2.3.3 Abstraction of Diagnostic variables

Another technique used in GCMs is to calculate first a diagnostic variable, then calculate the new prognostic variables as a function of the diagnostic variables. This also helps to divide and conquer the problem. This amounts to splitting the system into the equations

$$x^{t+\Delta t} = g\left(x^t, h(x^t)\right)$$

we now show how this system can be abstracted.

Theorem 2.3.5. *Given a concrete model $f_x^{x'} = \wp(h)_{xg(x)}^{x'}$, a semantics γ_X^x, abstraction operators $\alpha_{x'}^{X'}$ and β_d^D, then the compound system $H_{XD}^{X'}G_X^D$ is a stochastic abstraction of f if G is a stochastic abstraction of g and $H_{XD}^{X'}$ is a stochastic abstraction of h.*

Proof. By definition

$$H_{XD}^{X'}G_X^D = \left(\alpha_{x'}^{X'}\wp(h)_{xd}^{x'}\gamma'^{xd}_{XD}\right)\left(\beta_d^D\wp(g)_y^d\gamma_X^y\right)$$

but for a given x, γ'^{xd}_{XD} must have a deterministic d equal to $g(x)$ (i.e. $\gamma'^{xd}_{XD} = \delta^d_{g(x)}\delta_e^{g(x)}\gamma'^{xe}_{XD}$) so

$$H_{XD}^{X'}G_X^D = \left(\alpha_{x'}^{X'}\wp(h)_{xg(x)}^{x'}\gamma'^{x}_{XD}\right)\left(\beta_d^D\wp(g)_y^d\gamma_X^y\right) \ .$$

The abstraction of the whole system is

$$F_X^{X'} = \alpha_{x'}^{X'}\wp(h)_{xg(x)}^{x'}\gamma_X^x$$

so the two are equal as long as

$$\gamma'^{x}_{XD}\beta_d^D\wp(g)_y^d\gamma_X^y = \gamma_X^x$$

this still allows us some freedom in the definition of γ'^{x}_{XD}, but as long as the above is satisfied, the compound system is guaranteed to be an abstraction of the whole concrete system. □

2.3.4 Dynamic systems that have no abstraction

Given a concrete function and a semantics, there is no guarantee that there exists an abstraction. That is, although it is true that if F is an abstraction

then it must be equal to $\alpha f \gamma$, the converse is not true: if $F = \alpha f \gamma$, it is not necessarily true that it is an abstraction. i.e. that $\gamma F = f \gamma$. Since this implies that

$$\gamma \alpha f \gamma = f \gamma$$

which is not necessarily true. This is the case for the prognostic variables of a GCM.

Theorem 2.3.6. *There does not exist an abstraction of physical reality for a GCM with today's prognostic variables.*

Proof. Since we have no mathematical definition of γ for GCM states, we must appeal to our intuitive understanding of the properties of γ in this case.

Suppose that there does exist an abstraction F of f such that $\gamma F = f \gamma$. Consider an abstract state X such that $X_x = 1$ for some state x, now run it backwards for some time until $\alpha f^{-n} \gamma X$ is a mixture of abstract states (this is clearly possible from our intuitive understanding of the meaning of GCM states). Now consider two of the pure states in that mixture with non-zero probability, call them A and B (where $A_a = 1$ and $B_b = 1$ for some states a and b) and run them forward through F. Clearly, $F^n A$ has a non-zero probability of being in state X so

$$(F^n A)_x \neq 0 \ .$$

Again from our intuitive understanding of the meaning of GCM states, it wouldn't be difficult to find a concrete state, ψ, in X such that $(f^n \gamma B)^\psi \neq 0$ and $(f^n \gamma A)^\psi = 0$. Since $(f^n \gamma B)^\psi \neq 0$ if F is exact, $(\gamma F^n B)^\psi \neq 0$ and since ψ is in X then $\gamma_x^\psi \neq 0$ but since $(f^n \gamma A)^\psi = 0$ then $(\gamma F^n A)^\psi = 0$, but since ψ is in X and $\gamma_x^\psi \neq 0$ then

$$(F^n A)_x = 0$$

in contradiction to our previous deduction. So F cannot be an abstraction for any semantics that conforms to our intuitive understanding of GCM prognostic variables. □

As an aside, however...

Theorem 2.3.7. *For every dynamic system on its attractor and an abstract state of any size there is a semantics for which there does exist an abstraction.*

Proof. Consider only that the attractor of a computer program is a loop of states (in the case of non-deterministic computer programs, consider the seed of a pseudo random number generator as being part of the concrete state). Let N be the number of states on this loop and ψ_n be the n^{th} state from some arbitrary starting point. If L is the number of abstract states, choose the lowest integer, I, such that $\frac{N}{I}$ is an integer not higher than L. Now let the meaning of the i^{th} abstract state, X_i, be the set of concrete states $\{\psi_n : n = jI + i, 0 \leq j < \frac{N}{I}\}$ for $0 \leq i < I$ with equal probability, any remaining abstract states mean the null vector or, to maintain probability preservation, some dummy concrete state. □

With this semantics, the timestep function is particularly simple and is given by $F(X) = X + 1$. So timestepping is easy, the problem is that it is not clear how to go from abstract states to observables. It would seem that when designing an abstract model, the choice of prognostic variables should be made carefully to trade off ease of integration against ease of calculating observables while ensuring abstractability. It would be interesting to explore the properties of the set of semantics that admit of abstractions for a given concrete function, but for now we'll call this the end of the aside.

In order to interpret the meaning of models with the usual GCM prognostic variables we consider an abstract timestep F to be an abstraction of a concrete model timestep f with a random 'turbulent noise' perturbation p. p is chosen so that F is an abstraction of pf, i.e. $\gamma F = pf\gamma$. If we now calculate some observable $o(pf)^n\gamma X = oF^nX$ then this is our best guess at the value of o, given the noise. The uncertainty introduced by the noise will be reflected in an increase of the variance of the observable. That is, if o_σ is the operator giving the variance of the observable then $o_\sigma(pf)^n\gamma X$ should be larger than $o_\sigma f^n\gamma X$.

If the problem we're investigating with our model is well conditioned, then the observable we're interested in should have a fairly well defined value given our knowledge of the boundary conditions so $o_\sigma f^n\gamma X$ should be small. We would also expect that the value of the observable is insensitive to small, 'turbulent' perturbations. If we find that $o_\sigma(pf)^n\gamma X$ is also small, then this confirms the hypothesis that the observable is insensitive to these small perturbations.

We choose p so that the average perturbation in any observable over the operational lifetime of the model is zero. So, if $\Psi_{0\ldots N}$ are the concrete states that a

model will pass through in its operational lifetime then for some observable o

$$\bar{\pi} = \sum_{n=0}^{N} o\Psi_n - op\Psi_n = 0$$

so

$$0 = o(1-p)\sum_{n=0}^{N} \Psi_n$$

if we let $\bar{\Psi} = \sum_{n=0}^{N} \Psi_n$ then the above is true for all observables, o, if

$$p\bar{\Psi} = \bar{\Psi} \tag{2.3}$$

i.e. if the average of all states is an eigenvector of p. This can be satisfied if we let

$$p = \gamma\alpha \tag{2.4}$$

where γ and α are the semantics and the abstraction operator respectively, and for a given α (which is given by the thermodynamic interpretation of the prognostic variables of the GCM), γ can be derived since, by substituting equation 2.4 into equation 2.3

$$\gamma_x^{\psi}\alpha_{\phi}^{x}\bar{\Psi}^{\phi} = \bar{\Psi}^{\psi}$$

but, from the definition of α, for any given ψ, γ_x^{ψ} is non zero only if $\alpha_{\psi}^{x} = 1$, so the sum over x consists of only one term of magnitude $\frac{\bar{\Psi}}{\alpha\bar{\Psi}}$ when $\alpha_{\psi}^{x} = 1$. So

$$\gamma_x^{\psi} = (\alpha^T)_x^{\psi}\frac{\bar{\Psi}^{\psi}}{\alpha^x\bar{\Psi}}$$

i.e. $\gamma_x^{\psi} = P(\psi|x)$, the conditional probability of a system being in state ψ given that it is in state x (and that the model is in some state in its operational lifetime). In this case, $F = \alpha pf\gamma = \alpha\gamma\alpha f\gamma = \alpha f\gamma$ and F is unchanged by the perturbation; the perturbation is entirely sub-grid and undetectable on the large scale.

This perturbation technique extends to apply to the case of composite and compound systems in the obvious way. This definition of the semantics leads to some interesting properties of the abstract function in the case when the expected lifetime of the model tends to infinite time. To see this, we need to think about the concrete and abstract models as timesteps of dynamic systems and about the attractors of these systems.

2.3.5 Concrete and abstract attractors

The Sun-Earth system is travelling along some portion of its attractor (Palmer, '99), and all climate simulations will represent a trajectory along some portion of this attractor so it is the properties of this attractor that we are interested in. For this reason we take some time now to explore the relationship between the attractors of concrete and abstract models.

In a dynamic system, the attractor is defined as the points in the phase space of the system that an orbit passes arbitrarily close to an infinite number of times as the length of the orbit tends to infinity. In the case of our computer programs there are always a finite number of states, so we can define the corresponding concept as the states that an orbit visits an infinite number of times as its length tends to infinity. Clearly, this must consist of a loop of states. The attractor of a computer program can be represented as a single stochastic state vector by defining it as the probability of finding an orbit in a given state as its length tends to infinity.

It may be argued that by restricting our attention to one-dimensional attractors we are excluding chaotic systems which have attractors with fractal dimension. However, the fractal dimension of an attractor, while of some mathematical interest, has no physical significance in the sense that for any finite set of actual or potential observations of a chaotic system, there exists an indistinguishable dynamic system with a one dimensional attractor. As a corollary to this, considering our definition of a dynamic system as a computer program in the limit that a certain variable, δ, tends to zero. For any finite set of observations there exists a finite value of δ for which the program output is identical to that in the limit.

Definition 2.3.7. *The **attractor**, $a(x)$, of a stochastic model, f, for a given start state, x, is defined as the limit of the average stochastic state of the system over a number of iterations, as the number of iterations tends to infinity. So,*

$$a(x) = \lim_{n\to\infty} \frac{\sum_{m=1}^{n} f^m x}{n} \ .$$

Theorem 2.3.8. *An attractor of a stochastic model, f, is a fixpoint of f.*

Proof. By definition an attractor, a, is given by

$$a(x) = \lim_{n\to\infty} \frac{\sum_{m=1}^{n} f^m x}{n}$$

so

$$fa(x) = \lim_{n\to\infty} \frac{\sum_{m=2}^{n+1} f^m x}{n}$$

$$= a(x) + \lim_{n\to\infty} \frac{fx}{n} - lim_{n\to\infty} \frac{f^{n+1}x}{n}$$

but both limits in the above equation are zero since all elements of fx and $f^{n+1}x$ are always fininte, so

$$fa(x) = a(x) \ .$$

□

Definition 2.3.8. *For a given abstraction operator, α, and a concrete state, Ψ, Let the* ***induced semantics*** *be a semantics that satisfies*

$$\gamma_x^\psi = (\alpha^T)_x^\psi \frac{\Psi^\psi}{\alpha_\phi^x \Psi^\phi}$$

for all x such that $\alpha_\phi^x \Psi^\phi > 0$, where α^T is the transpose of α.

The induced semantics has a very intuitive interpretation: If we suppose that Ψ is the prior probability that the concrete system is in some state, then the induced semantics is $P(\psi|x)$, the Bayesian probability that the concrete system is in state ψ, given that it is in the abstract state x.

Theorem 2.3.9. *For a given abstraction operator, α, and a concrete state, Ψ, if γ is the semantics induced by Ψ and α then*

$$\gamma\alpha\Psi = \Psi \ .$$

Proof. By substitution

$$\gamma\alpha\Psi = (\alpha^T)_x^\psi \frac{\Psi^\psi}{\alpha_\phi^x \Psi^\phi} \alpha_x i^x \Psi^x i$$

$$= \sum_x (\alpha^T)_x^\phi \Psi^\psi$$

but, from the definition of the abstraction operator, the sum over x results in the unit vector, so

$$\gamma\alpha\Psi = \Psi \ .$$

□

Theorem 2.3.10. *For a given concrete function f with a fixpoint Ψ and an abstraction operator α, if $F = \alpha f \gamma$, where γ is the semantics induced by α and Ψ, then $\alpha\Psi$ is a fixpoint of F.*

Proof. Given that $F = \alpha f \gamma$.

$$F\alpha\Psi = \alpha f \gamma \alpha \Psi$$

but, from theorem 2.3.9 $\gamma\alpha\Psi = \Psi$, so

$$F\alpha\Psi = \alpha f \Psi$$

but, by definition, Ψ is a fixpoint of f so

$$F\alpha\Psi = \alpha\Psi \ .$$

□

So, an abstract function defined by $\alpha f \gamma$ will have the same attractor as f.

Theorem 2.3.11. *For a set of temporally contiguous states $\Phi_1 ... \Phi_N$ and a concrete function f, then an abstract function $F = \alpha f \gamma$ will have zero error in the first moment of any observable when averaged over the states $\Phi_1 ... \Phi_N$ as N goes to infinity if γ is the semantics induced by α and $\Psi = \frac{\sum_n \Phi_n}{N}$.*

Proof. For a given state, Φ, the error in an observable, o between f and F after t timesteps is given by

$$\epsilon = o\gamma F^t \alpha\Phi - o f^t \Phi \ .$$

When averaged over a number of states $\Phi_1 ... \Phi_N$, the average error is given by

$$\epsilon = \frac{1}{N} \sum_{n=1}^{N} (o\gamma F^t \alpha \Phi_n - o f^t \Phi_n) = o\gamma F^t \alpha\Psi - o f^t \Psi$$

where

$$\Psi = \frac{\sum_n \Phi_n}{N}$$

but as N tends to infinity, Ψ tends to the attractor of f so Ψ becomes a fixpoint of f. Also, from theorem 2.3.10 $\alpha\Psi$ becomes a fixpoint of F so

$$\epsilon = o\gamma\alpha\Psi - o\Psi$$

but from theorem 2.3.9 $\gamma\alpha\Psi = \Psi$ so

$$\epsilon = o\Psi - o\Psi = 0 \ .$$

□

2.4 Related work

The theory of abstraction presented here is new. However, the transformation of probability distributions over the phase space of a dynamical system by a timestep function has been considered before. It is useful to distinguish two types of timestep functions on probability distributions: deterministic and random. Deterministic transforms have the property that a delta function is always transformed to another delta function, and so there is an underlying deterministic timestep function in the phase space of the system. In this case the probability transform is known as the *Frobenius-Perron operator*. Lasota and Mackey (1985) present many very pleasing mathematical results for such systems; however, most of these are of very little practical use for the present project. Random transforms, on the other hand, do not transform delta functions to delta functions and so their underlying timestep operator in the phase space of the system must have some random element (as is generally the case for the semantics presented here). This case forms the subject of *random dynamical systems*, expositions of which can be found in Bhattacharya and Majumdar (2007) and Arnold (2002).

A stochastic approach to the dynamics of the atmosphere can be traced back at least as far as Epstein (1969) who realised that this is the proper way to treat uncertainty in the start state of the atmosphere due to paucity of observation. However, he did not pursue this very far, deeming it too difficult and computationally expensive. The first stochastic treatment of the global climate is given in Hasselmann (1976) who supposed that the prognostic variables of a global model can be partitioned into two sets: the 'climate' variables and the 'weather' variables such that the time scale of the weather variables is much smaller than that of the climate variables, this was later developed by Arnold (2001). The stochastic approach to parameterisation has attracted much interest recently, especially in the parameterisation of convection. Buizza et.al. (1999) present some quite provocative results; see Shutts and Palmer (2007) and references therein for more.

2.5 Conclusion

In this chapter we have shown how to correctly modularise a low-res model. We have developed a theory of abstraction and used it to show that the timestep of a low-res model should compute the function $F = \alpha f \gamma$, where f is the timestep of a high-res model. This ensures that the meaning of the attractor of F is the attractor of f. We have shown how to calculate the error introduced by modularising a model and how to minimise the error in each module. Finally we have seen how the semantics of each module relates to the semantics of the whole model.

Given a concrete model, f, it is easy enough to write a computer program that computes its abstraction $F = \alpha f \gamma$. Simply take f and write a wrapper that transforms its inputs, stochastically, via a simulation of γ and transforms its outputs via a simulation of α. A Monte Carlo simulation of F would then reproduce the PDF of outputs for any given input. However, if we did this we would end up with a low-resolution model that runs slower than the high-resolution model. So now we have identified ideal computer programs for each part of a GCM, our next step should be to develop a technique of identifying computer programs that approximate these ideal programs and execute as fast as possible. This will be the subject of the next chapter.

Chapter 3

Approximation of computer programs

We have now reduced the problem of simulating climate to that of making a number of modules that each approximate a high resolution code of the form $\alpha f \gamma$ where f is a detailed, high resolution model and α and γ form a 'wrapper' that ensures that the input and output of the module are of the appropriate, low resolution form. In this section, we present a method of transforming $\alpha f \gamma$ into a much faster program, call it F, that approximates $\alpha f \gamma$ with some provable bound on the error. So, $F = \alpha f \gamma + \epsilon$ where $N(\epsilon) < B$ for some bound B and some measure of error N.

3.1 Static analysis of computer programs

The basic idea behind our approach is to treat a computer program as a mathematical object to be analysed rather than a code to be executed. When a program is executed, its input variables are assigned particular values and the program describes a procedure for manipulating these values to produce a set of output values. In contrast, when a program is analysed, it is the structure of the program (i.e. its syntax) that is of interest while the values of the input variables remain unspecified. So, the computer program becomes the object of analysis and the analysis is an attempt to derive properties of the program that

hold independently of the input values.

To approximate a program, P, we begin by considering it as a function $Y = U_P(X)$ from a vector of input variables, X, to a vector of output variables, Y, so that each element of X or Y represents one input or output variable respectively. We then show that for each element Y_n of Y (i.e. for each output variable of the program P) there exist polynomials $Q(X)$ and $R(X)$ such that

$$Y_n = \frac{Q(X)}{R(X)} \ .$$

We call a function of this form a 'polynomial rational'. From this, we define a vector of polynomial rationals, A, so that $A(X) = U_P(X)$ for all input vectors X. That is, for any assignment of values to the input variables, the polynomial rationals evaluate to the same values as the program's outputs, and in this way the vector of polynomials, A, is equivalent to the program, P[1]. We then find an approximation to A in such a way that the error between the approximation and the original does not exceed some user specified value over some user-specified range of input values. This approximation is then converted back into a computer program which approximates the original program.

Since the approximation is derived from the original through a sequence of well defined mathematical transformations, formal bounds on the actual error introduced by the approximation can be calculated and reported. This describes, in very rough outline, how programs can be approximated. However, there remain many details to be filled in and complications to be addressed and we will introduce and deal with these in the following sections. We begin with very simple examples and work gradually toward more complex, realistic cases.

3.1.1 Approximating arithmetic assignments

Let's begin by deriving an approximation of the following very simple program:

[1]Strictly speaking, $U_P(X)$ is a vector of floating point numbers, fixed-length integers, characters and booleans, while the evaluation of A is a vector of real numbers. We consider a real number, r, to be equal to a floating point number, f, if there does not exist another floating point number f' such that $|r - f'| < |r - f|$. Similarly for integers. Characters and booleans are treated as integers. We leave the properties of the special floating point numbers `inf`, `-inf` and `NaN` undefined in the symbolic interpretation and allow A to differ from P if any floating point operation in P returns any of these values.

```
input(x)
  a = x + 1
  y = a*a;
  y = y*y;
output(y)
```

Normally, we would simply execute this program for some input, say $x = 0.1$. So, after the first line $a = 1.1$, after the second line $y = 1.1 \times 1.1 = 1.21$, the next line $y = 1.21 \times 1.21 = 1.4641$. So the output is 1.4641.

However, when analysing the program we don't want to think of the input and output as having specific values, like 1.4641, but rather we want to express the output as a polynomial rational with the input x as an independent variable. To do this, we consider the variables of the program to be polynomials, rather than floating point numbers with specific, numeric values. Since polynomials can be added together and multiplied, our program still has a clear interpretation as a sequence of operations on polynomials. So, we begin by setting the input variable `x` to be equal to the first order polynomial $f(x) = x$. The first line sets `a` to the polynomial $a(x) = 1 + x$, the second line sets $y(x) = (1 + x)(1 + x) = 1+2x+x^2$, the next line $y(x) = (1+2x+x^2)(1+2x+x^2) = 1+4x+6x^2+4x^3+x^4$ so the output of the program can be considered to be the polynomial $y(x) = 1 + 4x + 6x^2 + 4x^3 + x^4$. If we evaluate this polynomial at $x = 0.1$, for example, we get $y(0.1) = 1 + 0.4 + 0.06 + 0.004 + 0.0001 = 1.4641$, as we would expect.

Suppose we are now told that x, the program's input, is always in the range $-0.1 \leq x \leq 0.1$ and that we can apply simplifications as long as the absolute error in the output remains bounded by 0.1. The simplest way of approximating the program in this case is by getting rid of the higher order terms in $y(x)$, so the approximation becomes $y(x) \approx 1 + 4x$ with an error given by $\epsilon(x) = 6x^2 + 4x^3 + x^4$. Within the range $-0.1 \leq x \leq 0.1$, $\epsilon(x)$ has a maximum absolute value of 0.0641 when $x = 0.1$ so $y(x) = 1 + 4x \pm 0.0641$. This converts into the simplified program:

```
input(x)
  y = 1 + 4*x
output(y)
```

A better approximation can be made by using the first order Chebyshev ap-

proximation of $y(x)$, giving $y(x) = 1.03004 + 4.03x \pm 0.0310625$.

This method can be applied to any sequence of the four arithmetic operations $+$, $-$, $\times$ and $\div$, and the result can always be expressed in the form $\frac{Q}{R}$ by using the following identities:

$$\frac{Q}{R} \times \frac{S}{T} = \frac{QS}{RT}$$
$$\frac{Q}{R} + \frac{S}{T} = \frac{QT + RS}{RT}$$
$$-\frac{Q}{R} = \frac{-Q}{R}$$

and

$$\frac{\frac{Q}{R}}{\frac{S}{T}} = \frac{QT}{SR} .$$

It should be noted that the resulting polynomials interpret the algebraic operators in a program as exact operations, rather than the approximate arithmetic of floating point numbers. So to this extent the evaluation of a polynomial may differ from the result of executing a program. This can be thought of as replacing all finite precision floating point numbers with infinite precision. However, a properly written simulation should be insensitive to increases in the precision of floating point arithmetic. In this case any differences would be dwarfed by the approximations we are likely to want to introduce, so we do not consider this to be an issue for our purposes.

3.1.2 Random numbers

In our definition of a wrapped model, $\alpha f \gamma$, we specified that γ was, in general, a stochastic transformation. That is, for any given input, it's output may take on any of a number of values with some probability distribution. When written as a program, this would be implemented by using a pseudo random number generator. Let's say the function `rand()` calls a random number generator whose output is a floating point number in the range $[-1 : 1]$ with a top-hat probability distribution. Random numbers with any arbitrary distribution can be generated from the `rand()` function by passing it through a function $f($`rand()`$)$ which is related to the inverse cumulative probability density of the required distribution. More details are given in theorem 3.3.1 in section 3.3.

Let's suppose that the input of the program in the previous section represents some measured quantity and that the measuring device can only measure x to

within ± 0.005. This can be represented by letting γ be the program `x = x + 0.005*rand()`. So the wrapped program is

```
input(x)
  x = x + 0.005*rand();
  a = x + 1
  y = a*a;
  y = y*y;
output(y)
```

Random numbers can be dealt with by representing the output of each call to `rand()` as a unique, especially tagged, variable, let's say $r_0 ... r_n$. The moments of each output can then be calculated by integrating over each of the tagged, random variables. So, analysing the program above shows the first moment of `y` to be

$$\begin{aligned}\bar{y} = \quad & \tfrac{1}{2}\int_{-1}^{1}\big[(1+4x+6x^2+4x^3+x^4)+ \\ & (0.02+0.06x+0.06x^2+0.02x^3)r_0+ \\ & (0.00015+0.0003x+0.00015x^2)r_0^2+ \\ & (5\times 10^{-07}+5\times 10^{-07}x)r_0^3+ \\ & 6.25\times 10^{-10}r_0^4\big]\ dr_0\end{aligned}$$

evaluating the integral gives

$$\bar{y} = 1.000050000125 + 4.0001x + 6.00005x^2 + 4x^3 + x^4 \ .$$

3.1.3 Fixed loops

We now look at a more complex program that performs a numerical integration of the Lorenz equations (Lorenz, 1963):

$$\frac{dX}{dt} = \sigma(Y - X)$$

$$\frac{dY}{dt} = rX - Y - XZ$$

$$\frac{dZ}{dt} = XY - bZ \ .$$

Following Lorenz, we let $\sigma = 10$ and $b = \frac{8}{3}$. Here, we imagine that r has been measured to have value r_m with an instrument that has an associated error of e.

We suppose that the measured value lies in the range $0 < r_m < 28 - e$ (Lorenz uses the value $r = 28$). The following program integrates these equations:

```
input(rm)
r = rm + e*rand()
X = 0.0
Y = 1.0
Z = 0.0
loop 6 times {
  dx_dt = 10.0*(Y-X)
  dy_dt = r*X - Y - X*Z
  dz_dt = X*Y - (8.0/3.0)*Z

  X = X + dx_dt*0.01
  Y = Y + dy_dt*0.01
  Z = Z + dz_dt*0.01
}
output(X)
```

The new element here is the loop. To deal with loops that have a fixed number of iterations, first reduce the body of the loop to a vector of polynomial rationals in the same manner as we have already done. So, if we place the variables into a vector (X, Y, Z), then the body of the loop would be equal to the vector

$$L = \begin{pmatrix} x + 0.1y - 0.1x \\ y + 0.01rx - 0.01y - 0.01xz \\ z + 0.01xy - \frac{0.08}{3}z \end{pmatrix} .$$

The loop, then, is equal to the polynomial $L^6(X, Y, Z)$ and the output, X, is just the first element of this. On performing the calculation, the value of X comes out as

$$\begin{aligned} X = \quad & -1.09964 \times 10^{-15}(r_m + ei)^3 + 5.66995 \times 10^{-7}(r_m + ei)^2 + \\ & 0.00169011(r_m + ei) + 0.455595 \end{aligned}$$

where i is the value returned by the random number generator.

Using Chebyshev approximation (under the assumption that errors up to 10^{-4} are acceptable), this can be approximated as

$$X = 0.00171(r_m + ei) + 0.45532 \pm 5.6 \times 10^{-}5 .$$

This can now be easily integrated over i to find the average value of X

$$\bar{X} = \int_{-\infty}^{\infty} \left(0.00171(r_m + ei) + 0.45532 \pm 5.6 \times 10^{-5}\right) P(i)di$$

where $P(i)$ is the probability distribution of the random number generator. So, since by definition $\bar{i} = 0$

$$\bar{X} = 0.00171rm + 0.45532 \pm 5.6 \times 10^{-5} \ .$$

This equation converts to a computer program

```
input(rm)
  X = 0.00171*r_m + 0.45532;
output(X);
```

that returns the expectation value of X for any measured value in the range $0 < r_m < 28 - e$ in 2 arithmetic operations with an error bounded by 5.6×10^{-5}. This compares to 90 operations for the original program.

An important point to note here is that in our examples so far the calculation of the polynomial has proceeded sequentially, in much the same order as it would during an execution. The loop, L^6, however, illustrates that an analysis may proceed very differently from an execution. During an *execution* of the loop, the program pointer would loop round 6 times; during an *analysis*, however, we immediately define the meaning of the loop as L^6. This can be evaluated in any way we please. For example, we may evaluate $M = L \otimes L \otimes L$, then $L^6 = M \otimes M$, giving L^6 in 3 (albeit polynomial) operations. In some cases, there exists a closed form solution for a loop L^n in terms of n. As a simple example, suppose we have a loop with 100 iterations, and the body of the loop evaluates to $L =< 2X, Y + 1 >$ for an input vector $< X, Y >$. L^n can be immediately solved as $L^n =< 2^n X, Y + n >$ giving $L^{100} =< 2^{100} X, Y + 100 >$ without the need to go through the 100 iterations.

So, when a program is executed, a program pointer moves, step by step, through the program. When a program is analysed, however, it's equivalent polynomial is built up from the structures of the program. There is no program pointer, structures can be transformed in any order, the end of the program may be transformed before the beginning.

Finding a closed form solution for a loop L^n is the same as solving a recurrence relation. Much work has been done on this, for a survey of early work see Lueker

(1980) and references therein. More recently Bachmann et.al. (1994) introduced 'chains of recurrences' as a formal tool for solving recurrences, this was later developed by van Engelen (2000, 2001, 2004) to analyse loops in programs for compiler optimisation. Pop et.al. (2005, 2006) developed this idea further into 'trees of recurrences' for the same application.

3.1.4 `if` statements

Consider the following program which roughly simulates a ball bouncing on the floor in a gravitational field:

```
input(z) {
  g  = 10.0;
  dt = 0.01;
  v  = 0.0;

  loop 100 {
    z = z + v*dt - 0.5*g*dt*dt;
    v = v - g*dt;
    if(z <= 0) {
      v = -0.8*v;
      z = 0.0;
    }
  }

  output(z)
}
```

`z` is the height of the ball and `v` is its velocity in the upward direction. The input is the initial height that the ball is dropped from and is taken to be in the range $[1:2]$.

The new structure here is the `if` statement. This is dealt with by using the Heaviside step function, defined as

$$H(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases}$$

where $H(0)$ is undefined.

The body of the `if` statement can be calculated in the usual manner, giving the polynomial vector $P = (0.0, -0.8v)$. The condition of the `if` statement can be calculated by turning the inequality into the homogeneous form $B > 0$, in this case we get $-z > 0$. Applying the Heaviside function to the left hand side of this inequality gives $H(-z)$ which is equal to 1 if the inequality is true, 0 otherwise. Multiplying the condition by -1 gives $H(z)$ which is the reverse: 0 if true, 1 otherwise. So the whole `if` statement is equivalent to

$$F = H(-z)P + H(z)I$$

where $I = (z, v)$ is the identity polynomial vector (i.e. the n^{th} element is equal to the n^{th} variable). So, if $z < 0$ then $H(-z) = 1$ and $H(z) = 0$ so $F = P$. If $z > 0$ then $H(-z) = 0$ and $H(z) = 1$ so $F = I$. This is exactly the behaviour we require for equivalence to the `if` statement.

So, the whole loop equates to the vector

$$L = \begin{pmatrix} H(z + 0.01v - 0.0005)(z + 0.01v - 0.0005) \\ (1 - 1.8H(z + 0.01v - 0.0005))\,(v - 0.1) \end{pmatrix}$$

and the whole program equates to L^{100}.

Strictly speaking, the floating point variable `z` may equal 0.0 when the `if` statement is reached, in which case F should equal P according to the `if` statement. However, in this case, the analysis, L^{100} is a function of $H(0)$ which is undefined. The correct behaviour can be restored by noting that the floating point number `z` that forms the input of the program does not denote a single real number but rather denotes the range of numbers $[z - \frac{\epsilon}{2}, z + \frac{\epsilon}{2}]$, where ϵ is the smallest increment to z representable as a floating point number. So, we can associate some uncertainty to the input value by giving all inputs an implicit uncertainty of $\pm\frac{\epsilon}{2}$. All instances of H would then be integrated over this uncertainty. The case when $z = 0$ would then be interpreted as an integral of the form

$$\int_{-\frac{\epsilon}{2}}^{\frac{\epsilon}{2}} H(x)P(x)dx$$

which is formally independent of the value of $H(0)$ (as long as P stays finite) since the width of the undefined area is infinitely thin[2].

[2]Expressed more mathematically: H is defined 'almost everywhere'.

The conditions in `if` statements may also contain conjunctions `&&` or disjunctions `||` so that `(A && B)` is true if and only if `A` and `B` are both true, and `(A || B)` is true if and only if `A` or `B` or both are true. A disjunction `(A || B)` is equivalent to $A + B - AB$ and a conjunction `(A && B)` is equivalent to AB.

Introduction of the Heaviside step function means that, technically, we can no longer represent programs as polynomial rationals. However, we consider the step function to be the limit of an infinite series of polynomial terms, and so we can still express a program as a polynomial rational, but with the implicit understanding that this is in the limit of infinite degree polynomials.

3.1.5 Conditional loops and Rice's Theorem

Conditional loops can be implemented using the structures we have already described

```
while(A) {
  ...
}
```

is equivalent to

```
loop M {
  if(A) {
    ...
  }
}
```

for some M that gives the maximum number of times the `while` loop can iterate over the domain of inputs.

It may be argued that in full generality one cannot put a finite number on M since some programs enter infinite loops and never terminate. To make matters worse it is a well known result of theoretical computer science (Turing, 1936) that one cannot always tell if a given conditional loop will end up being infinite or not. Rice's theorem (Rice, 1953) uses this result to show that it is, in fact, impossible to construct an analysing machine that can, in full generality, tell us anything non-trivial about the properties of programs. Here, a trivial property

is defined as one that is true of either all or no computer programs. This seems very negative, but we must think carefully about what Rice's theorem is telling us: It is not telling us that for a given input program we cannot prove whether it has a given property; rather, it is telling us that for any given property, there exists at least one input program for which the analysis machine will either not terminate, or will end up showing something trivial about the input program.

In addition, by insisting that M is given a finite value we are, in effect, restricting ourselves to the subset of computer programs known as 'basic recursive'. These are the programs that can be proved to terminate. As it turns out (Solomonoff, 2005), almost all computer programs in practical use happen to compute basic recursive functions. Since a program that computes a basic recursive function can be proven to terminate, Rice's theorem does not apply. These programs *can all* be proven to have the kind of properties that we are likely to want to identify[3]. Upon reflection, it is not surprising that climate models in use can be shown to terminate: they are written that way. For example, if it was even suspected that an algorithm used in a GCM could take more than, say, a month of computer time to execute one timestep on some input, this would in all practical respects be considered to be a bad algorithm and would be rewritten or thrown out of the model[4]. We therefore restrict ourselves to the consideration of the basic recursive functions without fear that this will be a restriction for our proposed application.

3.1.6 Arrays

To complete the description of our analysis technique, we present a method of dealing with arrays. This is conveniently done by representing the whole array as a single polynomial with the array's index variable as its independent variable

[3]It is the author's opinion that this is no accident. Rather, it is a side-effect of the fact that programs are written by humans to do useful things

[4]The one exception to this is the use of randomised algorithms, some of which may technically never terminate. However, these algorithms all have the property that the probability of termination very quickly approaches 1 as the number of iterations of some loop increases. So, by limiting the number of iterations we effectively take an algorithm with a vanishingly small probability of not terminating and replace it with an algorithm with a vanishingly small probability of returning the wrong answer. So when we come to integrate over the random numbers, there is always a finite M that ensures that the result is the correct answer with a vanishingly small error.

(multidimensional arrays can trivially be reduced to one dimensional arrays by using the address in memory as the index of each element). Suppose we have an array, `A`, of size N. We choose the N equidistant points on the interval [-1:1]

$$x_n = \frac{2n}{N-1} - 1$$

where n is an integer in the range $0 \leq n < N$. From this, we let the Lagrange basis polynomials be defined as

$$l_n = \prod_{0 \leq i < N, i \neq n} \frac{x - x_i}{x_n - x_i} \ .$$

These polynomials have the important property that $l_n(x_m) = 0$ if $n \neq m$ and $l_n(x_n) = 1$. If we let a_i be the value of `A[i]` for all integers $0 \leq i < N$, then the $(N-1)^{th}$ degree polynomial

$$A(x) = \sum_{i=0}^{N-1} a_i l_i$$

has the property that for any integer $0 \leq j < N$, $A(\frac{2j}{N-1} - 1) = a_i$. So an array reference `A[j]` is equivalent to the polynomial

$$A\left(\frac{2j}{N-1} - 1\right) \ .$$

If we now define the bi-variate polynomial $L(i, x)$ as

$$L(i, x) = \sum_{j=0}^{N-1} l_j \left(\frac{2i}{N-1} - 1\right) l_j(x)$$

so that $L(i, x) = l_i(x)$ for any integer $0 \leq i < N$, then the value of an array after an assignment operation `A[i] = Y` is equivalent to the polynomial

$$A + \left(Y - A\left(\frac{2i}{N-1} - 1\right)\right) L(i) \ .$$

3.1.7 Related work

The approach described in this chapter can most generally be described as belonging to the computer science discipline known as 'static analysis' which covers any analysis that treats a computer program as a mathematical object to be analysed rather than as a code to be executed. Examples of this date back as far as Moore's interval arithmetic (Moore, 1966) although a formal treatment of the idea wasn't given until 1977 (Cousot and Cousot, 1977), when it was named

'abstract interpretation' since the variables of the program, rather than being considered as numbers, are considered to be some more abstract mathematical object that represents, in some way, our knowledge of the variable's true or possible values. The main contribution of Cousot and Cousot was the formal analysis of loops in terms of lattice theory which showed a way of ensuring that all programs can be analysed in finite time (although, as a corollary to Rice's theorem, this may result in a trivial analysis).

Static analysis has now developed into a mature field (Cousot 1996, Hinchey et.al. 2008) and its techniques are used in a range of applications including optimising compilers (e.g. Aho et.al. 2007) and to validate safety critical computer systems such as those used to control passenger aircraft or nuclear power plants (e.g. Blanchet et.al. 2002).

Closest to the approach presented here is the sub-discipline of 'Symbolic analysis' (e.g. Fahringer and Scholz, 2003) which uses symbolic expressions to represent the values of variables. This has been applied to parallelising compilers (Haghighat and Polychronopoulos, 1996; Kyriakopoulos and Psarris, 2009), compiler optimisation (Van Engelen, 2001), validation of cryptographic protocols (Bracciali et.al., 2008; Modersheim and Vigano, 2009; Canetti and Herzog, 2010), detection of run-time errors (Bush et.al, 2000; Cadar et.al., 2006) and execution time analysis (Blieberger, 2002), to give a few examples.

The idea of representing the value of variables as functions, rather than numbers, was also proposed by Epstein et.al. (1982a, 1982b) who named it "ultra arithmetic". This is of particular relevance to our project as this approach was used to prove that the output of programs was approximated by some polynomial. Recently Brisebarre and Joldes (2010) have extended this technique to perform a simple, mono-variate symbolic analysis of various simple functions by Chebyshev polynomial approximation, although there was no treatment of division or reciprocation of polynomials so this does not constitute a full arithmetic. Trefethen, Battles and Platte have also developed a library of functions called "Chebfun" that allows symbolic analysis with Chebyshev polynomials (BattlesTrefethen, 2004; Trefethen, 2007; Platte and Trefethen, 2010). However, this library only allows mono-variate polynomials, does not calculate error bounds and does not implement any syntactic analysis of programs. The work presented in this and the remaining chapters develops the approach of these authors much further. The treatment of random number generators and arrays

presented here is, to the author's knowledge, new. In the remaining chapters, techniques will be described that allows programs of much greater complexity to be analysed.

3.2 A formal, symbolic semantics of computer programs

We now present a formal development of the ideas discussed in this chapter. We begin by describing how a computer program can be converted into a rational of the form $\frac{Q}{R}$ where Q and R are polynomials. The first stage in our analysis is to consider a high resolution computer program as a mathematical function. In order to turn a computer program into a mathematical function we follow the method of 'denotational semantics' (see, e.g. Mosses, 1990). However, to understand how this can be done we need first to understand syntactic analysis.

3.2.1 Formal syntactic analysis

When syntactically analysing a program, we begin by considering a computer program as a long list of characters (i.e. a string). While some strings are meaningful computer programs, others (for example "The third policeman", Flann O'Brien, 1993) are not. It is the 'syntax' of the computer language that tells us which strings of characters are computer programs in that language and which are not. It does this by relying on the observation that all languages are made up of a finite number of recurring structures. So although there are an infinite number of possible Fortran computer programs, for example, they are all made up of a finite number of simple, recognisable structures, i.e. loops, subroutines, 'if' statements etc. Similarly, the English language is made up of nouns, verb phrases, subordinate clauses, sentences etc. Every string of a language must be built up from these recurring structures. If a string contains a structure that is not in the syntax of a language, then it is not part of that language.

Every programming language has a well defined syntax which can be expressed as a set of rules called 'rewrite rules'. Each rewrite rule describes how a given type of structure can be made up of characters and sub-structures. These rules

can conveniently be written in a form called Backus Naur Form (BNF) (Backus et.al., 1963). As a simple example, let's describe the syntax of arithmetic equations on integers like $1+1=2$. Let's begin by describing the syntax of integers. This can be expressed in BNF as

Rule 1 `<digit> ::= 0|1|2|3|4|5|6|7|8|9`

Rule 2 `<integer> ::= <digit>`

Rule 3 `<integer> ::= <integer><digit>`.

Here, `<integer>` and `<digit>` are called tokens which act as temporary placeholders for the structures of the language. The first rule states that a `<digit>` token can be exchanged for any one of the characters 0,1,...,9. The second rule states that an `<integer>` token can be exchanged for a `<digit>` token. The third rule states that an `<integer>` can be exchanged for another `<integer>` followed by a `<digit>`. By repeated application of these three rules, any integer can be constructed. So, for example, the integer 56 can be constructed in the following way: Start with an `<integer>` token, use rule 3 to rewrite this as `<integer><digit>`. Rule 1 gives `<integer>6`. Rule 2 gives `<digit>6` and rule 1 gives `56`.

Let's now introduce the arithmetic operators $+$, $-$, $*$ and $/$:

Rule 4 `<operator> ::= +|-|*|/`

Rule 5 `<expression> ::= <integer>`

Rule 6 `<expression> ::= <integer><operator><expression>`.

Adding these rules to the first three, we can now see how the rules can be used to construct arithmetic expressions: Let's say we begin with an `<expression>`, using rule 6 this expands to `<integer><operator><expression>`, using rule 5 we get `<integer><operator><integer>`, rule 4 gives, for example, `<integer>+<integer>` and applying rule 1 twice gives, for example, `1+1`.

To complete the syntax we add the equate operator:

Rule 7 `<equation> ::= <expression>=<expression>`.

Using this rule we get `1+1=<expression>` then using rule 5, 2 and 1 gives `1+1=2`.

So these 7 rules can describe all possible arithmetic equations on integers, although for completeness we might want to add a rule for parentheses:

Rule 8 `<expression> ::= (<expression>)`.

This syntax can distinguish between strings that are arithmetic equations on integers and those that are not: if there exists a way of rewriting an `<equation>` token into a given string, s, then s is an arithmetic equation, otherwise it isn't. Efficient algorithms exist to check this, most notably the 'chart parsing' algorithm (Kay, 1986). However, the syntax admits, for example, `1+1=3` just as well as it does `1+1=2`. That is, syntactic rules can describe the structure of a string but say nothing about the meaning of that string.

3.2.2 Formal semantics

The meaning of a string can be derived by adding some extra information to the rewrite rules of the syntax. We introduce the notation m_T to denote the meaning of syntactic token `<T>` and for each rewrite rule in the syntax we give a corresponding semantic rule that defines the meaning of the token on the left hand side of the rule in terms of the meaning of the tokens on the right hand side. Take, for example, binary strings which can be described syntactically in a similar way to the decimal integers using the rules

Rule 1 `<binarydigit> ::= 0|1`

Rule 2 `<binarynum> ::= <binarydigit>`

Rule 3 `<binarynum> ::= <binarynum><binarydigit>`

on to these, we add the semantic rules

Rule 1 `<binarydigit> ::= 0|1`

$m_{binarydigit} = 0|1$

Rule 2 `<binarynum> ::= <binarydigit>`

$m_{binarynum} = m_{binarydigit}$

Rule 3 `<binarynum> ::= <binarynum><binarydigit>`

$m_{binarynum} = 2 * m_{binarynum} + m_{binarydigit}$

here, the meaning of a token `<T>`, m_T, is an integer number. As an example, take the binary string `11`. This can be created in the following way: Start with a `<binarynum>` token whose meaning is $m_{binarynum}$. Use rule 3 to get `<binarynum><binarydigit>`. The semantic rule in rule 3 gives the meaning as $2 * m_{binarynum} + m_{binarydigit}$. Rule 1 gives `<binarynum>1` which means $2 * m_{binarynum} + 1$. Rule 2 gives `<binarydigit>1` which means $2 * m_{binarydigit} + 1$, and rule 1 gives `11` which means $2 * 1 + 1$ which is equal to 3. So the meaning of the binary string `11` is the number 3, which is what we would expect.

There is one, perhaps subtle, but very important point to note about Rule 1. In this rule the `0|1` in the syntax part refers to the characters '`0`' and '`1`' (that is ASCII codes 48 and 49 if the string is stored in ASCII encoding) while in the semantic part the 0|1 refers to the numbers 0 and 1. So in effect the rule is saying that ASCII code 48 (the character `0`) means the number 0 and ASCII code 49 (the character `1`) means the number 1.

3.2.3 A symbolic semantics of computer programs

We now have all we need to define the semantics of a computer program. Our aim is to define the semantics so that the meaning of a program is a vector of polynomial rationals.

The syntax and semantics of a modern, high level computer programming language is very large (see, for example, ISO/IEC, 1998). However, it is common practice for compilers to be split into a 'front end' and a 'back end'. The front end translates a computer program written in a high level language (such as Fortran or C++) into an intermediate level language which has a much simpler syntax. This intermediate code is then passed to the back end to be turned into an executable for the target machine (Aho et.al., 2007). GIMPLE (Merril, 2003) is an example of such an intermediate language. So, rather than give a semantics of a high-level computer programming language, we give the semantics of a typical intermediate language.

Any computer program that analyses this intermediate language can be attached to the front end of a compiler to make it capable of analysing a high level computer language. In this way, a semantics of an intermediate level language implies a semantics for any high-level language. For simplicity, we omit the semantics of subroutines as these are taken to have the standard semantics.

The meaning of a computer program is taken to be a vector of polynomial rationals in input variables. Each element of the vector represents the value of some variable, so we can think of this as a function, $Y = F(X)$, from a vector of input variable values, X, to a vector of output variable values Y. The meaning of a `<variable>` or an `<arrayvariable>` is the vector whose elements are all zero except for the element that represents that variable, whose value is 1. The meaning of a `<boolean>` is a Heaviside step function on a polynomial. The Heaviside step function is taken to be the limit of an infinite series of polynomial functions of increasing degree. We use the following notation:

$(F)^n$ is the n^{th} power of F

$F(X)^n$ is the n^{th} element of $F(X)$

I is the identity matrix

$H(X)$ is the Heaviside step function

$\vec{1}$ is the vector whose elements are all 1

m^T is the transpose of m

N is the maximum number of times a loop is repeated in the program of interest; since we are restricting ourselves to the basic recursive functions, this is always provably finite.

Our semantics of computer programs can be expressed in the following way (we omit the semantics of arithmetic expressions, as this is dealt with in the following chapters):

1. `<program> ::= main() {<codeblock>}`
   $m_{program} = m_{codeblock}$
2. `<codeblock> ::= <codeblock><statement>`
   $m_{codeblock} = m_{statement}(m_{codeblock})$
3. `<codeblock> ::= <statement>`
   $m_{codeblock} = m_{statement}$
4. `<statement> ::= end`
   $m_{codeblock} = I$

5. `<statement> ::= loop <integer> {<codeblock>}`
   $m_{statement} = (m_{codeblock})^{m_{integer}}$

6. `<statement> ::= <variable>=<expression>`
   $m_{statement}(X) = (I - m_{variable}^T m_{variable})X + m_{expression} m_{variable}$

7. `<statement> ::= while(<boolean>) {<codeblock>}`
   $m_{statement} = (m_{boolean} m_{codeblock} + (1 - m_{boolean})I)^N$

8. `<statement> ::= if(<boolean>) {<codeblock 1>} else {<codeblock 2>}`
   $m_{statement} = m_{boolean} * m_{codeblock1} + (1 - m_{boolean}) * m_{codeblock2}$

9. `<statement> ::= <arrayvar>[<variable>] = <expression>`
   $m_{statement}(X) =$
   $X + \Big(m_{expression} - (m_{arrayvar} \cdot X)(\frac{2m_{variable} \cdot X}{D-1} - 1) \Big) L(\frac{2m_{variable} \cdot X}{D-1} - 1) m_{arayvar}$
   where
   D is the dimension of the array
   $L(i, x) = \sum_{i=0}^{N-1} l_i(x) l_i(i)$
   and
   $l_i(x)$ is the i^{th} $(D-1)^{th}$ degree Lagrange basis on the equidistant points on the interval $[-1 : 1]$

10. `<statement> ::= <integervariable> = <expression>`[5]
    $m_{statement}(X) = (I - m_{intvariable}^T m_{intvariable})X + V m_{variable}$
    where
    $V = H(m_{expression} - 2^{31})2^{31} + H(m_{expression} - H(m_{expression} - 2^{31})2^{31} - 2^{30})2^{30} + H(m_{expression} - H(m_{expression} - H(m_{expression} - 2^{31})2^{31} - 2^{30})2^{30} - 2^{29})2^{29} ...$

11. `<boolean> ::= <expression-1>'>'<expression-2>`
    $m_{boolean} = H(m_{expression-1} - m_{expression-2})$

12. `<boolean> ::= <expression-1>'<'<expression-2>`
    $m_{boolean} = H(m_{expression-2} - m_{expression-1})$

[5]Note that this treatment of integers does not account for overflow, so the output of code that depends on the overflow of integers may compute different values than the analysis. Other than in the creation of random numbers, treated separately, this is not considered to be a problem for physical simulation code.

13. `<boolean> ::= <boolean-1>&&<boolean-2>`

    $m_{boolean} = m_{boolean-1} m_{boolean-2}$

14. `<boolean> ::= <boolean-1>||<boolean-2>`

    $m_{boolean} = m_{boolean-1} + m_{boolean-2} - m_{boolean-1} m_{boolean-2}$

15. `<boolean> ::= !<boolean>`

    $m_{boolean} = (1 - m_{boolean})$.

3.3 Random numbers

We now show how a random number generator with a top hat probability distribution can be used to generate random numbers with any arbitrary distribution.

Theorem 3.3.1. *Given a random number, r, generated with a probability distribution given by a top hat probability distribution function*

$$P(R = r) = \begin{cases} \frac{1}{2} & if -1 \leq r \leq 1 \\ 0 & otherwise \end{cases}$$

then for any arbitrary distribution $P(X = x)$

$$P(X = x)dx = P(R = r)dr$$

where

$$x = C^{-1}\left(X = \frac{r+1}{2}\right)$$

and $C^{-1}(X = x)$ is the inverse of the cumulative probability function of $P(X = x)$

Proof. Let R be a random variable so that $P(R = y)$ has a top hat probability distribution as above. Let $f(y)$ be any monotonically increasing function and let X be the image of R under the transformation f. It is a standard result that

$$P(X = f(y))df(y) = P(R = y)dy \tag{3.1}$$

integrating both sides from $-\infty \leq y \leq r$ gives

$$\int_{f(-\infty)}^{f(r)} P(X = y)dy = \frac{r+1}{2}$$

for $-1 \leq r \leq 1$, but the left hand side of this equation is just the definition of the cumulative probability function of $P(X = y)$, call it $C(X = y)$. So

$$C(X = f(r)) = \frac{r+1}{2}$$

so, if we let $C^{-1}(X = x)$ be the inverse of the resulting cumulative probability distribution then

$$f(r) = C^{-1}\left(X = \frac{r+1}{2}\right)$$

substituting this into equation 3.1 gives the required result. □

Although there does not always exist an analytic function for the inverse of the cumulative probability, $C^{-1}(y)$, for a given $P(y)$, this can always be approximated to any arbitrary precision by using, for example, a polynomial fit or some other analytic approximation.

3.4 Conclusion

In this chapter we have seen how a computer program can be converted into a vector of functions of the form $\frac{Q}{R}$ where Q and R are polynomials. Simple examples were given to illustrate how a computer program can be converted into such a vector, approximated and turned back into a computer program that approximates the original.

However, this chapter is meant to provide a theoretical grounding, there remain many practical considerations to be addressed before this method can be applied to computer programs of more realistic complexity. In the next two chapters we introduce the techniques necessary to develop this into a working implementation, and describe a number of numerical experiments that we have performed to test these techniques.

Chapter 4

Experiments with Chebyshev Polynomials

4.1 Introduction

A number of numerical experiments were performed to test the method of approximating programs described in the previous chapter. To do this, a program was written in C++ which analyses programs written in C++ and approximates them. We called this program *iGen*. The main obstacle that needed to be addressed in a practical implementation of this method was the exponential explosion of the degree of the polynomial rationals. For simple programs as in the examples given in the previous chapter (and for many not so simple programs) it is possible to explicitly calculate the polynomial rational for each of the program's output variables; modern computers are powerful enough to easily manipulate polynomials with thousands of terms. However, for many real world models the equivalent polynomials are of much too high an order to be stored or manipulated explicitly. A model of the atmosphere that integrates over only a few tens of timesteps, for example, would be equivalent to a vector of polynomials exceeding many millions of terms.

To deal with this, we began by representing variables as polynomials in the Chebyshev basis. The Chebyshev basis is a sequence of polynomials defined, for

all integers n as

$$T_n(x) = \cos\left(n \cos^{-1}(x)\right)$$

on the interval $-1 \leq x \leq 1$. This is known as *the n^{th} Chebyshev polynomial of the first kind*. As a consequence of De Moivre's theorem, which can be stated as:

$$\cos(nx) + i\sin(nx) = (\cos(x) + i\sin(x))^n$$

and of the identity

$$sin^2(x) + cos^2(x) = 1$$

it can be seen that $T_n(x)$ can be expressed as an n^{th} degree polynomial in x. The first few of which are:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2x^2 - 1$$

$$T_3(x) = 4x^3 - 3x$$

$$T_4(x) = 8x^4 - 8x^2 + 1 \ .$$

Any polynomial $P(x)$ of degree m, can be uniquely expressed as a sum of Chebyshev polynomials:

$$P(x) = \sum_{n=0}^{m} a_n T_n(x)$$

as can be seen by solving for degree m and working our way down to zero, so the set of coefficients a_n uniquely defines a polynomial P.

The Chebyshev basis was chosen for this implementation because of its good approximation properties. It is a well known result (see, e.g. Mason and Handscomb, 2003) that a close to optimal n^{th} degree approximation of an $(n+m)^{th}$ degree polynomial can be found by simply setting the highest m coefficients of a Chebyshev series to zero. Although there exists a unique, optimal n^{th} degree approximation and an algorithm to find it (Remez' algorithm, see, e.g. Cheney, 2000), this algorithm uses an iterative search method and so is much more computationally expensive to calculate than the Chebyshev method. The extra computational time necessary to find the optimal approximation is generally considered to be rarely worth while (e.g. Press et.al., 2007).

4.2 Approximate algebra with Chebyshev bounds

Our program implemented an approximate algebra on pairs (P, ϵ), where P is a multivariate Chebyshev polynomial and ϵ is a constant bound on the absolute error between P and the exact value of the variable it represents. So, (P, ϵ) represents the interval $[P - \epsilon : P + \epsilon]$. We call these pairs "Chebyshev bounds".

When a Chebyshev bound, (P, ϵ), becomes too large to manipulate efficiently, P is replaced by a (smaller) approximation and a constant bound on the error introduced by the approximation is added to ϵ. The approximation is found by ordering the terms of P by the absolute value of their coefficient. The coefficient with the smallest value is removed from P. Since each Chebyshev polynomial basis function is bounded by ± 1, the magnitude of the error introduced by removing this term is bounded by the magnitude of its coefficient, so this is added to ϵ. Then the next smallest term is removed and the absolute value of it's coefficient added to ϵ. This process is repeated until the remaining polynomial reaches a certain size or a limit on the acceptable error is reached. This process can be summarised as

$$(A + e, \epsilon) = (A, \epsilon + B(e))$$

where $B(e)$ is a function that returns the sum of the absolute values of the Chebyshev coefficients of e. Closer bounds than $B(e)$ could have been found by first converting e to a Bernstein polynomial. Rababah (2003) shows how the conversion can be done and Cargo and Shisha (1966) show how Bernstein polynomials can be used to obtain bounds. This method was implemented but it was found that the increase in accuracy of the bounds was outweighed by the computational effort of calculating them; better accuracy being achieved when the computational time was used to retain more terms of A instead.

4.2.1 Addition, Subtraction and multiplication

Addition, subtraction and multiplication of Chebyshev bounds is straightforward and implemented using the rules

$$(P, \epsilon_1) + (Q, \epsilon_2) = (P + Q, \epsilon_1 + \epsilon_2)$$

$$(P, \epsilon_1) - (Q, \epsilon_2) = (P - Q, \epsilon_1 + \epsilon_2)$$

$$(P, \epsilon_1) \times (Q, \epsilon_2) = (P \times Q, B(P)\epsilon_2 + B(Q)\epsilon_1 + \epsilon_1\epsilon_2)$$

where $B(P)$ is a bound on the absolute value of the polynomial P given by the sum of its Chebyshev coefficients.

4.2.2 Division

Division of Chebyshev bounds is somewhat more complicated because of the need to maintain formal bounds. We would like to solve

$$P = \frac{Q \pm \epsilon_q}{R \pm \epsilon_r} = \frac{Q}{R} \pm \epsilon_p$$

for ϵ_p. A little manipulation gives

$$P = \frac{Q}{R} \pm \left(B\left(\frac{1}{R}\right) \epsilon_q + B\left(\frac{P}{R}\right) \epsilon_r \right)$$

However, this involves calculating $B(\frac{1}{R})$ and $B(\frac{P}{R})$. This could be done explicitly by performing the polynomial divisions but this would be computationally expensive. Alternatively the inequality

$$B\left(\frac{P}{R}\right) \leq \frac{B(P)}{B_l(R)}$$

could be used, where $B_l(R)$ is a lower bound on R. Calculating $B_l(R)$ accurately, however, would also be computationally expensive (Cargo and Shisha, 1966; Lin and Rokne, 1995; Stahl, 1995; Cornelius and Lohner, 1984).

Reciprocation of a Chebyshev bound, on the other hand, has a neater form. We would like to solve

$$P = \frac{1}{R \pm \epsilon_r} = \frac{1}{R} \pm \epsilon_p$$

which gives

$$P = 1/R \pm \epsilon_r B(P^2)$$

$B(P^2)$ can be calculated without having to square P by using the inequality

$$B(P^2) \leq B(P)^2 \ .$$

So, division of Chebyshev bounds was performed by multiplication of the numerator with the reciprocal of the denominator. From the arguments above, reciprocation of Chebyshev bounds follows the rule

$$\frac{1}{(R, \epsilon)} = \left(\frac{1}{R}, \epsilon B\left(\frac{1}{R}\right)^2 \right) \ .$$

It was found that during the analysis of realistic programs, division was much less common than multiplication, addition and subtraction of Chebyshev bounds. For this reason it was deemed more computationally efficient to approximate reciprocals of polynomials as polynomials, so that subsequent operations would be on polynomials rather than polynomial rationals.

For ease of implementation, reciprocation of a polynomial, R, was performed by first scaling the range of R to lie in the interval $[-1:1]$, then using the recursion

$$P_{n+1} = P_n(2 - P_n R)$$

which converges on $P = \frac{1}{R}$ in the region bounded by $P = 0$ and $P = \frac{2}{R}$. Numerical experiments showed that the best first-order value for P_0 that optimises the rate of convergence is

$$P_0 = 2.9938 - 2.172R$$

if the range of R can be shown to be contained by the interval $[0:1]$,

$$P_0 = -2.9938 - 2.172R$$

if the range of R can be shown to be contained by the interval $[-1:0]$ and

$$P_0 = 1.8045R$$

otherwise. Loose bounds on the range of R were calculated as $[r_0 - B(R - r_0) : r_0 + B(R - r_0)]$ where r_0 is the zeroth degree coefficient of R. Here, it doesn't matter that the bounds are loose as it doesn't effect the result, just the rate of convergence to the result.

One advantage of this algorithm is that the residual is known at each iteration since it is given by $1 - P_n R$. Iteration stops when the residual can be bounded below the desired accuracy.

A more computationally efficient algorithm could be written by finding the polynomial P that satisfies $PR = 1$ for all coefficients up to the degree of P. In the Chebyshev basis, if the degree of P is known, this just involves solving a set of simultaneous equations. However, we do not know a-priori the degree of P necessary to obtain a given accuracy so some guesswork would be involved. Alternatively, R could be transformed to a power series polynomial. In this basis $PR = 1$ can be solved in linear time by solving for the lowest orders first and working upwards to higher orders. The algorithm could continue in this

fashion until the desired accuracy is achieved. However, in this basis, much higher degree terms would need to be calculated to achieve the same bound on accuracy as in the Chebyshev basis.

4.3 Transcendental functions

Strictly speaking, the transcendental functions (log, exp, cos etc.) cannot be expressed as polynomials. However, all computer implementations of these functions use algorithms that reduce to polynomials (e.g. Kropa 1978). These could be analysed symbolically using the methods described above. However, by implementing versions designed especially for symbolic evaluation, the analysis can be made much more efficient.

4.3.1 Non-integer powers of polynomials

Non-integer powers of Chebyshev bounds were implemented by noting that any power x^n can be expressed in the form

$$x^n = \begin{cases} \frac{B_u(x)^n \left(\left(\frac{x}{B_u(x)}\right)^q\right)^N}{x} & \text{if } n < 1 \\ B_u(x)^n \left(\left(\frac{x}{B_u(x)}\right)^q\right)^N & \text{otherwise} \end{cases}$$

where N is the lowest integer not smaller than $\frac{n}{2}$, $1 \leq q < 2$ and $B_u(x)$ is an upper bound on x. Since raising polynomials to an integer power N is easily implemented in $O(ln(N))$ multiplications, this reduces the problem to that of finding R^q for a polynomial whose range is in the interval $[0:1]$. This can be easily implemented as a Chebyshev polynomial $P(x,q)$ which approximates x^q over the necessary ranges, giving $R^q = P(R(x),q) \pm \epsilon_p$, where ϵ_p is a bound on the error due to the approximation in P. Composition of Chebyshev polynomials was reduced to a sequence of additions and multiplications using Clenshaw's recurrence (Clenshaw, 1962; Press et.al., 2007).

4.3.2 Exponentiation of polynomials

The function e^P on a polynomial P was implemented by noting that

$$e^P = e^{(P'+c_0)} = e^{c_0} \left(e^{\frac{P'}{M}}\right)^M$$

where M is a the lowest integer that bounds P' above and below and c_0 is the zeroth degree coefficient of P. It is clear that $-1 \leq \frac{P'}{M} \leq 1$, so $e^{\frac{P'}{M}}$ can be approximated by composition with a Chebyshev polynomial that approximates exponentiation over this range.

4.3.3 Sine and cosine

Sine and cosine functions can be implemented with help from the trigonometric definition of the Chebyshev polynomials:

$$T_n(x) = \cos\left(n \cos^{-1}(x)\right)$$

so that

$$\cos(x) = \cos\left(N \cos^{-1}\left(\cos\left(\frac{x + 2\pi M}{N}\right)\right)\right)$$

where N is the smallest integer not smaller than $\frac{B_u(x)}{2\pi} + M$ and M is the smallest integer not smaller than $\frac{B_l(x)}{2\pi}$ (B_u and B_l are upper and lower bounds on x). So

$$\cos(x) = T_N\left(cos\left(\frac{x + 2\pi M}{N}\right)\right) \ .$$

The cosine on the right hand side is now bounded between $[0 : 2\pi]$ and can be approximated by a Chebyshev polynomial.

The sine function can then be trivially implemented as $sin(x) = cos(x + \frac{pi}{2})$.

4.4 Random numbers

Integration over random numbers was performed analytically on the Chebyshev polynomials using the identity on Chebyshev polynomials (Mason and Handscomb, 2003)

$$\int T_n(x)dx = \begin{cases} \frac{1}{2}\left(\frac{T_{n+1}(x)}{n+1} - \frac{T_{|n-1|}(x)}{n-1}\right) & \text{if } n \neq 1 \\ \frac{1}{4}T_2(x) & \text{if } n = 1 \ . \end{cases}$$

Error bounds are unchanged on integration since

$$\int_{-\infty}^{\infty} (R(r) \pm \epsilon)P(r)dr = \int_{-\infty}^{\infty} R(r)P(r)dr \pm \epsilon \int_{-\infty}^{\infty} P(r)dr$$

but

$$\int_{-\infty}^{\infty} P(r)dr = 1 \ .$$

If a program calls the `rand()` function many thousands of times then adding an extra variable for each call would be undesirable as it would lead to an unnecessarily large number of variables. In this case the random numbers are considered to be created by a pseudo random number generator $r(n, s)$ where s is a random 'seed' which gets the generator started and $r(n, s)$ is the n^{th} random number to be generated (see, e.g. Press et.al., 2007, chapter 7). We then integrate over the value of the seed rather than integrate separately over each random number. In this way, we introduce only one extra variable, s, for any number of calls to `rand()`. This method can be justified by the same argument as used to justify Monte-Carlo simulations (Metropolis and Ulam, 1949).

Most standard pseudo-random number generators rely on modulo algebra, bit operations or overflow of variables. Analysis of these tends to produce very high order polynomials so a direct analysis of a random number generator would not be ideal for our purposes. Instead, we use an alternative generator that returns polynomials of any desired order and is constructed in the following way: Choose any m seeds $s_1...s_m$ for a standard pseudo random number generator $r(n, s)$. In our implementation, we chose the Gauss-Lobatto collocation points on the interval $[-1 : 1]$ as the seeds. During an analysis, the n^{th} call to `rand()` returns the unique m^{th} order polynomial in s that passes through the points $(s_1, r(n, s_1))...(s_m, r(n, s_m))$ (this can be found in $O(mlog(m))$ time for the Gauss-Lobatto points by using an FFT). Note that the calls to r are with actual values, not polynomials. To average over s, rather than integrating using a top-hat function for the distribution of s, use the distribution

$$P(s) = \frac{\sum_{l=1}^{m} \delta(s - s_l)}{m}$$

where δ is the Kronecker delta function. The result after integration is equal to that of averaging over m monte-carlo simulations using the standard pseudo random number generator. Doing this symbolically, rather than as a Monte Carlo simulation has the advantage that it allows approximations to be made that may reduce the amount of computation required to calculate the averages.

If this method is not sufficient for some application, a more powerful method could be implemented: Since the analysis of the program can be done in any order, we could choose to evaluate it starting with the output and working our way backwards towards the input. The first call to `rand()` that is encountered in

the analysis will be the last to be executed and can be immediately integrated over before analysis continues backwards. In this way, random numbers are eliminated as soon as they are encountered and any number of calls to `rand()` can be encountered without introducing large numbers of variables.

In some cases, backward evaluation can also be more efficient than forwards evaluation. Similar results have been found for the same reasons in the different context of automated differentiation of programs (Werbos, 2006).

4.5 Lorenz equations

iGen was tested on a source program which simulates the Lorenz equations:

$$\frac{dX}{dt} = \sigma(Y - X)$$

$$\frac{dY}{dt} = rX - Y - XZ$$

$$\frac{dZ}{dt} = XY - bZ$$

where σ, r and b are constants. The simulation used a simple forward finite difference method to integrate the equations, as shown in figure 4.1.

Unless otherwise stated, the start state of the simulation was that used by Lorenz:

$x = 0.0$
$y = 1.0$
$z = 0.0$
$s = 10.0$
$b = \frac{8}{3}$
$r = 28$
$\Delta t = 0.01$ (the timestep).

The Lorenz code was wrapped in three different ways, as shown in table 4.1. An execution of the source code over 150 timesteps takes around 1200 multiplications and 1050 additions. So, to calculate the average values of the three parameterisations using an ensemble run of, say, 20 explicit simulations would take round 24000 multiplications and 21000 additions. iGen analysed the three wrapped codes and produced three parameterisations with much improved execution times, as shown in table 4.2.

```
void lorenz(double r, int I, double &X, double &Y, double &Z) {
  const double s = 10.0;
  const double b = 8.0/3.0;
  const double Dt = 0.01;
  double       dx,dy,dz;
  int          iteration;

  for(iteration = 0; iteration < I; ++iteration) {
    dx = s*(Y-X);
    dy = r*X - Y - X*Z;
    dz = X*Y - b*Z;

    X += dx*Dt;
    Y += dy*Dt;
    Z += dz*Dt;
  }
}
```

Figure 4.1: Program to integrate the Lorenz equations

iGen took less than one second to produce a parameterisation on a 1.66GHz Intel Core-2 processor. Memory usage was less than 1Mb.

4.6 Ideal Gas

The next test was a program that simulates an atom bouncing around a 2-dimensional box of unit dimension. The input to the program is the start position and velocity of the atom. The output was the pressure on the right wall, given by the average impulse per second:

$$\bar{I} = \frac{2Nmv_x}{t}$$

where N is the number of impacts with the wall during the simulation, m is the mass of the atom (taken to be of numeric value 1), v_x is the velocity of the atom perpendicular to the wall and t the simulated duration of the simulation.

The model was wrapped so that its input is the speed of the atom. This was converted to the simulation's input by randomly choosing the atom's position and direction of motion so as to give an isotropic, homogeneous distribution within the box.

Thermodynamics tells us that the pressure on the wall should be proportional to the temperature of the gas, and kinetic theory tells us that the temperature is proportional to the square of the speed of the atom.

iGen was executed with this wrapped program as input in order to find the first moment of pressure in terms of the speed of the atom. The program correctly identified the proportionality between the pressure on the wall and the square of the speed of the atom. The time taken by the analysing program was less than one second on a 1.66GHz Intel Core-2 processor. Memory usage was less than 1Mb.

4.6.1 Reasoning with Heaviside functions

The presence of `if` statements in this program meant that the analysis involved Heaviside functions. These were not expanded into their approximate polynomials but kept as separate terms. Without any extra reasoning, the number of terms in the output polynomial would double for each time an `if` statement

Parameterisation 1	The initial value of Y was unknown, but lies somewhere in an interval $[Y_{min}, Y_{max}]$ with a flat distribution. The output was the average value of Y after 150 timesteps.
Parameterisation 2	The initial value of r is unknown but lies somewhere in an interval $[r_{min}, r_{max}]$ with a flat distribution. The output was the average value of X after 150 timesteps.
Parameterisation 3	The initial value of r is unknown but lies somewhere in an interval $[r_{min}, r_{max}]$ with a flat distribution. The output was the average value of Y after 150 timesteps.

Table 4.1: The three wrapped concrete functions of the Lorenz equations

Parameterisation	Multiplications	Additions	Accuracy (sig. figs)
Parameterisation 1	9	8	5
Parameterisation 2	44	43	5
Parameterisation 3	100	99	3

Table 4.2: The execution speed and accuracy of the parameterisations

```
double atomicTheory(double x, double y, double vx, double vy) {
  const double DT       = 0.01;         // timestep
  const double TMAX     = 10.0;         // duration of simulation

  double p = 0.0; // pressure
  double t = 0.0; // time

  while(t < TMAX) {
    x = x + vx*DT;
    y = y + vy*DT;
    if(x > 1.0) {
      x  = 2.0 - x;
      p = p + (2.0 * vx);
      vx = -vx;
    }
    if(x < 0.0) {
      x  = -x;
      vx = -vx;
    }
    if(y > 1.0) {
      y  = 2.0 - y;
      vy = -vy;
    }
    if(y < 0.0) {
      y  = -y;
      vy = -vy;
    }
    t = t + DT;
  }
  p = p/TMAX;
  return(p);
}
```

Figure 4.2: Program to simulate an atom bouncing around a 2-dimensional box

```
double kineticTheory(double s) {
  const double PI       = 3.14159;      // pi

  double angle;                         // initial angle
  double p;                             // pressure
  double x;                             // x-position
  double y;                             // y-position
  double vx;                            // x-velocity
  double vy;                            // y-velocity

  // set up initial position and angle at random
  x = Rand();
  y = Rand();
  angle = 2.0 * PI * Rand();
  vx = s*sin(angle);
  vy = s*cos(angle);

  p  = atomicTheory(x,y,vx,vy);

  return(p);
}
```

Figure 4.3: Wrapper for the atomicTheory simulation

is executed, and so the size of the representation would explode exponentially. To prevent this, a number of reasoning rules were applied to the representation whenever its size became too large.

If the argument to a Heaviside function could be proved not to cross zero then the Heaviside could be replaced with 0 or 1:

$$H(A) = 1 \text{ if } B_l(A) > 0$$

where $B_l(A)$ is a lower bound on A.

$$H(A) = 0 \text{ if } B_u(A) < 0$$

where $B_u(A)$ is an upper bound on A. Lower and upper bounds on Chebyshev polynomials were calculated using $a_0 \pm B(A - a_0)$, where a_0 is the zeroth order coefficient of A.

To simplify the form of complex booleans, the following identities were used whenever the left hand sides were encountered.

$$1 - H(A) = H(-A)$$

$$H\left(H(A)P + H(-A)Q\right) = H(A)H(P) + H(-A)H(Q)$$

$$\begin{array}{ll} \text{if} & H(A)H(B)H(-C) = 0 \\ \text{and} & H(-A)H(C)H(-B) = 0 \\ \text{then} & H(A)H(B) + H(-A)H(C) = H(B)H(C)\ . \end{array}$$

The final identity is proved by noting that

$$H(A)H(B)(H(C) + H(-C)) + H(-A)H(C)(H(B) + H(-B)) = H(B)H(C) + H(A)H(B)H(-C) + H(-A)H(C)H(-B)\ .$$

This may seem like a rather arbitrary piece of reasoning, but because of the way `if` statements split the input space into two partitions, this structure was found to occur quite often. Its effect is to join together neighbouring partitions that have the same approximation.

Products of Heaviside functions of the form

$$H(P_1)H(P_2)...H(P_N)$$

can sometimes be proved to be trivially true or false, and so replaced by 1 or 0 respectively. The problem reduces to that of deciding whether a set of inequalities on polynomials is satisfiable. Algorithms exist that can always detect

this but they tend to be inhibitively slow to execute. The first algorithm was due to Tarski (1951) but this ran in worse than exponential time. Exponential time algorithms were found by Seidenberg (1954) and later by Collins (1975). More recently, a sub-exponential time algorithm has been found by Grigorev and Vorobojov (1988) but execution times remain high for our purposes.

It was found that a simple and fast algorithm based on the Gaussian elimination method was sufficiently powerful to detect all instances encountered in the 'kinetic theory' program. The algorithm first transforms the inequalities to equalities in the following way: Each Heaviside term $H(P_n)$ is equivalent to the inequality $P_n > 0$. Since P_n can be bounded above by $B_u(P_n)$ (as calculated using the sum of its Chebyshev coefficients) then there exists a y_n in the range $0 < y_n \leq B_u(P_n)$ that satisfies $P_n - y_n = 0$. If we let

$$y_n = \frac{B(P_n)(1 + z_n)}{2}$$

then z_n is in the range $[-1 : 1]$ and can be treated as a normal Chebyshev variable. This leads to a set of equalities

$$P'_n = P_n - \frac{B(P_n)(1 + z_n)}{2} = 0$$

for all $0 < n \leq N$.

Once in this form, the highest degree terms that occur in more than one equation can be successively removed by Gaussian elimination. At each stage, the bounds of the remaining polynomials are checked. If any has an upper bound that is below zero or lower bound above zero, the equation cannot be satisfied and so there is no solution. Note that if an equation is reduced to a sum of first degree terms, B_u and B_l become tight bounds so it can immediately be seen whether the equation is satisfiable or not.

An equation P'_n was removed if it was implied by the set of earlier equations $P'_1 ... P'_m$ where $m < n$. Implication was proved if P'_n could be reduced to a form $z_n = Q$ and Q could be bounded by the interval $[-1 : 1]$.

A possibly better algorithm is as follows: Satisfaction of the simultaneous equations implies that

$$S = \sum_{n=1}^{N} \left(P_n - \frac{B(P_n)(1 + z_n)}{2} \right)^2 = 0 \ .$$

This polynomial is strictly non negative. If it can be shown to be bounded above zero over the range of all variables then the equations cannot be satisfied and

the product of Heaviside functions must not be satisfiable. To check whether a polynomial is bounded above zero, first convert it to Bernstein form. If all Bernstein coefficients are above zero, the polynomial cannot be equal to zero anywhere (Cargo and Shisha, 1966). The converse is not true; if there exists a negative or zero Bernstein coefficient, the polynomial doesn't necessarily touch the $S = 0$ plane. In this case, take the coordinates of the maximum of the Bernstein polynomial that has the negative coefficient and use them as the start point of a Newtonian approximation to get a better approximation of the value of the minimum (since S is strictly non-negative, if it touches $S = 0$, it must be at a minimum point).

4.7 Rayleigh-Benard convection

The Lorenz equations are themselves a simplified model of Rayleigh-Benard convection. To show that the compiler can cope with finite difference equations on gridded data, a governing model was written that simulates laminar convection on a 80x28 grid. The model was wrapped so that its inputs were the Lorenz parameters x, y and z, the integration was over a single timestep with $r = 28$ and $b = \frac{8}{3}$. The output was the change in x, y and z divided by the timestep. iGen analysed the source code and produced the following simplified code:

```
input(x,y,z)
  dx_dt = 9.95076*y - 9.94443*x
  dy_dt = -0.991175*x*z - 0.999187*y + 27.9712*x
  dz_dt = -2.65625*z + 0.997019*x*y
output(dx_dt, dy_dt, dz_dt)
```

which differs from the Lorenz equations by less than 0.9% in the constants and represents an increase in execution speed of 5 orders of magnitude compared to the wrapped model. The slight difference between the analysis and the Lorenz equations is attributed to the finite resolution of the wrapped model's grid, the finite time over which the integration was performed and the accuracy of the algorithm used to solve the Poisson equation in the simulation. The error between the outputs of the simplified model and the wrapped model is bounded by 0.1% of the maximum value of each output variable.

iGen was used to make an alternative set of equations which model Rayleigh-Benard convection more accurately than the Lorenz equations for the variables and timestep used by Lorenz ($0.01\tau = 675\mu s$). This was done by wrapping the model so that its inputs were the Lorenz parameters x, y and z, the integration was over a duration of 0.01τ and the output was the change in the Lorenz parameters divided by the duration of integration. The resulting simplified program was

```
input(x,y,z)
  Dx_Dt = -0.04088*x*z + 9.554*y - 8.401*x
  Dy_Dt = -0.04140*y*z - 0.9398*x*z + 0.1897*y + 26.74*x
  Dz_Dt = -2.629*z + 0.02103*y*y + 0.9521*x*y + 0.05570*x*x + 0.07673
output(Dx_Dt, Dy_Dt, Dz_Dt)
```

Where the acceptable error in each output, compared to the wrapped model, was specified as 0.1% of the maximum value of each output variable.

4.8 Mie scattering

A program was written to simulate the scattering of parallel light by spherical water droplets. This was done using Mie theory (see, for example, Bohren and Huffman, 1998). The equations solved by the program are given in appendix A.

The program was wrapped to calculate the scattering cross section per unit mass of water for light of wavelength 500nm scattered by a thin layer of cloud made up of spherical water droplets with complex refractive index of $1.33 + 1 \times 10^{-8}\imath$ relative to the surrounding air. This was done by leaving the radius of the droplets unspecified, and instead giving a probability distribution over possible radii. The probability distribution was specified to take the form of a gamma distribution, given by

$$P(r) = Ar^{\alpha} \exp^{-\beta r}$$

where

$$\alpha = \frac{1}{v_e} - 3.0$$

and

$$\beta = \frac{1}{v_e r_e}$$

and A is a normalisation factor, v_e is the relative 'effective variance' of the distribution and is set to 0.172, and r_e defines an 'effective radius' of the droplets. r_e was taken as the input of the wrapped model, and defined to lie in the range $5\mu m$ to $40\mu m$. The output of the wrapped model was defined to be the reciprocal of the scattering cross section per unit mass.

iGen was used to analyse this wrapped model and produced the simplified model for the scattering cross section K_{sca}:

$$K_{sca} = \frac{1}{660.1r_e - 2.188 \times 10^{-4}}$$

with an error bounded by $4m^2kg^{-1}$. This is plotted in figure 4.4 together with the exact result calculated using numerical integration.

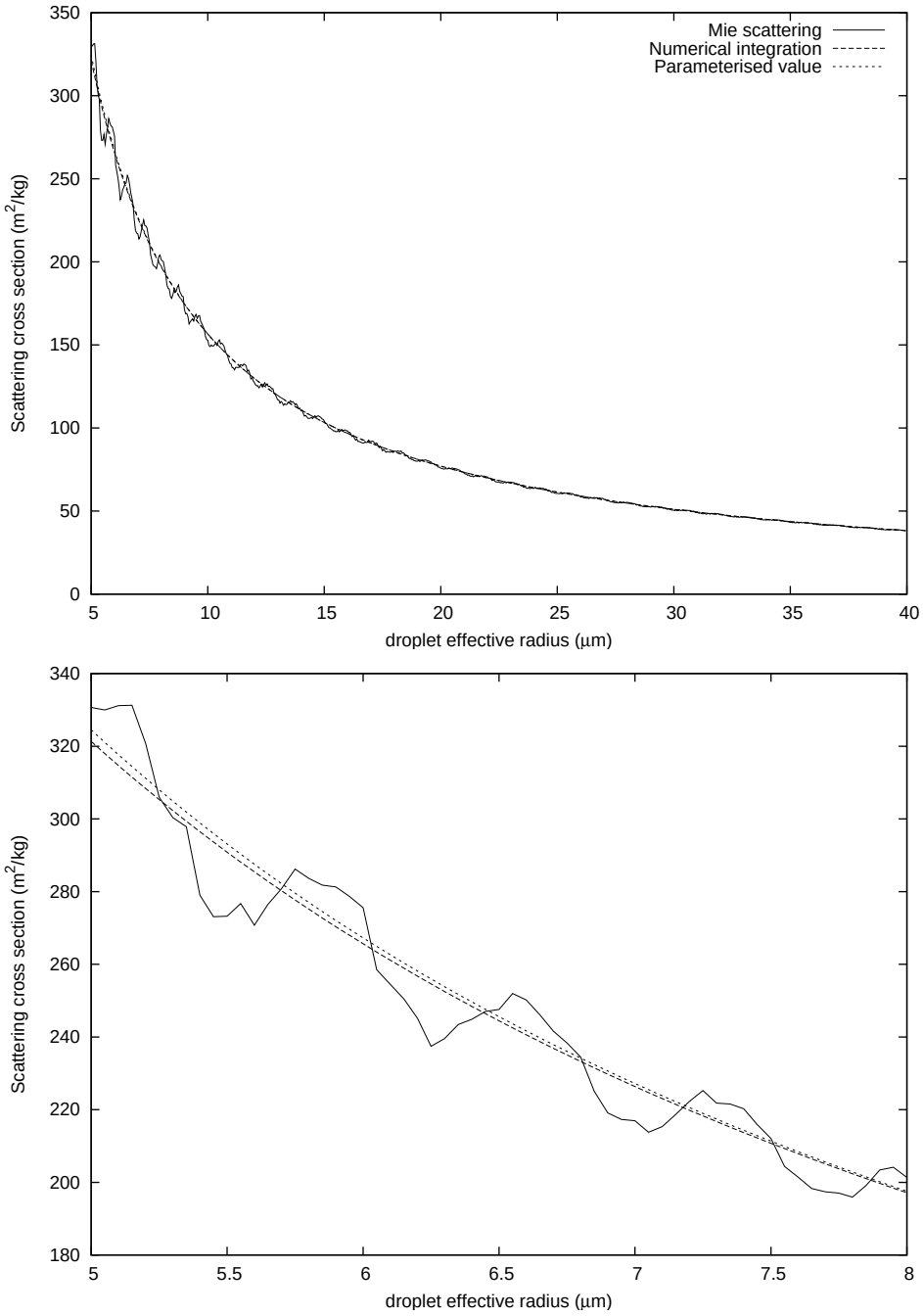

Figure 4.4: Plot of scattering cross section for fixed radius droplets (solid line), numerically integrated over the droplet radius distribution (dashes), and iGen's simplified model (dots). For clarity, a smaller portion is reproduced in the lower plot.

Chapter 5

Experiments with DeSelby Polynomials

5.1 Introduction

The experiments with Chebyshev polynomials described in the previous chapter showed that the Chebyshev basis is a powerful tool for approximating computer programs. However, the experiments also uncovered a number of drawbacks to this representation. It was found that the multiplication of Chebyshev polynomials was a significant computational bottleneck. In order to multiply two monovariate Chebyshev polynomials of N terms each, it takes $O(N^2)$ computer operations using the naïve algorithm of multiplying each term separately. Even worse, in the D dimensional case, it takes $O(N^2 2^D)$ operations. This represents a significant computational load when dealing with polynomials of many thousands of terms. Another problem was encountered when the value of a polynomial spans more than about 15 orders of magnitude over the domain of the input. In this case, the Chebyshev coefficients become very large and the value of the polynomial depends on the cancellation of these very large basis terms. Analytically this is not a problem but when the Chebyshev coefficients are stored in computer memory as finite precision floating point numbers, truncation errors in floating point arithmetic become significant.

The final problem concerns the interaction between higher and lower order bases.

When we are given a program to approximate, its inputs can be represented by first order polynomials, so these can be represented very easily. The outputs also will generally be quite smooth, perhaps with the exception of a few discontinuities in the higher order rates of change, so there does exist a good polynomial approximation of reasonable size (i.e. that will fit in the memory of a PC). This is a consequence of the fact that any well conditioned problem should be insensitive to small perturbations in the inputs. The problem with the Chebyshev basis is that although the inputs and outputs are representable, there may be a necessity for extremely large polynomials to represent the value of variables mid way through a simulation. This is because a change in a very high order Chebyshev coefficient of a mid-way-through variable can have a non-trivial effect on the low order coefficients of the output. This is a consequence of the way Chebyshev polynomials multiply: when two high degree polynomials are multiplied, the values of their high order coefficients affect the low order coefficients of the result as well as the high. This is not the case for all bases. For example, the power series polynomials do not have this property: multiplying two power-series terms can only lead to a higher-power term. However, power-series polynomials also lack the good approximation properties of the Chebyshevs. What we need for our purposes is a polynomial basis that combines the multiplicative qualities of the power series with the approximative qualities of the Chebyshev polynomials, in this way we can indefinitely avoid the exponential explosion in polynomial size while producing close to optimal approximations.

5.2 DeSelby polynomials

For this purpose, a new type of polynomial was invented which we decided to call the DeSelby polynomials[1]. The polynomial can be thought of as consisting

[1]Rather than follow the somewhat egocentric tradition of naming polynomials after their inventor, I instead name these polynomials after the much more deserving but largely unrecognised DeSelby. Very little is known about the details of DeSelby's life and personality, aside from his inability to distinguish between the sexes; famously referring to the Countess Schnapper as 'that cultured old gentleman' and to his own mother as 'a man of stern habits' and 'a man's man'. DeSelby had no children. Of his work, the largest remaining evidence is the so called 'Codex': a collection of some two thousand sheets of foolscap closely hand-written on both sides. The true import of the manuscript is not at all clear and has engendered more than a little debate. On this matter O'Brein (1993) writes "Attempts made by different commentators to decipher certain passages which look less formidable than the others have been

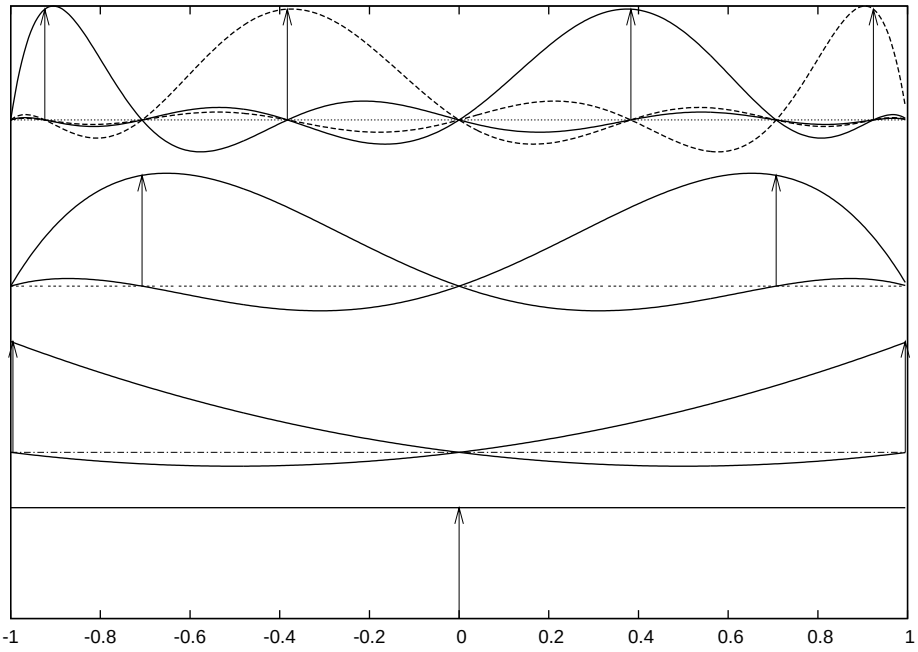

Figure 5.1: The first 9 DeSelby basis functions, grouped into shells. The first shell is at the bottom, the fourth at the top. The arrow shows the points at which one basis function has a value 1 and all functions of higher degree have a value 0

of a sequence of 'shells', each of successively higher order than the last (see figure 5.1). The first shell is just a constant, which can be thought of as a zeroth order approximation. The next shell gives 2^{nd} order perturbations to the first shell. The sum of the first and second shells gives a 2^{nd} order approximation. The next shell is a 4^{th} order perturbation to the 2^{nd} order approximation, and so on for the 8^{th}, 16^{th}, 32^{nd}...order approximation.

This basis has the important property that the n^{th} basis function can be associated with a point at

$$x_n = \begin{cases} \cos\left(\frac{\pi 2n}{2^{\lfloor ln_2(n)+1 \rfloor}}\right) & \text{if } n < 2 \\ \cos\left(\frac{\pi 2n+1}{2^{\lfloor ln_2(n)+1 \rfloor}}\right) & \text{otherwise} \end{cases}$$

characterised by fantastic divergences, not in the meaning of the passages (of which there is no question) but in the brand of nonsense which is evolved. One passage, described by Bassett as being 'a penetrating treatise on old age' is referred to by Henderson (biographer of Bassett) as 'a not unbeautiful description of lambing operations on an unspecified farm'."

which is equal to 1 in the n^{th} basis function and 0 in all basis functions of higher degree. This entails that when two polynomials are multiplied, $PQ = M$, the higher degree coefficients of P and Q have no effect on the lower degree coefficients of M (see theorem 5.5.1). So, lower orders can be correctly calculated without knowledge of the higher orders. In addition, the error introduced by truncating an entire shell can be bounded by twice the bound on the error introduced by Chebyshev truncation to the same degree (see theorem 5.5.2). So DeSelby polynomials also have the multiplicative and approximative properties we required.

5.2.1 Computing with DeSelby polynomials

The problem of dealing with polynomials that span many orders of magnitude was solved by developing an algorithm that converts between the DeSelby representation and the Gauss-Lobatto representation in $O(N)$ operations, where N is the number of terms in the polynomial.

The Gauss-Lobatto representation of an N^{th} degree polynomial consists of a set of coefficients $y_0...y_N$ that are just the values of the polynomial at the collocation points

$$x_i^N = \cos\left(\frac{\pi i}{N}\right) \ , \ 0 \leq i \leq N \ .$$

So, a set of Gauss-Lobatto coefficients $y_0...y_N$ represents the unique N^{th} degree polynomial that passes through the points $(x_0, y_0), ..., (x_N, y_N)$. In this representation, arithmetic operations can be performed without loss of precision even if the value of a polynomial spans many orders of magnitude.

In common with the Chebyshev polynomials, an exact conversion to or from the Gauss-Lobatto representation takes $O(NlogN)$ operations to perform. However, an algorithm was devised that allowed an approximate transformation to be performed between DeSelby and Gauss-Lobatto representation in only $O(N)$ operations (see section 5.6.1). This allowed the polynomial to be efficiently transformed to DeSelby representation without causing a computational bottleneck. This allowed the development of algorithms for all necessary functions that took only $O(N)$ operations. This is a very significant result, particularly in the case of multiplication. As mentioned above, the naïve algorithm to multiply two Chebyshev polynomials takes $O(N^2 2^D)$ operations, where N is the number of terms and D is the number of variables. So, for a 5-variable polynomial with

2000 terms that's something of the order of 128×10^6 operations. With DeSelby polynomials this is reduced to just $2,000$ operations, over $60,000$ times faster.

The approximate transformation algorithm was extended to allow fast, approximate differentiation of DeSelby polynomials (see section 5.6.2). This algorithm is, to the author's knowledge, new. It is significant not only in the context of differentiating DeSelby polynomials but could also be adapted to allow differentiation on uniform grids. This means that it could be used as a much higher-order accurate replacement for finite difference calculations without significant penalty in speed. This algorithm has the additional advantage that bounds can be put on the error between the calculated rate of change and the exact value.

It is important to note that, although arithmetic is done in Gauss-Lobatto form, this is not the same as simply evaluating the program at a grid of collocation points. A set of values at collocation points makes no claim about the values in-between the collocation points. A DeSelby polynomial, on the other hand, defines the value at every point in the domain. In this way, bounds can be put on the error in the final approximation. In addition, the nature and ordering of the DeSelby basis functions implies a very special set of collocation points that would have non-trivial cardinal functions. This amounts to a type of adaptive mesh refinement which can substantially reduce the number of points in comparison to a grid of collocation points, especially in multivariate domains. For example, a 9^{th} order accurate approximation of a function in 5 variables can be represented in a little over 2,000 DeSelby coefficients, whereas a 5-dimensional grid of 9 collocation points along each side would contain just over 59,000 points.

5.3 DeSelby bounds

The DeSelby polynomials extend naturally to the DeSelby bounds (P, ϵ) where P is a DeSelby polynomial and ϵ is an associated bound on error, in the same way as Chebyshev bounds. These obey the same rules of algebra as described in section 4.2. However, it remains to define a way of bounding DeSelby Polynomials and multiplying them together.

5.3.1 Bounding DeSelby polynomials

Two algorithms were devised that bound a DeSelby polynomial in $O(N)$ operations where N is the number of coefficients. These are described in sections 5.6.4 and 5.6.5.

Many algorithms exist that give tighter bounds on polynomials than these algorithms, but all have computational complexities worse than $O(N)$. See, for example, Cornelius and Lohner (1984), Lin and Rokne (1995), Smith (2009).

5.3.2 Addition/Subtraction of DeSelby polynomials

Addition and subtraction of DeSelby polynomials is performed in $O(N)$ operations by the straightforward process of adding and subtracting bases respectively.

5.3.3 Multiplication of DeSelby polynomials

An algorithm was devised that combines the tasks of multiplying two degree N polynomials, truncating the result and calculating bounds on the truncation error. The algorithm completes in $O(N)$ operations. As mentioned earlier the multiplication itself is done in the Gauss-Lobatto representation, this is not new. However, Gauss-Lobatto multiplication alone does not allow bounds to be put on the resulting truncation error. Traditionally, the fastest algorithm that bounds error involves interpolating the multiplicands, performing the multiplication, transforming to Chebyshev form (or other pseudo-spectral form) then truncating the result. The interpolation and subsequent transformation to Chebyshev form would traditionally require algorithms that take $O(Nlog(N))$ time. DeSelby polynomials can be interpolated in $O(N)$ time, so this method could be used to perform fast multiplication. In order to achieve a slightly greater increase in speed, however, we devised an algorithm in which the interpolation need not be done at all. This was achieved by noting that, in the DeSelby representation, the product of a term in the m^{th} shell and a term in the n^{th} shell, where $m \leq n$ has its highest order term in the $(n+1)^{th}$ shell. This means that only multiplications that involve terms from certain combinations of shells can result in truncation error. For example, in the monovariate case, a DeSelby multiplication can be

expressed as the form

$$PQ = (p_{00} + P_l + P_h)(q_{00} + Q_l + Q_h)$$

where P_h and Q_h consist of the terms in the highest shell of P and Q respectively, p_{00} and q_{00} are the first shell coefficients (which are just constants so cannot cause truncation error) and P_l and Q_l are the remainder of terms. By bounding these terms, we can bound the truncation error by

$$B_{\text{trunc}} = B(P_l + P_h)B(Q_h) + B(P_h)B(Q_l + Q_h) \ .$$

This can be done in $O(N)$ operations, without the need for interpolation.

5.4 Testing with a Cloud Resolving Model

A simple, 2 dimensional simulation of dry, turbulent convection was written in C++ in order to test program approximation using DeSelby polynomials. The model was based on that of Klemp and Wilhelmson (1978) with all moisture variables and microphysics removed.

The model was used to simulate dry convection over 30 simulated minutes on a 20x7 grid. 30 minutes was chosen as it is the typical duration of a single timestep of a global model so is relevant to the parameterisation of processes for global models. The domain was horizontally periodic with solid boundaries at the top and bottom.

The model was wrapped so that its inputs were the horizontally averaged temperature perturbations at each vertical level and the outputs were the average heat fluxes across each vertical boundary. When transforming the input, initial velocity was taken to be zero and initial sub-grid perturbations of temperature were taken to be sinusoidal with a period equal to the width of the domain. The average temperature perturbations of each vertical level were defined to be in the range $[0 : 0.06]$K, which is typical of values found during convective events.

Parameterisations showed speed increases of the order of 1000 compared to the high resolution model for 0.1% error compared to the maximum value of each output variable.

5.5 Mathematical development

The basis of the DeSelby polynomials is a set of functions defined on the interval $[-1:1]$ of the form

$$C_j^N(x) = (-1)^{(j+1)} \frac{(1-x)^2}{c_j N^2 (x - \cos(\frac{\pi j}{N}))} \frac{dT_N(x)}{dx}$$

where $T_N(x)$ is the N^{th} degree Chebyshev polynomial and

$$c_j = \begin{cases} 2 & \text{if } |j| = N \\ 1 & \text{if } |j| < N \end{cases}$$

These are known as the 'Chebyshev-Gauss-Lobatto' cardinal functions (see, e.g., Boyd, 2001), they have the important property that C_j^N is zero at the points $x_i = \cos(\frac{\pi i}{N})$, $i \neq j$.

Using these, we define the n^{th} DeSelby basis function as

$$D_n(x) = \begin{cases} C_{2n}^2(x) & \text{if } n < 2 \\ C_{2n+1}^{2^{\lfloor ln_2(n)+1 \rfloor}}(x) & \textit{otherwise} \end{cases}$$

where $\lfloor i \rfloor$ denotes the floor operator which gives the highest integer not larger than i. A univariate DeSelby polynomial is a sum over this basis

$$P(x) = \sum_{n=0}^{N} d_n D_n(x) + d_{00}$$

where d_{00} is an 'extra' coefficient which defines $P(0)$. Multivariate polynomials can be defined on this basis in the normal way.

Theorem 5.5.1. *The product of two DeSelby basis functions $D_n D_m$, where $n \leq m$, is a DeSelby polynomial whose coefficients of degree less than or equal to n are all zero.*

Proof. This can be seen by noting the position of the zero's of the DeSelby basis functions. Since C_j^N is zero at the points $x_i = \cos(\frac{\pi i}{N})$, $i \neq j$, then since the DeSelby basis functions are defined as:

$$D_n(x) = \begin{cases} C_{2n}^2(x) & \text{if } n < 2 \\ C_{2n+1}^{2^{\lfloor ln_2(n)+1 \rfloor}}(x) & \text{otherwise} \end{cases}$$

the set of zero's of the n^{th} DeSelby basis is

$$D_n^0 = \begin{cases} \{x_i : x_i = \cos(\frac{\pi i}{2}) \wedge i \neq 2n\} & \text{if } n < 2 \\ \{x_i : x_i = \cos(\frac{\pi i}{2^{l^{n_2(n)+1}}}) \wedge i \neq 2n+1\} & \text{otherwise.} \end{cases}$$

From this it can be seen that for any DeSelby basis D_r there exists a point at

$$x = \begin{cases} \cos(\frac{\pi 2n}{2^{l^{n_2(r)+1}}}) & \text{if } n < 2 \\ \cos(\frac{\pi 2n+1}{2^{l^{n_2(r)+1}}}) & \text{otherwise} \end{cases}$$

that is not a zero of D_r but is a zero of all D_s where $s > r$. Finally we note that all D_r have a zero at $x = 0$.

If we now express the product of two DeSelby bases as

$$D_n D_m = \sum_r d_r D_r + d_{00}$$

then, starting with the extra coefficient, d_{00}, since all DeSelby basis functions have a zero at $x = 0$, the product $D_n D_m$ must also have a zero at $x = 0$. So, at $x = 0$ the product reduces to $D_n(0)D_m(0) = 0 = d_{00}$. So $d_{00} = 0$. Moving then onto d_0; since all D_r are zero at $x = -1$ when $r > 0$ then so is the product $D_n D_m$ as long as $n > 0$. D_0 is not zero at $x = -1$ so at this point the product reduces to $D_n(-1)D_m(-1) = 0 = d_{00} + d_0 D_0(-1)$. Since $d_{00} = 0$ then $d_0 = 0$. This process of induction can continue at least until we reach the $(n-1)^{th}$ basis function. □

Theorem 5.5.2. *Suppose we have a DeSelby polynomial P whose highest non-zero coefficient is in shell N. We truncate P by removing all terms in shell N, to give a truncated DeSelby polynomial $P_d = P + \epsilon_d$. We also truncate P by expanding in the Chebyshev basis and truncating all Chebyshev terms above 2^{N-1} so that $P_c = P + \epsilon_c$ and P_c and P_d have the same number of coefficients. Then the bound on ϵ_d is twice the bound on ϵ_c. That is, the error introduced by the DeSelby truncation can be bounded by twice the bound due to Chebyshev truncation.*

Proof. When the Gauss-Lobatto cardinal function is expressed as a Chebyshev expansion it is given by (Boyd, 2001)

$$C_j^N(x) = \frac{2}{N p_j} \sum_{m=0}^{N} \frac{1}{p_m} T_m(x_j) T_m(x) = \sum_{m=0}^{N} c_{jm}^N T_m(x) \tag{5.1}$$

where

$$p_i = \begin{cases} 2 & \text{if } i = 0, N \\ 1 & \text{otherwise} \end{cases}$$

since, by definition $x_j = \cos(\frac{\pi j}{N})$ and $T_m(x) = \cos(m\cos^{-1}(x))$ then, by substitution into equation 5.1, the m^{th} degree Chebyshev coefficient is proportional to

$$c_{jm}^N = \frac{2\cos\left(\frac{m\pi j}{N}\right)}{Np_jp_m}$$

letting $m = \frac{N}{2} + n$

$$c_{jm}^N = \frac{2\cos\left(\pi\left(\frac{j}{2} + \frac{nj}{N}\right)\right)}{Np_jp_m} \ .$$

This implies that for any odd j,

$$c_{j(\frac{N}{2}-n)}^N = -c_{j(\frac{N}{2}+n)}^N \ .$$

All DeSelby bases other than those in the first shell have odd j so their Chebyshev coefficients are reflected about their mid-point coefficient. From this, a weighted sum of all DeSelby bases in a shell also has a Chebyshev expansion with coefficients that are reflected about its mid-point. So, if we let the original polynomial, P, have a Chebyshev expansion given by

$$P(x) = \sum_{n=0}^{N} C_n T(x)$$

then

$$\epsilon_d = \sum_{n=0}^{n=\frac{N}{2}} \left(C_{j(\frac{N}{2}-n)}^N T_{\frac{N}{2}-n} - C_{j(\frac{N}{2}-n)}^N T_{\frac{N}{2}+n}\right) \ .$$

The Chebyshev truncation error is bound by the sum of the absolute values of the coefficients above $\frac{N}{2}$ so

$$B(\epsilon_c) = \sum_{n=\frac{N}{2}+1}^{N} |C_n| \ .$$

DeSelby truncation error is bounded by the sum of all coefficients in the truncated shell

$$B(\epsilon_d) = \sum_{m=0}^{\frac{N}{2}} |C_n| + |C_n| \ .$$

Since $c_{\frac{N}{2}} = 0$

$$B(\epsilon_d) = 2B(\epsilon_c) \ .$$

□

5.6 Algorithms

5.6.1 O(N) Gauss-Lobatto/DeSelby conversion

A Gauss-Lobatto representation can be converted to a DeSelby representation with less than 0.5% error and with bounds on the error by using the following algorithm.

Given a polynomial, P, that passes through points $y_0...y_N$ at the Gauss-Lobatto points $x_0...x_N$ such that

$$x_i = \cos\left(\frac{\pi i}{N}\right)$$

it is well known (e.g. Mason and Handscombe, 2003) that under the transformation $x' = \cos^{-1}(x)$, $P(x')$ can be expressed as the discrete cosine transform of the points $y_0...y_N$. It follows that $P(x')$ can also be expressed as the discrete Fourier transform of $y_N...y_1, y_0, y_1...y_N$. The discrete Fourier transform can be written in the form

$$F(x) = \frac{1}{2N}\sum_{j=-N}^{N} y_j \sin(N(x - x_j)) \cot\left(\frac{x - x_j}{2}\right) \tag{5.2}$$

where

$$x_j = \frac{\pi j}{N} \ .$$

We define the interpolation points

$$x'_i = \frac{\pi\left(i + \frac{1}{2}\right)}{N}, \ -N \leq i < N \ .$$

At these points, from equation 5.2,

$$F(x'_i) = \frac{1}{2N}\sum_{j=-N}^{N} y_j (-1)^{(i-j)} \cot\left(\frac{x'_i - x_j}{2}\right) \ . \tag{5.3}$$

To calculate this sum explicitly for all interpolation points would take $O(N^2)$ operations. However, this can be reduced to $O(N)$ by approximating the cot function with an approximant of the form

$$\cot(x) \approx \begin{cases} A_1 e^{-k_1 x} + A_2 e^{-k_2 x} & \text{if } 0 < x < \pi \\ -A_1 e^{k_1 x} - A_2 e^{k_2 x} & \text{if } -\pi \leq x < 0 \end{cases}$$

($\cot(x)$ is not defined at $x = 0$). Using this approximation, equation 5.3 can be expressed as

$$\begin{aligned} F(x'_i) \approx \quad & \tfrac{1}{2N} \Big(\\ & - \textstyle\sum_{j=i-N}^{i-1} y_j (-1)^{(i-j)} A_1 e^{k_1(x_j - x'_i)} - \sum_{j=i-N}^{i-1} y_j (-1)^{(i-j)} A_2 e^{k_2(x_j - x'_i)} \\ & + \textstyle\sum_{j=i}^{i+N-1} y_j (-1)^{(i-j)} A_1 e^{k_1(x'_i - x_j)} + \sum_{j=i}^{i+N-1} y_j (-1)^{(i-j)} A_2 e^{k_2(x'_i - x_j)} \Big) \end{aligned}$$

where $y_j = y_{(2N-j)}$ when $j > N$.

If we let

$$S_n^+(i) = \sum_{j=i+1}^{i+N} y_j (-1)^{(i-j)} A_n e^{k_n(x'_i - x_j)}$$

and

$$S_n^-(i) = - \sum_{j=i-N+1}^{i} y_j (-1)^{(i-j)} A_n e^{k_n(x_j - x'_i)}$$

then

$$F(x'_i) \approx \frac{1}{2N} \left(S_1^-(i) + S_1^+(i) + S_2^-(i) + S_2^+(i) \right) \ .$$

The algorithm makes use of the observation that, in the range $0 \leq j \leq N$,

$$S_n^+(i-1) = \sum_{j=i}^{i+N-1} y_j (-1)^{(i-1-j)} A_n e^{k_n(x'_{i-1} - x_j)}$$

$$= - \sum_{j=i+1}^{i+N} y_j (-1)^{(i-j)} A_n e^{k_n(x'_i - x_j)} e^{\frac{-k_n \pi}{N}} + \left(y_i + (-1)^N y_{(i+N)} e^{-k_n \pi} \right) A_n e^{\frac{-k_n \pi}{2N}}$$

$$= -S_n^+(i) e^{\frac{-k_n \pi}{N}} + \left(y_i + (-1)^N y_{(i+N)} e^{-k_n \pi} \right) A_n e^{\frac{-k_n \pi}{2N}}$$

and, similarly

$$S_n^-(i+1) = \sum_{j=i-N+1}^{i} y_j (-1)^{(i-j)} A_n e^{-k_n(x_j - x'_i)} e^{\frac{-k_n \pi}{N}} + \left(y_{i+1} + (-1)^N y_{(i-N+1)} e^{-k_n \pi} \right) A_n e^{\frac{-k_n \pi}{2N}}$$

$$= -S_n^-(i) e^{\frac{-k_n \pi}{N}} + \left(y_{i+1} + (-1)^N y_{(i-N+1)} e^{-k_n \pi} \right) A_n e^{\frac{-k_n \pi}{2N}} \ .$$

In addition, we observe that

$$S_n^+(N) = S_n^-(N-1)$$

and

$$S_n^+(0) = S_n^-(-1) \ .$$

From these relationships, the complete set of interpolation points can be approximated in $O(N)$ operations by supposing that $S_n^-(-1) = 0$, sweeping forward

from y_0 to y_N, calculating the values of $S_n^-(n)$ as we go, then sweeping backward from y_N to y_0, calculating $S_n^+(n)$ as we go then finally sweeping forward again to correct for the initial assumption of $S_n^-(-1) = 0$.

This algorithm can be used to calculate the DeSelby coefficients from the Gauss-Lobatto coefficients by starting at the lowest DeSelby shell and working to successively higher shells, calculating the values at the interpolation points and subtracting from the Gauss-Lobatto points to find the perturbations in each shell.

It was found that by optimising the parameters A_1, A_2, k_1 and k_2 so as to minimise the error at the points evaluated in equation 5.3, the error in the approximation could be bounded to around 0.5% of the final value. For lower degree shells, this level of accuracy could be achieved with only one exponential. By placing bounds on the error of our approximation of cot, bounds on the error of a given transformation can be calculated by error tracking in the usual way.

5.6.2 O(N) differentiation

A DeSelby polynomial can be differentiated with respect to a variable using the following algorithm.

The rate of change of a Gauss-Lobatto cardinal function at the i^{th} Gauss-Lobatto point is given by

$$\left.\frac{dC_j^N}{dx}\right\|_{x_i} = \begin{cases} 0 & \text{if } i = j \\ 0.5(-1)^{(i-j)} \frac{\cot\left(\frac{\pi(i-j)}{2}\right)}{\sqrt{1-x_i^2}} & \text{otherwise.} \end{cases}$$

So, for some DeSelby polynomial $F(x)$ of degree N

$$F'(x_i) = \frac{1}{2\sqrt{1-x_i^2}} \sum_{j=0, j\neq i}^{N} y_j (-1)^{(i-j)} \cot\left(\frac{\pi(i-j)}{2}\right)$$

but the sum in this equation is of exactly the same form as equation 5.3 which describes interpolation. So, exactly the same type of algorithm can be used.

If the differentiation is performed using the same values of k_n and A_n as used during interpolation, the rate of change and interpolated values could both be calculated from the same calculation, saving some operations if both are required. However, because the cot term is evaluated at different points during differentiation than those used during interpolation, slightly greater accuracy

can be achieved by using values of k_n and A_n that are optimised for these points.

5.6.3 DeSelby/Gauss-Lobatto to Chebyshev conversion

Converting from DeSelby to Chebyshev basis is easily performed in $O(NlogN)$ operations by transforming each DeSelby shell using the a fast cosine transform, then summing the results.

To convert a polynomial from Gauss-Lobatto form to Chebyshev form one could convert first to DeSelby form, then to Chebyshev. Alternatively, an algorithm was developed that converts directly from the Gauss-Lobatto form to Chebyshev form in $O(Nlog(N))$ operations. The algorithm makes use of a natural generalisation of Smolyak's algorithm (Smolyak, 1963; Wasilkowski and Wozniakowski, 1995) to allow for the shell structure of the DeSelby polynomial.

Transforming from Gauss-Lobatto to Chebyshev is easy in the case when the grid points form a Cartesian grid; one would just use a fast cosine transform. However, a difficulty arises because, in the multivariate case, the shell structure of the DeSelby basis does not generally lead to a Cartesian grid. However, this grid structure can be described as the union of a number of Cartesian grids (i.e. a number of Cartesian grids superimposed on each-other), and the Chebyshev form can be built up out of the fast cosine transforms of these Cartesian grids.

Let each shell of a d-variate polynomial be identified by a d-dimensional vector, i. Each element of i identifies the degree of the shell in one variable by numbering the shells consecutively in order of increasing degree. Each shell can then be thought of as belonging to the Cartesian grid formed from itself and all shells whose degree is lower than or equal to itself in all variables. The new development is the concept of the 'valency' of a shell, i, in a polynomial, P, which we define as:

$$v(i) = \sum_{a \in \{0,1\}^d, s(i+a) \in P} (-1)^{|a|}$$

where $s(n) \in P$ if and only if P contains a shell with identity n, and $|a|$ is the sum of the elements of a (i.e. its 1-norm). This is a natural generalisation of the multiplier in Smolyak's algorithm for a polynomial of total degree j, which

is defined as:

$$b(j) = \sum_{a \in \{0,1\}^d, |a| \leq j} (-1)^{|a|} \ .$$

If we let C_i be the polynomial formed from the Cartesian grid to which shell i belongs, then we have the relation

$$P = \sum_{s(i) \in P} v(i) C_i$$

the proof of this follows naturally from that for Smolyak's algorithm (Smolyak, 1963) and is not given here.

By the nature of the discrete cosine transform, the C_i's can be inductively built up from each other, starting with the lowest degree, allowing the sum to be calculated in $O(Nlog(N))$ time.

5.6.4 Bounding a polynomial

An algorithm was devised to bound a DeSelby polynomial above and below in $O(N)$ operations with similar tightness as obtained by the summing of Chebyshev coefficients.

The algorithm works by bounding each DeSelby shell. The polynomial is then bounded by the interval-sum of the bounds of the shells. Each shell can be bounded by removing variables one at a time. Consider first a monovariate polynomial, $P(x)$. The value at any given point, $x = \cos^{-1}(x')$, is equal to

$$s(x') = \sum_{j=-N/2}^{N/2-1} d_{\|j\|} \frac{1}{2N} \sin(N(x' - x_{2j+1})) \cot(0.5(x' - x_{2j+1}))$$

where d_j are the DeSelby coefficients and $x_j = \frac{\pi j}{N}$. Since

$$\sin(N(x' - x_{2j+1})) = -\sin(Nx')$$

then

$$\begin{aligned} s(x') < \quad & \overline{d} \sum_{j=-\frac{N}{2}}^{\left\lfloor \frac{Nx'}{2\pi} \right\rfloor - 1} \frac{\sin(N(x' - x_{2j+1})) \cot(0.5(x' - x_{2j+1}))}{2N} \\ & -\underline{d} \sum_{j=\left\lceil \frac{Nx'}{2\pi} \right\rceil - 1}^{\frac{N}{2}-1} \frac{\sin(N(x' - x_{2j+1})) \cot(0.5(x' - x_{2j+1}))}{2N} \end{aligned}$$

where $\overline{d}$ and $\underline{d}$ denote the maximum and minimum DeSelby coefficient. Letting $\bar{D} = \frac{\overline{d}+\underline{d}}{2}$ and $\Delta = \frac{\overline{d}-\underline{d}}{2}$

$$s(x') < \bar{D} + \Delta \left(\sum_{j=-N/2}^{\left\lfloor \frac{Nx'}{2\pi} \right\rfloor - 1} \frac{\sin(N(x' - x_{2j+1}))\cot(0.5(x' - x_{2j+1}))}{2N} - \sum_{j=\left\lceil \frac{Nx'}{2\pi} \right\rceil - 1}^{N/2-1} \frac{\sin(N(x' - x_{2j+1}))\cot(0.5(x' - x_{2j+1}))}{2N} \right) .$$

The sums are now independent of the DeSelby coefficients so their maximum value can be calculated off line and stored in a lookup table for all degrees of polynomial we are likely to encounter. If we denote this as M_N then

$$s(x') < \bar{D} + \Delta M_N \ .$$

An analogous calculation gives the minimum bound.

The same treatment extends naturally to multivariate shells by bounding one dimension at a time using the relation

$$P(x_0, ..., x_n) = \sum_{j_0,...,j_n} d_{(\|j_0\|,...,\|j_n\|)} \prod_{m=0}^{n} C_{j_m}(x_m)$$

$$= \sum_{j_0,...,j_{n-1}} d'_{(\|j_0\|,...,\|j_{n-1}\|)} \prod_{m=0}^{n-1} C_{j_m}(x_m)$$

where

$$d'_{(\|j_0\|,...,\|j_{n-1}\|)} = \sum_{j_n} d_{(\|j_0\|,...,\|j_n\|)} C_{j_n}(x_n) \ .$$

Solving this gives bounds

$$s(x') < \bar{D} + \Delta M_N^m$$

where m is the number of variables.

5.6.5 Bounding a polynomial (alternative method)

A tighter way of bounding a polynomial in $O(N)$ operations was discovered, but not implemented. Begin by considering the univariate case, $P(x)$. Under the transformation $x' = \cos(x)$, the value at any point is given by

$$P(x') = \sum_{j=-N}^{N} \frac{d_{|j|}}{2N} \sin(N(x' - x_j))\cot(0.5(x' - x_j))$$

but if we let $x' = x_k + \Delta x$ then

$$P(x') = \sum_j \frac{d_{|j|}}{2N} \sin(N(\Delta x + x_k - x_j)) \cot(0.5(\Delta x + x_k - x_j))$$

but $N(x_k - x_j)$ is always a multiple of π so

$$P(x') = \sin(N\Delta x) \sum_j \frac{d_{|j|}}{2N} (-1)^{j-k} \cot(0.5(\Delta x + x_k - x_j))$$

applying the same approximation of cot as in section 5.6.1 allows the Δx to be taken outside the sum

$$P(x') \approx \sin(N\Delta x)(A_1 e^{k1\Delta x} + A_2 e^{k2\Delta x}) sgn(\Delta x) \sum_j \frac{d_j}{2N} (-1)^{j-k} \cot(0.5(x_k - x_j))$$

where $sgn(\Delta x)$ is 1 if Δx is positive and -1 if Δx is negative. The sum in the above equation can be calculated for all k in $O(N)$ operations using the same method as used to differentiate polynomials described in 5.6.2. The term when $j = k$ contains a singularity at $\Delta x = 0$ and the approximation loses accuracy, so this term is treated separately and bound above and below in the region $-\frac{\pi}{2N} \leq \Delta x \leq \frac{\pi}{2N}$ by

$$1 - \frac{2N(1 - y_{0.5})}{\pi} \Delta x \leq \frac{1}{2N} \sin(N\Delta x) \cot(0.5\Delta x) \leq 1 - \frac{4N^2(1 - y_{0.5})}{\pi^2} \Delta x^2$$

where

$$y_{0.5} = \frac{\cot\left(\frac{\pi}{4N}\right)}{2N} \ .$$

The sum of other terms can be bounded using the inequalities

$$\frac{2N\Delta x}{\pi} \leq \sin(N\Delta x) \leq \frac{4N\Delta x}{\pi} - \frac{4N^2\Delta x^2}{\pi^2}$$

in the region $0 \leq \Delta x \leq \frac{\pi}{2N}$ and

$$\frac{4N\Delta x}{\pi} + \frac{4N^2\Delta x^2}{\pi^2} \leq \sin(N\Delta x) \leq \frac{2N\Delta x}{\pi}$$

in the region $-\frac{\pi}{2N} \leq \Delta x \leq 0$. The exponential terms are bounded above by A and below by $Ae^{-k\frac{\pi}{2N}}$. In this way, the region is bounded by quadratics whose maxima/minima can be found immediately.

This can be extended to the bounding of multivariate shells by bounding one dimension at a time and using interval arithmetic to calculate the bounds-on-bounds. It can also be extended to the bounding of a complete multivariate

polynomial directly, rather than by bounding shells individually. However, dealing with multiple variables is a little more complicated in this case since we need to deal with the situation when the DeSelby shells describe a set of collocation points that do not lie on a square grid. In this case, bounds must additionally be put on the higher shells individually, and these bounds must be added to the shells which do not contain these higher orders.

Chapter 6

Entrainment in marine stratocumulus

6.1 Introduction

iGen was used to analyse a simulation of a non precipitating, stratocumulus topped, well mixed boundary layer (STBL) overlying a sea surface. Climatological observations (e.g. McDonald, 1938; Hartmann and Short, 1980) have shown that this regime of "marine stratocumulus" is persistently found over large areas of the ocean where there is large scale subsidence. Because of the large area of coverage and the very different radiative properties of a stratocumulus covered ocean compared to an exposed sea surface, marine stratocumulus has an important role to play in the Earth's radiative equilibrium. Understanding how marine stratocumulus reacts to climate forcings, then, is crucial to understanding climate change. Bony and Dufrence (2005) showed that there was great disagreement between climate models in their estimation of radiative forcing due to marine stratocumulus under increased sea surface temperature. They also showed that it is in the simulation of marine stratocumulus extent that climate models differ most when compared to present day observations. A more recent study (Dufrence and Bony, 2008) shows that this situation has not improved with time.

The large scale structure and dynamics of this regime has been described by

Lilly (1968). Typically, there is a well mixed layer from sea surface up to cloud top where, due to strong turbulent mixing, the total water and liquid potential temperature is close to homogeneous. The well mixed layer is capped at cloud top by a strong inversion leading into a much warmer and dryer free atmosphere. The turbulence is driven partly by surface fluxes of heat and moisture but predominantly by strong radiative cooling at cloud top and, to a lesser extent, by radiative warming at cloud base due to the temperature difference between the cloud base and the sea surface. This turbulence causes some of the warmer, drier free-atmosphere air to be mixed, or 'entrained' into the boundary layer. Given the rate of this entrainment, the large scale dynamics of the system is easily calculated from budgets of mass, energy and moisture. However, no analytic derivation of this entrainment rate has been found. Lilly (1968) derives upper and lower bounds and Stevens (2002) gives details of various parameterisations.

It was proposed to use iGen to analyse a cloud resolving model in order to derive a fast, approximate way to calculate entrainment from the large scale state of the STBL, thereby closing the large-scale equations of motion.

6.2 A Cloud Resolving Model for stratocumulus

A simple, 2-dimensional cloud resolving model was written in C++ in order to simulate entrainment in stratocumulus under nocturnal, rain-free conditions. A new cloud resolving model was written, rather than using existing code, for two reasons: firstly, iGen can at present only analyse C++ programs, while the existing models available to the author are written in Fortran; secondly, writing a new model gave us much more freedom to test iGen to see how it performed with different schemes and algorithms. The model was based on that of Klemp and Wilhelmson (1978) with modifications detailed in Skamrock and Klemp (1994).

It was decided to write a 2-dimensional model, rather than a 3-dimensional model, so that simulations and analyses could be performed in a reasonable time on a desktop computer, as the project did not have funding for supercomputer time. This remains a valid test of the analysis techniques as one would expect something like two orders of magnitude increase in processing power on a supercomputer compared to a desktop. This means that what can be done

on a desktop computer in 2-dimensions can be done on a supercomputer in 3-dimensions in around the same amount of time.

A number of changes were made to the Klemp and Wilhelmson (hereafter KW) model to better suit our needs. It was found that the second-order finite difference vertical advection scheme described in KW did not cope well with the steep gradients at cloud top. This caused 'ringing' effects which led to unrealistic cooling below cloudtop and heating above. To deal with this, a flux limiting advection scheme was used instead (Nikiforakis, 2007). This calculated advection as a mix between a fourth order, centred finite difference scheme and an upstream scheme. The flux limiting function used was

$$\phi(r) = \begin{cases} 0 & \text{if } r < 0 \\ 2r & \text{if } 0 \leq r \leq \frac{1}{2} \\ 1 & \text{otherwise.} \end{cases}$$

Other changes are as follows:

- A more accurate version of Teten's formula was used (Emmanuel, 1994).
- Temperature was stored as liquid water potential temperature.
- Liquid water was stored as total specific water content, cloud being diagnosed when this exceeds saturation.
- In order to simulate longwave radiative heating/cooling, the radiation scheme described in Larson et.al (2007) was added.
- Prognostic variable and equation for rain was removed.
- Surface fluxes of heat and moisture as a function of velocity were added.

The full set of equations are given in Appendix B.

6.2.1 Testing the model against observation

The model was compared against observations and other cloud resolving models by performing a simulation of the first research flight of the second "dynamics and chemistry of marine stratocumulus" field study (DYCOMS-II). This case was chosen as it has been used in an intercomparison study of large eddy models (Stevens et.al., 2005). As part of this study, a detailed specification of an

idealised simulation was given, and results from an ensemble of large eddy models from ten different modelling centres are available, allowing our model to be compared against these commonly used models.

Our model showed a longer spin-up period than the models in the intercomparison (figure 6.1) and this was attributed to the 2-dimensional turbulence of the model, compared to the 3-dimensional turbulence of the models in the intercomparison. The cascade of turbulent kinetic energy and vorticity is known to be different in 2 and 3 dimensions (Kraichnan, 1967). During this spin-up period, the low turbulent kinetic energy led to low entrainment and so the prescribed large scale subsidence caused the cloudtop to descend. In order to account for this descent during the spin-up period, the initial cloudtop height was raised by 10m, this had the effect of bringing the cloudtop height in-line with the other models at 2-hours into the simulation when the spin-up period was over.

From 2-hours into the simulation to the end of the simulation the model was in good agreement with both observation and the models of the intercomparison. Cloudtop height, and therefore entrainment, was very close to the ensemble average (see figure 6.2). Cloudbase height was also very close to the ensemble average (see figure 6.3).

6.2.2 Wrapping the CRM to calculate entrainment

The cloud resolving model was wrapped so that it calculated the mean and standard deviation of the entrainment velocity for a given specification of the large scale state. The large scale state could be specified using the variables

- Temperature jump across cloudtop
- Jump in q_t at cloudtop
- Height of cloudtop
- Down-welling radiation just above cloudtop
- Average boundary layer liquid water potential temperature
- Average boundary layer q_t
- Sea surface temperature

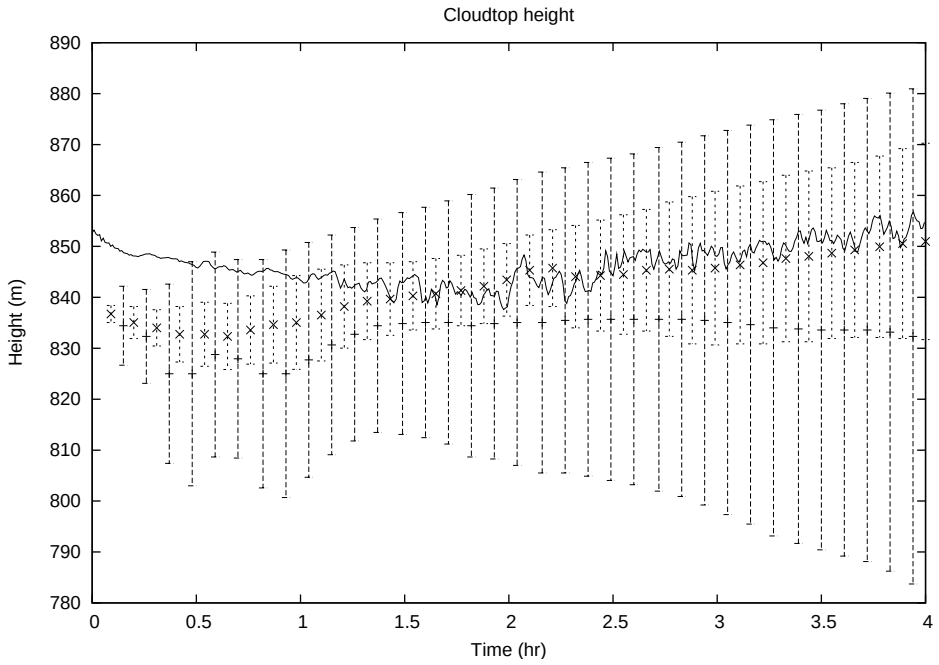

Figure 6.1: Cloudtop height of DYCOMS-II simulation: Solid line shows results from our 2D CRM. The inner error bars show the first and third quartiles of the ensemble of models in the Stevens et.al. (2005) intercomparison, the outer error bars show the maximum and minimum values of the ensemble. The mid-points of the error bars are marked by crosses and plus signs respectively.

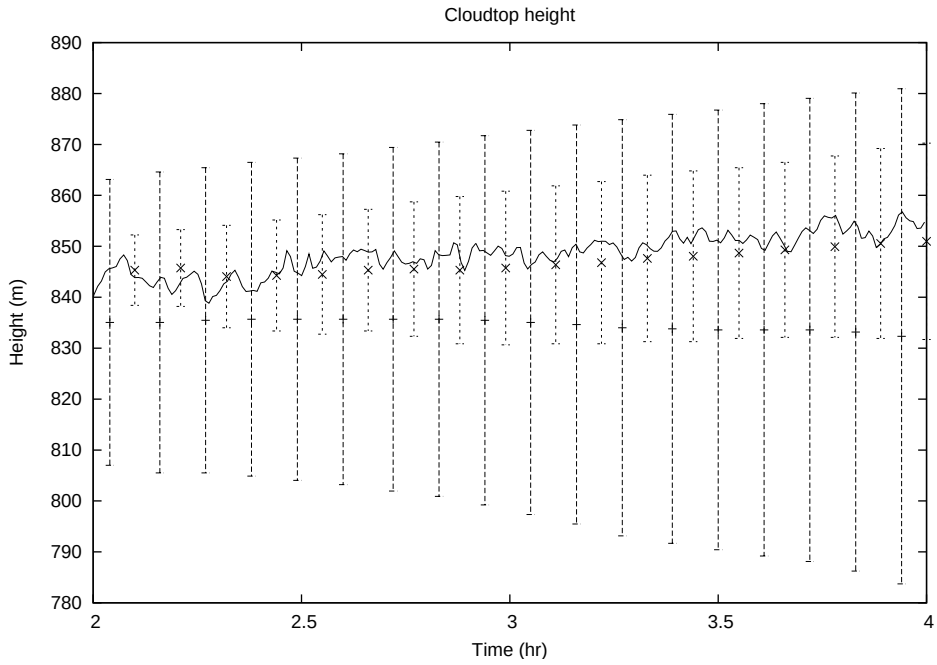

Figure 6.2: DYCOMS-II simulation: Cloudtop height from two hours into the simulation. The solid line shows the results from the 2D CRM. Inner error bars show the first and third quartiles of the ensemble of intercomparison models, the outer error bars shows the maximum and minimum values of the ensemble.

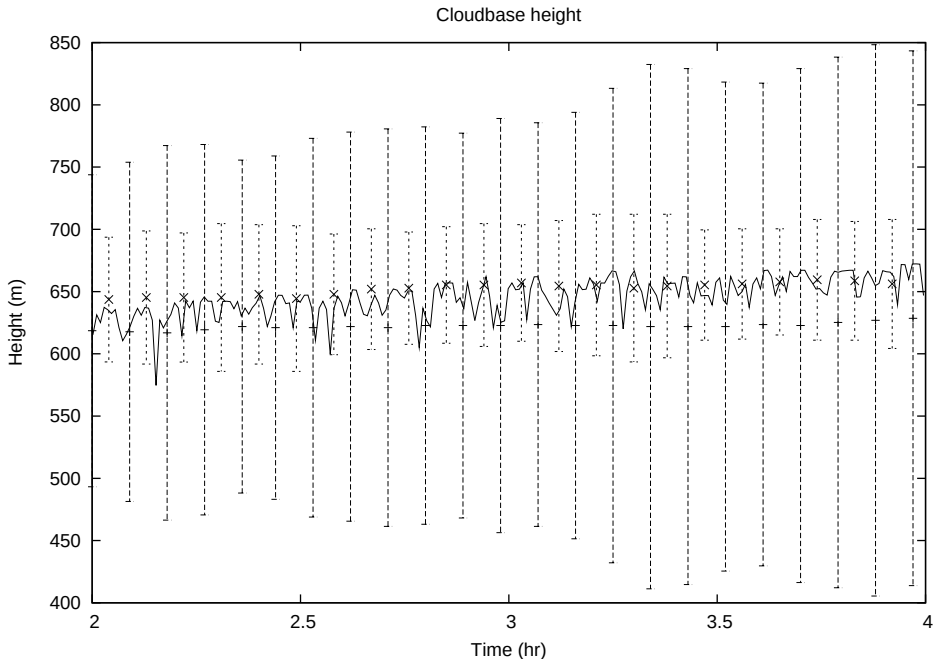

Figure 6.3: DYCOMS-II simulation: Cloudbase height from two hours into the simulation. The solid line shows the results from the 2D CRM. Inner error bars show the first and third quartiles of the ensemble of intercomparison models, the outer error bars shows the maximum and minimum values of the ensemble.

where q_t is the specific total water content. However, by transforming the variables to the set

q_{lct} Specific liquid water content at cloud top

Δq_t Jump in specific total water at cloud top

ΔB Jump in Buoyancy at cloud top

F_0 Down-welling radiation just above cloud top divided by average boundary layer temperature

F_1 Up-welling radiation just below cloud base divided by average boundary layer temperature

θ_{lbl} Average boundary layer liquid water potential temperature

ΔT_{sst} Difference between sea surface temperature and boundary layer temperature

the dependency on boundary layer temperature was shown to be very weak over the range of values we expect to experience.

In order to find the mean and standard deviation of entrainment rate for some large scale state, X, we follow our abstraction theory and average over the high resolution states γX. Calculating this explicitly would involve finding the prior probabilities of the high-resolution states and writing a program to calculate γ as described in chapter 2. The analysis would then average over the seed to the random number generator used during the calculation of γ. However, if we assume that the system is ergodic then the moments of the instantaneous entrainment averaged over all random seeds are equal to the moments of a single simulation, averaged over a sufficiently long period of time, where the large scale state is held constant.

In order to keep the large scale state constant, a set of fluxes were calculated at 12 second intervals and added at each timestep. The boundary layer height was kept constant by adding a homogeneous, large-scale divergence. This was calculated according to:

$$\nabla \cdot v = \frac{m_d + \frac{h-H}{5\Delta t}}{H}$$

where m_d is the gradient of the least squares linear fit to the total entrainment over the duration of the simulation so far, h is the measured height of the

boundary layer, Δt is the time between updates (12 seconds) and H is the required height. The height of the boundary layer was defined to be the average height of the isoline of total water content half way between the large-scale boundary layer and free atmosphere values.

In order to keep boundary layer temperature and moisture constant a total water flux and temperature flux was added. Total water flux was added to the sub-cloud portion of the boundary layer. This included a flux that tended to homogenise the field and was calculated at each gridpoint as

$$\frac{\partial q_t}{\partial t} = m_q + \frac{q_{tbl} - q_t}{8\Delta t}$$

where m_q is the gradient of the least squares linear fit of the total flux from the beginning of the simulation, q_{tbl} is the large-scale total water in the boundary layer and q_t is the field of actual total water. The homogenisation is not physical but is justified on the grounds that we want to find a formula for entrainment in order to close the large scale dynamics of the boundary layer. However, the large scale dynamics is only valid under the assumption of a homogeneous boundary layer so we are merely enforcing the assumption made by the large scale dynamical view. In terms of abstraction theory, this can be viewed as skewing the value of γX, for some large scale state, X, towards a homogeneous boundary layer; which is implicit in the assumptions of the large scale dynamical view. The homogenisation has the advantage that it reduces the sensitivity of the output to small-scale structure, this will be discussed at greater length in the next section.

The flux of liquid water potential temperature was calculated so as to add a constant buoyancy to the whole boundary layer from the ground up to the isoline of temperature half way between the large-scale boundary layer and free atmosphere values. In this way, the dynamics of the boundary layer is not affected by the flux. The calculation was performed by first calculating a homogeneous buoyancy flux

$$\frac{\partial B}{\partial t} = m_b + \frac{\theta_{lbl} - \bar{\theta_l}}{30\theta_{lbl}\Delta t}$$

where $\bar{\theta_l}$ is the average liquid water temperature between 200m and 100m below cloud top and m_b is the gradient of the least squares linear fit of the total flux of buoyancy since the beginning of the simulation.

The flux of liquid water potential temperature necessary to achieve a given change in buoyancy ΔB over a single timestep, given a change in total water Δq_t, was calculated and added at the end of each timestep. The change in liquid water potential temperature $\Delta\theta_l$ at each gridbox was calculated using the following procedure: In the absence of liquid water

$$\Delta\theta_{l,dry} = \theta_{lbl}\left(\Delta B - 0.61\Delta q_t\right)\Delta t$$

in the presence of liquid water

$$\Delta\theta_{l,wet} = \theta_{lbl}\left(\Delta B - \Delta q_t\left(0.61 + (\frac{\gamma}{\theta_{lbl}} - 1.61)\right)\left(1 - \frac{\partial q_{sat}}{\partial q_t}\right)\right)$$

where $\frac{\partial q_{sat}}{\partial \theta_l}$ is the rate of change of saturation with θ_l at constant q_t, and $\frac{\partial q_{sat}}{\partial q_t}$ is the rate of change of saturation with q_t at constant θ_l. In the case that the flux causes a transition between clear sky and cloud, it is necessary to calculate the fraction of buoyancy and q_t change that occurs in cloud and the fraction in clear sky and to add these contributions separately. When going from clear sky to cloudy, the fraction in clear sky is given by

$$m = \frac{q_{sat} - q_t}{\Delta q_t - \frac{\partial q_{sat}}{\partial \theta_l}\Delta\theta_{l,dry}} \ .$$

When going from cloudy to clear, the fraction in cloudy sky is

$$m = \frac{q_t - q_{sat}}{\left(1.0 - \frac{\partial q_{sat}}{\partial q_t}\right)\Delta qt - \frac{\partial q_{sat}}{\partial \theta_l}\Delta\theta_{l,wet}} \ .$$

The side boundaries of the simulated domain were periodic, the lower boundary was solid (no fluxes across the boundary) and the upper boundary was defined to have no sub-grid turbulent fluxes. Air entered through the top of the domain at the large-scale, free-atmosphere state in order to replace that lost by large scale subsidence. More details of the boundary conditions are given in appendix B.

The initial state of the atmosphere was a homogeneous boundary layer and homogeneous free atmosphere separated by a linear transition of 25m height. Initial velocities were zero everywhere and there was no sub-grid turbulent kinetic energy. Pressure was initialised to the hydrostatic value. In order to break symmetry, a random perturbation of $\pm 0.0025K$ was added to each gridbox below 100m and within 100m below the inversion. Geostrophic winds were not included for the same reason as presented in Moeng et.al. (1996): If we are to

include geostrophic winds, this raises the question of the orientation of the 2D domain in relation to the wind direction. Since roll motions tend to be aligned closely to the wind direction, the natural choice would be perpendicular to the wind direction, meaning no geostrophic wind across the domain.

6.3 The ill conditioning of entrainment

A numerical experiment was performed on the wrapped model to test the sensitivity of entrainment rate to the initial random perturbation of $\pm 0.0025K$ to each gridbox in the lowest 100m of the boundary layer and within 100m below the inversion. The large-scale state was chosen to be around the centre of the expected ranges of each value:

Boundary layer θ_l $= 290$K

Boundary layer q_t $= 8 \times 10^{-3}$Kg Kg^{-1}

$\Delta\theta_l$ **at inversion** $= 8.5$K

Δq_t **at inversion** $= -6 \times 10^{-3}$Kg Kg^{-1}

Net radiation flux above inversion $= -55$W m^{-2}

Net radiation flux at cloud base $= 22$W m^{-2}

$\Delta\theta_l$ **at sea surface** $= 1$K (sea surface warmer).

The domain size was 1166m horizontally and 770m vertically. The inversion height was 600m above the bottom of the domain.

Six simulations were made with random perturbations provided by the C++ `rand()` function, seeded at the beginning of the simulation by the current state of the computer's internal clock. The fluxes of heat and moisture which keep the boundary layer at a constant large scale state were turned off in order to discount them as the source of sensitivity. The resulting total entrainment of the simulations are shown in figure 6.4. After 6 hours there was a 10% spread in total entrainment, showing that there is significant sensitive dependence on initial conditions under these conditions.

Debugging showed no memory leaks or out-of-range references in the program, which could have caused the differing behaviour. Running the simulations with

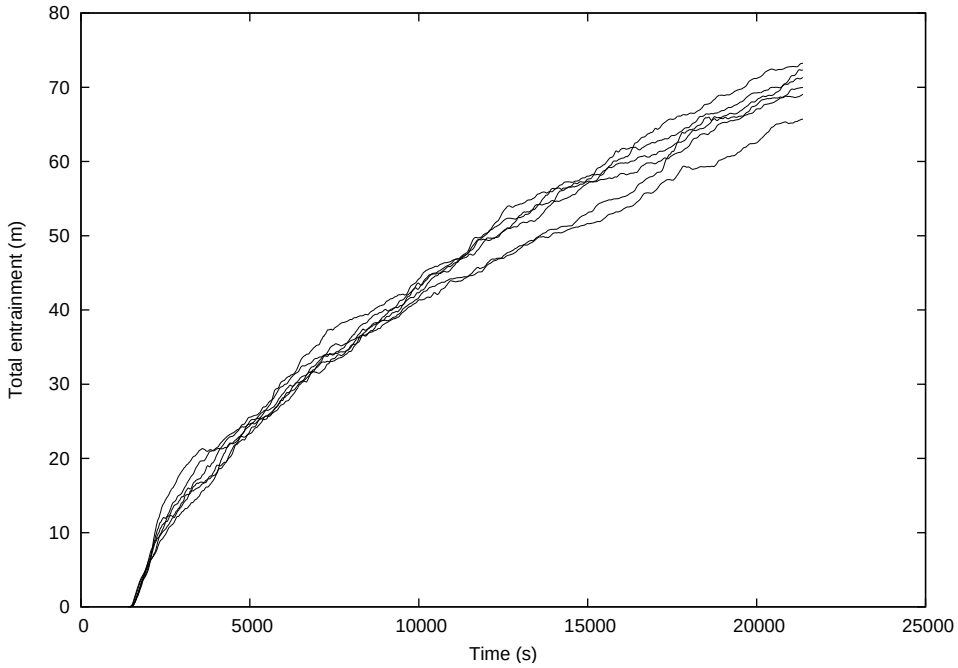

Figure 6.4: Total entrainment against simulated time for six simulations differing only in a 0.0025K perturbation to the initial conditions.

the same random seed at various times and on different computers always returned exactly the same result. Simulations were also made with constant large-scale divergence in order to discount feedbacks with the divergence as the source of sensitivity. Results still showed sensitivity to initial conditions. Different domain geometries did not show any overall reduction in sensitivity. Sensitivity was reduced to around 5% when the large-scale boundary layer state was held constant by turning on the fluxes of heat and moisture.

This sensitivity would explain the large range of results obtained from the ensemble of simulations presented in Stevens et.al. (2005), despite the presence in these simulations of negative feedbacks in the form of a vertical gradient of subsidence velocity and a vertical gradient of temperature in the free atmosphere.

One would expect to see fluctuations in entrainment due to small scale turbulent eddies. However, these should manifest themselves as high frequency noise in the total entrainment. The fluctuation due to this noise, as a percentage of the total entrainment, should reduce in proportion to $\frac{1}{sqrt(N)}$ where N is the number of small-scale turbulent entrainment events that have occurred. For small scale

turbulence, N should be very high when averaged over 6 hours. This suggests that either the amplitude of the small scale fluctuations is very high compared to the average, or there are lower frequency fluctuations coming from larger-scale, longer-lived events.

What we require of an analysis of entrainment in terms of the large scale state is a value of entrainment averaged over a typical timestep of a large scale model. This would be of the order of 10s of minutes. However, the large scale state simply does not contain enough information to say what rate of entrainment one should expect when averaged over this length of time. Our initial question was shown to be ill-posed.

In light of this, it is imperative to render the definition of entrainment into a form that is well conditioned. It is clear that entrainment cannot be modelled as a simple deterministic function of large scale state. The way to proceed comes naturally out of our theory of abstraction. We have an abstract, large-scale model of entrainment, $\alpha f \gamma$, made out of a wrapped, high-resolution model. The output of this is a distribution over possible entrainment values. In this case, the distribution is quite wide, so if we want to model it we must do so with a stochastic model, for which we need to know the moments of the output. At each timestep of the large scale model, an entrainment value is chosen at random from a distribution given by the moments. Mathematically, this is known as a random walk (see e.g. Weiss, 1994). A random walk consists of a number of steps (i.e. timesteps of the large-scale model), and at each step the entrainment is chosen at random from a fixed probability distribution.

A 15 hour simulation was made with the same large scale state as above, but this time with all fluxes turned on. The resulting total entrainment was recorded at 12 second intervals. A histogram of the amount of entrainment in each 12 second interval is shown in figure 6.5.

This clearly shows a Gaussian distribution, as would be expected from the central limit theorem if the entrainment is made up of the action of a large number of random turbulent events. This means that a random walk should be a good model of the long term behaviour of entrainment. Figure 6.6 shows six simulated random walks. Each step spanned 80 seconds and had a Gaussian distribution. The plot is presented for comparison to figure 6.4.

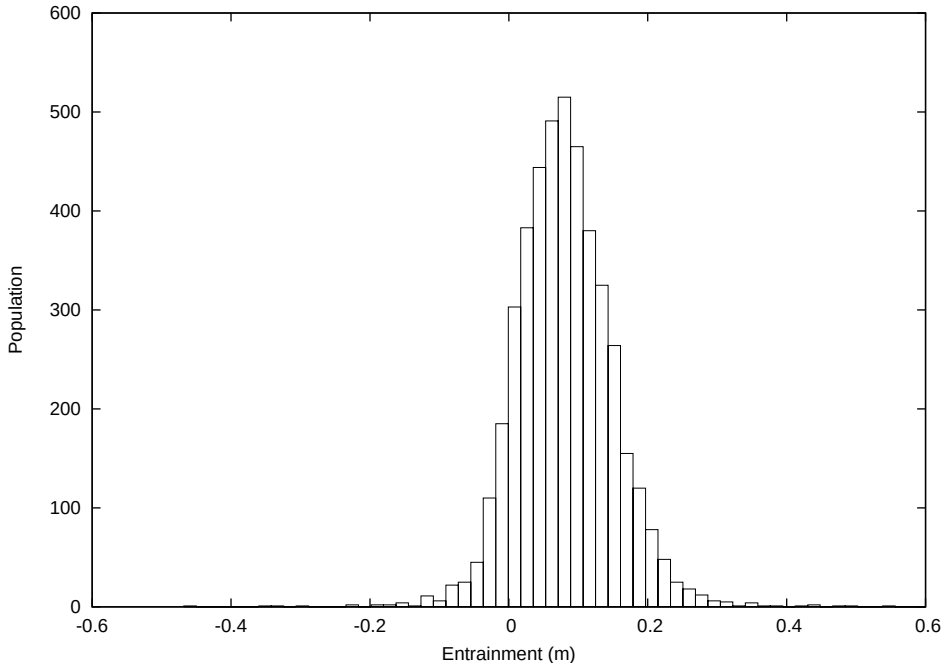

Figure 6.5: Histogram showing the amount of entrainment in 12 seconds. Sampled over a 15 hour simulation.

In order to re-pose the problem into a well conditioned form, then, we assume that entrainment is caused by the action of turbulent eddies and that these can be modelled on the large-scale as a random walk with a Gaussian distribution. Our aim is to find the mean and standard deviation of the step of this random walk. Accordingly, the output of the wrapped model was made to be the mean entrainment rate and the mean of the entrainment rate squared, with samples taken every 12 simulated seconds. This assumption implies that, in the large-scale model, the fluctuations in the entrainment of neighbouring gridboxes are uncorrelated; or at the very least that any correlations do not affect the large scale dynamics. This is plausible, but future study may show that this is not the case. For example, it may turn out that entrainment is caused by some ordered structure or wave that travels between gridboxes. Further experiment would be necessary to discount this possibility.

It is interesting to note that the Gaussian nature of this distribution, when combined with the fact that entrainment predominantly occurs in only one direction (from the free atmosphere into the boundary layer) allows us to calculate

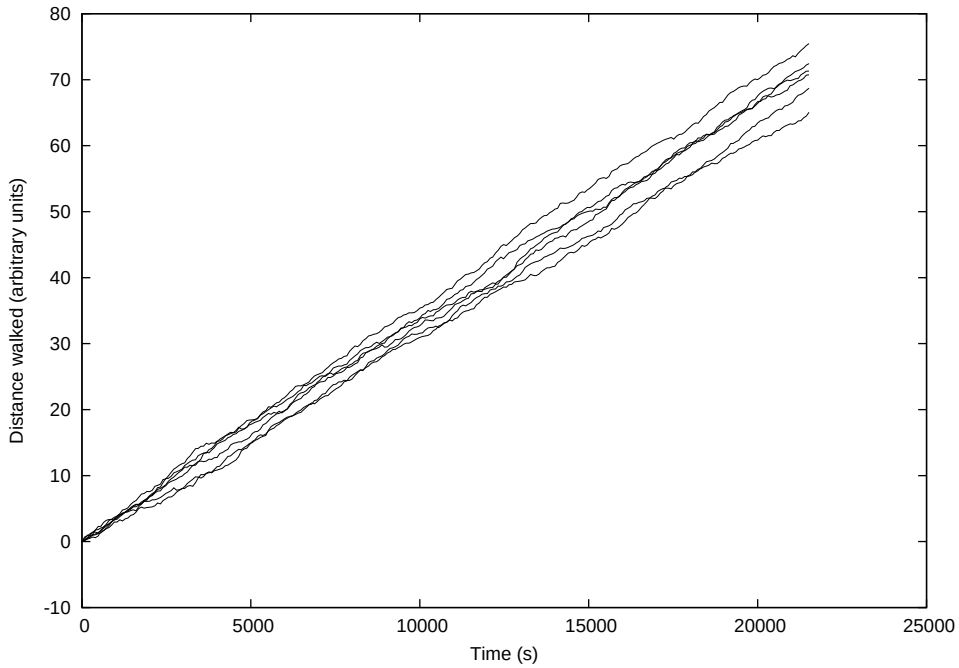

Figure 6.6: Six random walk simulations. Each step spanned 80 seconds and had a Gaussian distribution. Units were chosen to give a walk of the same magnitude as the entrainment shown in figure 6.4

an order of magnitude lower bound on the size/timescale of the process that dominates entrainment. Since entrainment only goes one way, the steps of the random walk must predominantly go forward. For this to be the case, the mean must be larger than the standard deviation, so the time scale of the process of entrainment must be no smaller than that where the standard deviation of each step equals its mean. By fitting a Gaussian to figure 6.5 we get a mean of $0.08ms^{-1}$ and a variance of $0.008m^2s^{-2}$ so the mean would equal the standard deviation at a step of $\frac{0.008\times12}{0.08^2} = 15s$. If we suppose a characteristic velocity of $1ms^{-1}$ this gives a characteristic size of $15m$.

6.4 Analysing entrainment

Since, in a random walk, the standard deviation of the velocity scales as $T^{-\frac{1}{2}}$, where T is the duration of the walk, we express standard deviation as that when averaged over a 30 minute period. This can be converted to any averaging period, T, by multiplying by $\sqrt{\frac{1800}{T}}$, where T is in seconds.

Because of the essentially Gaussian nature of entrainment, formal bounds on the mean and standard deviation are neither appropriate or useful in this case. Suppose, for example, we are given a sample of output from a random process with a Gaussian distribution. Even though we can calculate the *statistical* mean and standard deviation of the sample, no formal bounds can be put on the mean or standard deviation of the process. Although it is very unlikely that the mean and standard deviation of the process is very far from the statistical values, there remains a finite probability that the process could have any values we care to mention, and so this possibility cannot be formally discounted. In the case of entrainment, then, the appropriate bound is the standard deviation of the mean, defined as $\sqrt{(\frac{\sigma}{N})}$ where σ is the variance between samples of entrainment of the wrapped model and N is the number of samples used to form the mean. This gives us a measure of uncertainty in the mean entrainment due to the finite number of samples over which we average. It was found that a simulation of 6 hours with 2 hours spin-up gave a standard deviation of the mean around 2.5%. In any case, if the resulting model of entrainment is to be used with a timestep of a typical GCM of, say 30 minutes, then averaging over 4 hours makes the standard deviation of the mean $\frac{1}{\sqrt{(8)}}$ times the standard deviation of the entrainment. It would also be possible to calculate the standard deviation

of the standard deviation, but it is not clear what practical use this value would have, so it was not calculated.

6.4.1 Sensitivity of the wrapped model to domain geometry

Numerical experiments were performed to find the sensitivity of the wrapped model output to the domain geometry of the CRM. The reference geometry was 770m vertical by 1166m horizontal, with the inversion at 600m. The following perturbations to the reference geometry were tested:

- 5,500m horizontal
- 1,200m vertical
- inversion at 1100m, 1270m vertical.

In all cases, the values of the large scale inputs to the model were chosen to be the value at the centre of the expected range of values, as follows:

$$\begin{aligned} q_{l,ct} &= 5.5 \times 10^{-4} KgKg^{-1} \\ \Delta q_t &= -6.0 \times 10^{-3} KgKg^{-1} \\ \Delta B &= 0.215 ms^{-2} \\ F_0 &= 55 Wm^{-2} \\ F_1 &= 22 Wm^{-2} \\ \theta_{l,bl} &= 290K \\ \Delta T_{sst} &= 1K \ . \end{aligned}$$

In addition, sensitivity to boundary layer liquid water potential temperature (with all other variables fixed) was tested by performing a simulation at $295K$, the upper limit of the expected range.

All simulations were performed at 5mx11m gridbox resolution. The simulations lasted 15 simulated hours and the initial spin-up period was 9 hours.

The resulting entrainments of the simulations are shown in figure 6.7 as a function of time. The gradients of the least squares fits are shown in table 6.1. The results show that the reference geometry, although small, gives values for entrainment that agree well with different geometries, considering the intrinsic standard deviation of entrainment.

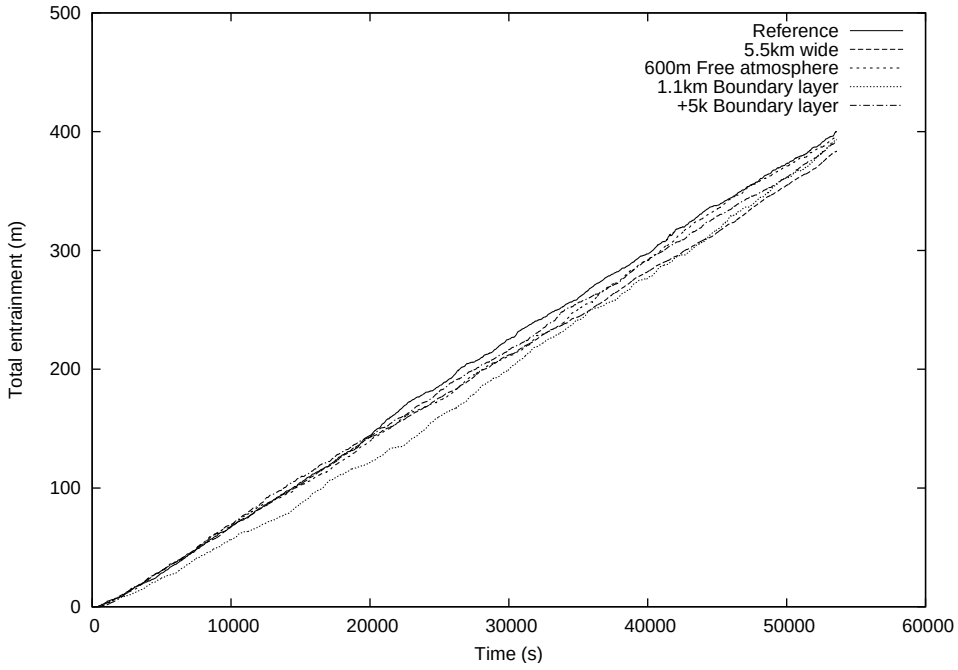

Figure 6.7: Entrainment of the CRM for different geometries and different boundary layer temperature.

Simulation	Entrainment (ms^{-1})
Reference	7.45×10^{-3}
Wide	7.28×10^{-3}
Free Atmosphere	7.59×10^{-3}
1100m Boundary layer	7.31×10^{-3}
$\theta_{l,bl} = 295K$	7.50×10^{-3}

Table 6.1: The least squares fit of the rate of entrainment for different domain geometries.

6.4.2 Analysis

The expected ranges of the large scale variables were calculated from the results of a number of field campaigns and idealised cases of nocturnal marine stratocumulus as shown in table 6.2.

Based on these values, the ranges used for iGen's analysis of the wrapped model were:

- $1 \times 10^{-4} \leq q_{l,ct} \leq 1 \times 10^{-3} KgKg^{-1}$
- $-8.0 \times 10^{-3} \leq \Delta q_t \leq -2.0 \times 10^{-3} KgKg^{-1}$
- $0.065 \leq \Delta B \leq 0.5 ms^2$
- $20 \leq F_0 \leq 110 Wm^2$
- $7 \leq F_1 \leq 33 Wm^2$.

In light of the insensitivity of entrainment to $\theta_{l,bl}$, given the other state variables, it was decided to set $\theta_{l,bl}$ to 290K, the centre of its range. ΔT_{sst} was held fixed at $+1K$. Atmospheric pressure at sea level was assumed to be $1 \times 10^5 Nm^2$.

6.4.3 Results

iGen was left running for 28 days on a desktop computer with 1.8GHz Intel Core-Duo. On return, the analysis had terminated after calculation of all DeSelby shells up to 10^{th} degree for both mean entrainment and standard deviation. Convergence of the resulting mean entrainment polynomial was shown by converting it to Chebyshev form and extracting the highest order terms (i.e. the terms for which all other terms have at least one variable of lower degree). The absolute value of the sum of the highest order terms was then compared against the standard deviation of the mean. As the polynomial converges, we would expect the highest order terms to reduce in amplitude to the level of 'noise' due to the standard deviation in the mean. At this point we would expect the sum of the highest degree terms to lie within 0.674 standard deviations of the mean 50% of the time. The polynomials of high order terms and standard deviation of the mean were evaluated at 10,000 randomly chosen points in the input domain. The proportion of points for which the high order polynomial was found to lie

Campaign	Reference	$\theta_{l,bl}(K)$	$\Delta\theta_l(K)$	$q_{t,bl}(KgKg^{-1})$	$\Delta q_t(KgKg^{-1})$	H (m)	$F_0(Wm^{-2})$	$q_{l,ct}(\times 10^{-}4KgKg^{-1})$
DYCOMS II	Stevens et.al., 2005	289.0	8.5(liquid)	9.0	-7.5	840	70	4.75
FIRE	Moeng et.al., 1996	288.0	5.5(liquid)	8.0	-4.6	[662.5 : 690]	[15 : 30]	[4.4 : 6.6]
FIRE	Albrecht et.al. 1988	289	[5 : 11]	7.0	-6.0	[500 : 1100]	40	-
ASTEX (L1)	Bretherton & Pincus, 1995	292.0	[2 : 4]	10.0	$[-5 : -8]$	[435 : 1358]	-	[2.2 : 5.2]
ASTEX (L1)	Bretherton et.al., 1995	[290 : 294.5]					-	
ASTEX (L2)	Bretherton & Pincus, 1995	[291 : 292]	11	9.0	$[-3 : -8]$	1800	-	[0 : 10.0]
ASTEX (L2)	Brethertom et.al., 1995	[291 : 293.5]						
Metastudy	Klein & Hartmann, 1993						max = 100	

Table 6.2: The ranges of large scale boundary layer values found in various published sources.

within 0.674 standard deviations was found to be 49.85%, so the polynomial was assumed to have effectively converged.

The polynomials for mean and standard deviation that resulted from the analysis are shown in full in appendix C. These can be converted into a program that evaluates the mean and standard deviation at any point in just over 2000 multiplications and additions by using Horner form evaluation. Approximations that require fewer operations can easily be created by truncating the polynomial, by finding the minimax polynomial fit using Remez' algorithm (Press et.al., 2007) or by finding the least squares fit by solving the appropriate set of linear equations (Press et.al., 2007).

The polynomials were tested against the ensemble of DYCOMS-II CRM simulations (Stevens et.al., 2005). The ensemble-average large-scale state for the final hour of the simulations was used as input to the polynomial, and the entrainment over 1 hour was predicted to be $5.27 \times 10^{-3} \pm 0.62 \times 10^{-3} ms^{-1}$. This compares very well with the ensemble average of the CRM's entrainment rate which was $5.2 \times 10^{-3} \pm 0.8 \times 10^{-3} ms^{-1}$. It is interesting to note that because of the very short simulation lengths used in the DYCOMS-II ensemble study, much of the variation between simulation results can be explained by the natural variation of the 'random walk' of entrainment.

6.5 Conclusion

iGen has analysed a wrapped, high-resolution cloud resolving model of entrainment and from this has derived a model of entrainment in terms of the large scale state. This model can be used as a closure of the large scale dynamics of the stratocumulus topped boundary layer and could be used as a parameterisation of entrainment in a global climate model.

It was also found that entrainment was sensitive to sub-grid scale structure. It was proposed that this could be modelled using a stochastic model, evidence was presented to show that the process can be described as a Gaussian random walk. Further theoretical and empirical studies would be useful to ascertain whether an alternative set of large-scale variables exists which are better predictors of entrainment. On the theoretical side, the author proposes that iGen could be extended to automatically generate good large-scale variables by using

techniques from automatic polynomial decomposition (Corless et.al., 1999) and from artificial intelligence to search the space of possible prognostic variables.

The biggest limitation of the large-scale model presented here is that it is based on a 2-dimensional simulation and, as already mentioned, 2-dimensional turbulence is known to have different characteristics than 3-dimensional turbulence. The similarity in results between our model and the 3-dimensional models in the DYCOMS-II case, however, would suggest that this does not necessarily affect entrainment rates. This is rather surprising but is in line with Moeng et.al. (1996) who also show a similar insensitivity of entrainment rate to model dimensionality. Nevertheless, it would be worthwhile repeating this experiment with a 3-dimensional simulation. It would also be worthwhile treating boundary layer temperature and sea surface temperature as input variables in order to formally show their functional role in entrainment.

Chapter 7

Further work and Conclusion

In these pages we have shown that existing climate models lack the ability to formally justify claims about the real climate system and have shown how formal methods can be used to generate models that *can* be used to formally justify claims about reality. We have presented a new 'theory of abstraction' which links statements about computer programs to statements about dynamic systems. We used this theory to show how error in computer models should be dealt with, and drew attention to certain types of error that are not properly treated in existing climate models. We showed how this theory reduces the problem of model generation to that of approximating computer programs and presented a method of approximating computer programs by analysing their code. This was illustrated by analysing some simple examples. We introduced a new type of polynomial, the DeSelby polynomial, and presented a number of algorithms that allow them to be used to efficiently approximate computer programs. These methods were implemented in a computer program, iGen, which analyses high resolution computer models and automatically produces fast, low resolution approximations of these high resolution models. We used iGen to generate a number of models of simple dynamical systems and, finally, to generate a new model of entrainment in marine stratocumulus. In this way we presented a solution to a problem that has been identified as a large source of uncertainty and error in existing climate models (Bony and Dufrence, 2005;

Dufrence and Bony, 2008).

The methods and experiments we have described here are in no way complete but are meant to lay down a foundation onto which further techniques can be built; the ultimate aim being to create a new generation of epistemically responsible climate models. There remains much scope for the development of algorithms to improve the accuracy, efficiency and capability of iGen's analysis. For example, DeSelby polynomials can be used to implement a type of adaptive mesh refinement. The author has already devised an algorithm and written code to 'grow' a DeSelby polynomial by adaptively adding shells where uncertainty is greatest. A formal analysis of the convergence properties of this 'growing' process in the multivariate case remains to be done. This process could also be extended to adaptively add individual terms within shells.

iGen's analyses could be made more efficient by adding automatic differentiation of the program code (see, e.g. Rall, 2006). This would allow iGen to calculate the response of the model's output to the addition of small perturbations at any point during the model's execution, and so allow iGen to make a higher order analysis of points in the program's execution that sensitively affect the output, while leaving lower sensitivity parts with lower order analyses. Taking this further, the sensitivity information obtained by the automatic differentiation could be combined with the functional information of the DeSelby polynomial analysis to create a higher order accurate analysis. This reduces to solving a set of simultaneous equations so could quite easily be done. This could be generalised even further by extending the automatic differentiation to allow the calculation of higher order rates of change, at which point the analysis becomes a synthesis of DeSelby polynomials and 'Taylor Models' (Berz and Hoffstatter, 1998).

In order to make iGen execute efficiently on parallel computers, it may be appropriate to modify the DeSelby basis functions to functions that are zero over a large part of their domain. This would reduce the need for inter-processor communication and thus speed up analysis. Mathematically, this technique falls under the classification of 'wavelet analysis' (see, e.g. Meyers et.al., 1993).

There is also a great potential for increased efficiency in the deeper analysis of loops. The work of Pop et. al. (2005, 2006) has been developed by the author into the 'ΠΣ-algebra' which is a formalism for dealing with loops. This remains

to be fully developed.

Another important development of iGen is to give it the ability to analyse programs 'in the limit' that a certain variable goes to zero (see section 1.1.2). This is certainly possible by developing the arithmetic on DeSelby bounds to include formal limits. At the moment, iGen's analysis gives an error bound *compared to the high resolution model*. However, this raises the question of how trustworthy the high-resolution model is. With the inclusion of limits, this problem is removed and the error reported is that compared to the underlying equations of motion. At this limit, many traditional numerical algorithms for solving partial differential equations become formally equivalent, so it may be appropriate to present the user of iGen with higher level functions that hide these details. This would make model specification faster and easier for the scientists by allowing them to express themselves in a language very close to the form of the underlying equations of motion. At the same time it would allow iGen to intelligently choose appropriate algorithms based on its analysis of the underlying equations.

iGen's ability to deal with probabilistic bounds should also be developed further. As the stratocumulus case showed, upper and lower bounds on the moments are not always appropriate. The arithmetic on DeSelby bounds could be extended to deal with a 'Gaussian noise' bound. Information would need to be held on co-variances and higher order moments. Further research would have to be done on how best to deal with truncation of higher order moments, although there seem to be no fundamental difficulties. This would allow iGen to deal better with the deterministic generation of 'noise' by chaotic systems. In this case, because of the sensitive dependence of the output on initial conditions, any attempt at automatic differentiation would fail (this has been confirmed in numerical experiments by the author).

The efficiency of the program generated by iGen could potentially be improved by searching the space of programs that calculate the equivalent, simplified polynomial. This could include calculating the Padé approximant (Guillaume and Huard, 1998; Matos, 2007), approximate polynomial decomposition (Corless et.al., 1999; Gathen et.al., 2003), and approximate polynomial factorisation (Lecerf, 2007; Gao et.al, 2004).

It would be very interesting to explore the implications of the algorithm given

in section 5.6.1 for converting between DeSelby and Gauss-Lobatto form. This was presented as an approximate algorithm but could equally be interpreted as an exact algorithm to convert to a non-polynomial basis. If this basis could be shown to have good convergence properties (as one certainly would expect, based on its close similarity to the DeSelby basis) then it could be used as a new basis for computationally efficient approximation. This basis could also be used to induce a basis which closely approximates the Chebyshev basis.

These are just a few of the ways that formal model building could be developed. It only remains for me to thank the reader for their tenacity in battling through this thesis despite the challenges of certain formidable passages and to express my hope that I have managed to give some feeling for the enormous potential of this approach, and it's importance for bringing the use of computer models squarely into the Scientific Method.

Appendix A

Mie Theory

We now give the equations necessary to calculate the scattering cross section, C_{sca}, of parallell light incident on a trasparent sphere. For a detailed description of Mie theory and derivations of these equations, see Chapter 4 of Bohren and Huffman (1998).

We take as given:

a The radius of the sphere

λ The wavelength of the incident radiation

N The refractive index of the surrounding medium

N_1 The (complex) refractive index of the sphere

Let the wave number be defined as

$$k = \frac{2\pi N}{\lambda} \ .$$

Let the size parameter be defined as

$$x = ka = \frac{2\pi N a}{\lambda} \ .$$

Let the relative refractive index be defined as

$$m = \frac{N_1}{N} \ .$$

Let the Logarithmic Derivative D_n be a function from complex numbers to complex numbers such that

$$D_{n-1}(\rho) = \frac{n}{\rho} - \frac{1}{D_n(\rho) + \frac{n}{\rho}}$$

with boundary condition

$$D_\infty = 0 + 0i \ .$$

Let $\chi_n(x)$ be a function from reals to reals such that

$$\chi_{n+1}(x) = \frac{2n+1}{x}\chi_n(x) - \chi_{n-1}(x)$$

and

$$\chi_{-1}(x) = -\sin x$$

$$\chi_0(x) = \cos x \ .$$

Let ψ and ξ be the Ricatti-Bessel functions. ψ is from reals to reals such that

$$\psi_{n+1}(x) = \frac{2n+1}{x}\psi_n(x) - \psi_{n-1}(x)$$

and

$$\psi_{-1}(x) = \cos x$$

$$\psi_0(x) = \sin x$$

ξ is from reals to complex numbers such that

$$\xi_n(x) = \psi_n(x) + i\chi_n(x) \ .$$

Now define

$$a_n = \frac{(\frac{D_n(mx)}{m} + \frac{n}{x})\psi_n(x) - \psi_{n-1}(x)}{(\frac{D_n(mx)}{m} + \frac{n}{x})\xi_n(x) - \xi_{n-1}(x)}$$

and

$$b_n = \frac{(mD_n(mx) + \frac{n}{x})\psi_n(x) - \psi_{n-1}(x)}{(mD_n(mx) + \frac{n}{x})\xi_n(x) - \xi_{n-1}(x)} \ .$$

The scattering cross section is now given by

$$C_{sca} = \frac{2\pi}{k^2}\sum_{n=1}^{\infty}(2n+1)(a_n a_n^* + b_n b_n^*)$$

in the numerical implementation, this sum is truncated to the first $N = \lfloor x + 4x^{\frac{1}{3}} + 2 \rfloor$ terms, where x is the size parameter, similarly, the boundary condition for the logarithmic derivative is set to $D_{(N+15)} = 0$, following Bohren and Huffman (1998).

Appendix B

Cloud Resolving Model Equations

B.1 Symbols

B.1.1 Prognostic variables

u_j	Velocity in j direction
π	perturbation of the Exner function from equilibrium
θ	potential temperature
q_t	total specific water content
K_m	Measure of turbulent kinetic energy ($K_m = 0.2lE^{\frac{1}{2}}$)

B.1.2 External parameters

Δx	Horizontal grid spacing
Δz	Vertical grid spacing
T_0	Equilibrium ground temperature
p_0	Equilibrium ground pressure
$\bar{\theta_v}$	Equilibrium virtual potential temperature

B.1.3 Constants

c_p	$= 1015.0 JKg^{-1}K^{-1}$	heat capacity of air at constant pressure
R_d	$= 287.1 JKg^{-1}K^{-1}$	Gas constant for dry air
c_v	$= C_p - R$	heat capacity of air at constant volume
g	$= 9.8 ms^{-1}$	gravitational acceleration
L	$= 2.47 \times 10^6 Jkg^{-1}$	latent heat of vaporization
E_s	$= 0.984$	Emissivity of seawater
C_m	$= 0.2$	turbulence constant
C_e	$= 0.2$	turbulence constant

B.1.4 Diagnostic variables

f_{u_i}	slow processes in acceleration
T_l	liquid water temperature
θ_e	Equiv. potential temp perturbation from equilibrium
$\bar{p}$	Equilibrium pressure
$\bar{\rho}$	Equilibrium density
$\bar{\Pi}$	Equilibrium Exner function
B	Buoyancy
l	turbulence length scale
T	Temperature
$\bar{\theta}_v$	Equilibrium virtual potential temperature
$\bar{c}$	Speed of sound in equilibrium conditions
q_v	specific water vapour content
q_l	specific liquid water content
q_{vs}	saturation specific water content (over water)
B	Turbulence creation due to buoyancy
S	Turbulence creation due to shear

B.2 Equilibrium state

$$\bar{\Pi} = 1 - \frac{gz}{c_p \bar{\theta}_v}$$

$$\bar{p} = p_0 \bar{\Pi}^{\frac{c_p}{R_d}}$$

$$\bar{\rho} = \frac{p_0}{R_d T_0} \bar{\pi}^{\frac{c_v}{R_d}}$$

$$\bar{c}^2 = \frac{c_p R_d \bar{\pi} \bar{\theta}_v}{c_v}$$

B.3 Diagnostic equations

$$\gamma = \frac{L}{c_p \bar{\Pi}} \tag{B.1}$$

$$\theta_e = \theta_l + \gamma q_t \tag{B.2}$$

$$l = (\Delta x \Delta z)^{\frac{1}{2}} \tag{B.3}$$

$$T = \theta(\bar{\Pi} + \pi) \approx \theta \bar{\Pi} \tag{B.4}$$

Tetens formula:

$$p_{vs} = 611.2 e^{\frac{17.27(T-273.15)}{(T-35.85)}} \tag{B.5}$$

$$q_{vs} = 0.622 \frac{p_{vs}}{p - 0.378 p_{vs}} \tag{B.6}$$

Saturation was calculated from liquid water temperature using the method described in Sommeria and Deardorff (1977). This involves calculating q_{vs} using the above equation, then if $q_{vs} \leq q_t$ the value is correct, otherwise multiply by the correction factor

$$\frac{1 + \beta q_t}{1 + \beta q_v s}$$

where

$$\beta = 0.622 \frac{L^2}{R C_p T_l^2}$$

$$q_v = \begin{cases} q_t & \text{if } q_t < q_{vs} \\ q_{vs} & \text{otherwise} \end{cases} \tag{B.7}$$

$$q_l = q_t - q_v \tag{B.8}$$

B.4 Prognostic equations

The prognostic equations are the fully compressible equations of fluid motion. Acoustic waves were integrated by splitting forcings into fast and slow parts and integrating them separately, as described by Kelmp and Wilhelmson (1978) and Skamrock and Klemp (1994). In order to improve numerical stability, a filter term is added to the prognostic equations for velocity, as described in Skamrock and Klemp (1994). This has the effect of gently filtering acoustic waves. The turbulence parameterisation is that of Klemp and Wilhelmson (1978). In the following, the Einstein summation convention is used, so repeated indices are implicitly summed over.

$$\frac{\partial u_i}{\partial t} = -c_p \bar{\theta}_v \frac{\partial \pi}{\partial x_i} + + \frac{\alpha_d}{\bar{\rho}} \frac{\partial}{\partial x_i} \frac{\partial (\bar{\rho} u_j)}{\partial x_j} f_{u_i} \tag{B.9}$$

$$\frac{\partial \pi}{\partial t} = -\frac{\bar{c}^2}{c_p \bar{\rho} \bar{\theta}_v^2} \frac{\partial \bar{\rho} \bar{\theta}_v u_j}{\partial x_j} + f_\pi \tag{B.10}$$

$$\frac{\partial q_t}{\partial t} = D_{q_t} - u_j \frac{\partial q_t}{\partial x_j} \tag{B.11}$$

$$\frac{\partial \theta_l}{\partial t} = D_{\theta_l} - u_j \frac{\partial \theta_l}{\partial x_j} + f_r \tag{B.12}$$

$$\frac{\partial K_m}{\partial t} = -u_j \frac{\partial K_m}{\partial x_j} + \frac{c_m^2 l^2}{2K_m} (B + S) + \frac{1}{2} \frac{\partial^2 (K_m^2)}{\partial x_j^2} - \frac{c_e K_m^2}{2 c_m l^2} \tag{B.13}$$

where

$$B = 3 g K_m \left((1 - H(q_l)) \frac{-1}{\theta} \frac{\partial \theta}{\partial z} + H(q_l)(-A \frac{\partial \theta_e}{\partial z} + \frac{\partial q_c}{\partial z}) \right) \tag{B.14}$$

where H is the Heaviside step function and

$$A = \frac{1}{\bar{\theta}} \left(\frac{1 + \frac{1.61 L q_v}{R_d T}}{1 + \frac{0.622 L^2 q_v}{c_p R_d T^2}} \right)$$

and

$$S = K_m \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)^2 \tag{B.15}$$

B.4.1 Turbulence fluxes

$$D_{u_i} = \frac{\partial}{\partial x_j}\left(K_m\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \frac{2}{3c_m^2 l^2}\delta_{ij}K_m^2\right) \tag{B.16}$$

For $\phi \in \{\theta, q_t\}$

$$D_\phi = 3\frac{\partial}{\partial x_j}\left(K_m\frac{\partial \phi}{\partial x_j}\right) \tag{B.17}$$

B.4.2 Slow (non-acoustic) waves

Slow forcings on the velocity fields are those due to buoyancy, turbulence and advection:

$$f_{u_i} = \delta_{i3}g\left(\frac{\theta_l + \gamma q_l}{\bar{\theta}} - 1 + 0.61q_t - 1.61q_l\right) + D_{u_i} - u_j\frac{\partial u_i}{\partial x_j} \tag{B.18}$$

B.4.3 Radiative fluxes

In-cloud radiative fluxes were calculated using the scheme described in Stevens et.al. (2005) and Larson et.al. (2007) where the rate of change of liquid water potential temperature is given by

$$f_r = \frac{1}{C_p\bar{\rho}\bar{\pi}}\frac{\partial}{\partial z}\left(F_0 e^{-Q(z,\infty)} + F_1 e^{-Q(0,z)}\right) \tag{B.19}$$

where

$$Q(a,b) = \kappa\int_a^b \rho q_t dz$$

where, following Larson et.al. (2007), $\kappa = 119m^2kg^{-1}$.

B.4.4 Surface fluxes

The surface fluxes of latent and sensible heat were calculated using a simple bulk aerodynamic formulation described in Krishnamurti and Bounoua (1995). Fluxes were added to the lowest gridbox of each column according to

$$\left.\frac{\partial \theta}{\partial t}\right|_{surf} = \frac{1}{\Delta z}\|u_{10}\|C_h(T_{sst} - T) \tag{B.20}$$

and

$$\left.\frac{\partial q_t}{\partial t}\right|_{surf} = \frac{1}{\Delta z}\|u_{10}\|C_q(q_{sat} - q_t) \tag{B.21}$$

where T and q_t are the temperature and total water of the lowest gridbox, respectively, and

$$u_{10} = u_x \frac{\log(\frac{10.0}{z_0})}{\log(\frac{\Delta z}{2z_0})}$$

z_0 is the roughness length, which was taken to have a constant value at $5\times10^{-4}m$ based on figures in Stull (1988). The exchange coefficients were set constant at $C_h = 1.4 \times 10^{-3}$ and $C_q = 1.6 \times 10^{-3}$ based on figures in Krishnamurti and Bounoua (1995).

B.5 Numerical implementation

The equations were integrated on a staggered grid in which pressure, temperature and total water are defined on one grid, while horizontal velocity is represented at points displaced half a grid spacing to the right and vertical velocity is represented at points displaced half a grid spacing below the thermodynamic variables.

Integration was done on using a leapfrog scheme, following Klemp and Wilhelmson (1978).

B.6 Numerical treatment of timesplitting

Pressure and velocity fields were updated on a smaller timestep than that of the other fields in order to account for acoustic waves. The prognostic equations for each small timestep are given in equations B.9 and B.10, where f_{u_i} and f_π are taken to be constant and evaluated at the leapfrog mid-point. During the integration, the vertical variation of the speed of sound was ignored and taken to be fixed at the top-of-domain value.

B.7 Boundary conditions

The left and right boundaries are periodic in all variables. The upper and lower boundaries each lie on v grid points. At the ground u and v have the Dirichlet boundary condition of $v = 0$ and $u = 0$. Other variables have the condition that $\frac{\partial \pi}{\partial z}$ goes to zero in order that there is no sub-grid turbulent flux across

the boundary, allowing surface fluxes to be dealt with separately. At the top of domain boundary, $v = -Dh$ where D is the large scale divergence and h is the height; u goes to zero since we are assuming no geostrophic wind in the free atmosphere air entering the top of the domain; π goes to zero (above is in equilibrium, π is a perturbation) q_t and θ_l go to the large-scale free atmosphere values and K_m has the boundary condition $\frac{\partial K_m}{\partial z} = 0$ in order to ensure that there is no sub-grid turbulent flux of turbulence.

Appendix C

Polynomials for entrainment

C.1 Mean entrainment

The mean entrainment polynomial, in Chebyshev form, is:

$$0.000120718F_0^2F_1^8+9.21863\times10^{-05}F_0F_1^8-0.000141767\Delta B^2F_1^8+0.000526718\Delta BF_1^8-$$
$$1.13686\times10^{-05}\Delta qt^2F_1^8+0.000141682\Delta qtF_1^8+6.8708\times10^{-05}ql^2F_1^8+0.000117345qlF_1^8-$$
$$0.000158558F_1^8-8.4255\times10^{-05}F_0^2F_1^7-0.000110234F_0F_1^7-0.000102062\Delta B^2F_1^7+$$
$$0.000118472\Delta BF_1^7-0.000110617\Delta qt^2F_1^7-4.82755\times10^{-05}\Delta qtF_1^7-1.86547\times$$
$$10^{-06}ql^2F_1^7+0.000176195qlF_1^7-0.000180746F_1^7+0.000266137F_0^2F_1^6+0.00023476F_0F_1^6+$$
$$0.000427795\Delta B^2F_1^6-0.000422387\Delta BF_1^6+0.000253881\Delta qt^2F_1^6-0.000177268\Delta qtF_1^6+$$
$$0.000299972ql^2F_1^6+0.000242528qlF_1^6+0.000831622F_1^6+0.000446523F_0^2F_1^5+$$
$$6.72172\times10^{-05}F_0F_1^5+0.000236846\Delta B^2F_1^5-5.13986\times10^{-05}\Delta BF_1^5+0.000252128\Delta qt^2F_1^5+$$
$$0.000148603\Delta qtF_1^5+0.000294844ql^2F_1^5-0.000115784qlF_1^5+0.000700522F_1^5-$$
$$4.98684\times10^{-05}\Delta B^2F_0^4F_1^4+0.000147548\Delta BF_0^4F_1^4+2.06662\times10^{-05}\Delta qt^2F_0^4F_1^4-$$
$$3.5955\times10^{-05}\Delta qtF_0^4F_1^4+0.000106293ql^2F_0^4F_1^4+7.97842\times10^{-05}qlF_0^4F_1^4+$$
$$1.7325\times10^{-05}F_0^4F_1^4+2.2407\times10^{-05}\Delta B^2F_0^3F_1^4+0.000109838\Delta BF_0^3F_1^4+5.39368\times$$
$$10^{-05}\Delta qt^2F_0^3F_1^4+0.000122118\Delta qtF_0^3F_1^4+0.000137683ql^2F_0^3F_1^4+0.000142811qlF_0^3F_1^4+$$
$$9.67008\times10^{-05}F_0^3F_1^4-8.33753\times10^{-05}\Delta B^4F_0^2F_1^4+0.000157462\Delta B^3F_0^2F_1^4+$$
$$2.5246\times10^{-05}\Delta qt^2\Delta B^2F_0^2F_1^4-4.22562\times10^{-05}\Delta qt\Delta B^2F_0^2F_1^4+1.8974\times10^{-05}ql^2\Delta B^2F_0^2F_1^4+$$
$$1.73125\times10^{-05}ql\Delta B^2F_0^2F_1^4+9.78452\times10^{-05}\Delta B^2F_0^2F_1^4-0.000226556\Delta qt^2\Delta BF_0^2F_1^4+$$

$$0.000198925\Delta qt\Delta BF_0^2F_1^4-0.000136279ql^2\Delta BF_0^2F_1^4+5.85139\times10^{-05}ql\Delta BF_0^2F_1^4-$$
$$0.000268105\Delta BF_0^2F_1^4-0.000108133\Delta qt^4F_0^2F_1^4+1.13139\times10^{-05}\Delta qt^3F_0^2F_1^4-$$
$$1.20202\times10^{-05}ql^2\Delta qt^2F_0^2F_1^4+4.66491\times10^{-05}ql\Delta qt^2F_0^2F_1^4+0.000155852\Delta qt^2F_0^2F_1^4+$$
$$6.31406\times10^{-05}ql^2\Delta qtF_0^2F_1^4-5.96438\times10^{-06}ql\Delta qtF_0^2F_1^4-8.99464\times10^{-05}\Delta qtF_0^2F_1^4-$$
$$8.77068\times10^{-05}ql^4F_0^2F_1^4+9.90035\times10^{-05}ql^3F_0^2F_1^4+5.43902\times10^{-05}ql^2F_0^2F_1^4-$$
$$0.000106607qlF_0^2F_1^4+0.000293113F_0^2F_1^4-2.82823\times10^{-05}\Delta B^4F_0F_1^4+0.000180355\Delta B^3F_0F_1^4+$$
$$6.39463\times10^{-06}\Delta qt^2\Delta B^2F_0F_1^4+7.68636\times10^{-05}\Delta qt\Delta B^2F_0F_1^4+8.20314\times$$
$$10^{-05}ql^2\Delta B^2F_0F_1^4+6.25263\times10^{-05}ql\Delta B^2F_0F_1^4-3.05733\times10^{-05}\Delta B^2F_0F_1^4+$$
$$3.56843\times10^{-05}\Delta qt^2\Delta BF_0F_1^4-4.87898\times10^{-05}\Delta qt\Delta BF_0F_1^4-0.000148511ql^2\Delta BF_0F_1^4-$$
$$0.000173876ql\Delta BF_0F_1^4-0.000331168\Delta BF_0F_1^4+3.54761\times10^{-05}\Delta qt^4F_0F_1^4+$$
$$2.47051\times10^{-05}\Delta qt^3F_0F_1^4+3.27847\times10^{-05}ql^2\Delta qt^2F_0F_1^4-4.9746\times10^{-05}ql\Delta qt^2F_0F_1^4-$$
$$9.0314\times10^{-05}\Delta qt^2F_0F_1^4-1.19916\times10^{-05}ql^2\Delta qtF_0F_1^4-0.000123904ql\Delta qtF_0F_1^4-$$
$$0.000116553\Delta qtF_0F_1^4-2.53845\times10^{-05}ql^4F_0F_1^4+8.31465\times10^{-05}ql^3F_0F_1^4-$$
$$8.26584\times10^{-05}ql^2F_0F_1^4-0.000168653qlF_0F_1^4-5.97035\times10^{-05}F_0F_1^4-0.000153261\Delta qt^2\Delta B^4F_1^4+$$
$$0.000110064\Delta qt\Delta B^4F_1^4-1.53256\times10^{-05}ql^2\Delta B^4F_1^4+9.46363\times10^{-05}ql\Delta B^4F_1^4-$$
$$9.19283\times10^{-05}\Delta B^4F_1^4+0.000220923\Delta qt^2\Delta B^3F_1^4-4.43544\times10^{-05}\Delta qt\Delta B^3F_1^4+$$
$$1.04811\times10^{-05}ql^2\Delta B^3F_1^4-0.000256416ql\Delta B^3F_1^4+0.000158222\Delta B^3F_1^4-8.66921\times$$
$$10^{-05}\Delta qt^4\Delta B^2F_1^4+0.000117528\Delta qt^3\Delta B^2F_1^4+3.23779\times10^{-05}ql^2\Delta qt^2\Delta B^2F_1^4+$$
$$9.96191\times10^{-05}ql\Delta qt^2\Delta B^2F_1^4+7.73728\times10^{-05}\Delta qt^2\Delta B^2F_1^4+6.56758\times10^{-05}ql^2\Delta qt\Delta B^2F_1^4-$$
$$8.23853\times10^{-05}ql\Delta qt\Delta B^2F_1^4-4.48242\times10^{-05}\Delta qt\Delta B^2F_1^4-0.000127349ql^4\Delta B^2F_1^4-$$
$$8.40089\times10^{-05}ql^3\Delta B^2F_1^4+2.68065\times10^{-05}ql^2\Delta B^2F_1^4+0.000180006ql\Delta B^2F_1^4+$$
$$0.000273915\Delta B^2F_1^4+9.28133\times10^{-06}\Delta qt^4\Delta BF_1^4-0.000225091\Delta qt^3\Delta BF_1^4-$$
$$4.35986\times10^{-05}ql^2\Delta qt^2\Delta BF_1^4+3.40657\times10^{-05}ql\Delta qt^2\Delta BF_1^4-0.000228237\Delta qt^2\Delta BF_1^4+$$
$$5.66919\times10^{-05}ql^2\Delta qt\Delta BF_1^4+0.000332733ql\Delta qt\Delta BF_1^4+0.000268683\Delta qt\Delta BF_1^4+$$
$$0.000182818ql^4\Delta BF_1^4+0.000338417ql^3\Delta BF_1^4-0.000148401ql^2\Delta BF_1^4-0.000323938ql\Delta BF_1^4-$$
$$0.000659279\Delta BF_1^4-7.73986\times10^{-05}ql^2\Delta qt^4F_1^4-5.1938\times10^{-05}ql\Delta qt^4F_1^4-$$
$$0.000106091\Delta qt^4F_1^4+6.8263\times10^{-05}ql^2\Delta qt^3F_1^4-1.83195\times10^{-05}ql\Delta qt^3F_1^4+$$
$$0.000116535\Delta qt^3F_1^4-6.8845\times10^{-05}ql^4\Delta qt^2F_1^4+8.19104\times10^{-06}ql^3\Delta qt^2F_1^4+$$
$$2.03708\times10^{-06}ql^2\Delta qt^2F_1^4-5.51161\times10^{-05}ql\Delta qt^2F_1^4+0.00013162\Delta qt^2F_1^4-$$
$$3.60505\times10^{-05}ql^4\Delta qtF_1^4-6.01226\times10^{-05}ql^3\Delta qtF_1^4-7.44489\times10^{-05}ql^2\Delta qtF_1^4-$$
$$0.000129137ql\Delta qtF_1^4-0.000118585\Delta qtF_1^4-0.000102458ql^4F_1^4+6.37388\times10^{-05}ql^3F_1^4+$$
$$2.02165\times10^{-05}ql^2F_1^4-6.89003\times10^{-05}qlF_1^4+0.000337495F_1^4-4.51634\times10^{-05}\Delta B^2F_0^4F_1^3+$$
$$6.54037\times10^{-05}\Delta BF_0^4F_1^3-7.03457\times10^{-05}\Delta qt^2F_0^4F_1^3+4.08406\times10^{-05}\Delta qtF_0^4F_1^3-$$
$$4.24284\times10^{-05}ql^2F_0^4F_1^3-7.3821\times10^{-05}qlF_0^4F_1^3-0.000139284F_0^4F_1^3+3.02588\times$$
$$10^{-05}\Delta B^2F_0^3F_1^3-0.000175593\Delta BF_0^3F_1^3-8.80499\times10^{-05}\Delta qt^2F_0^3F_1^3-0.000126225\Delta qtF_0^3F_1^3-$$
$$1.64297\times10^{-05}ql^2F_0^3F_1^3+0.000133127qlF_0^3F_1^3+4.27086\times10^{-05}F_0^3F_1^3+0.0001099\Delta B^4F_0^2F_1^3+$$

$$2.03046\times10^{-05}\Delta B^3F_0^2F_1^3+0.000136884\Delta qt^2\Delta B^2F_0^2F_1^3-4.22315\times10^{-05}\Delta qt\Delta B^2F_0^2F_1^3-$$
$$4.22068\times10^{-06}ql^2\Delta B^2F_0^2F_1^3-0.000210773ql\Delta B^2F_0^2F_1^3-9.22632\times10^{-06}\Delta B^2F_0^2F_1^3-$$
$$3.44055\times10^{-05}\Delta qt^2\Delta BF_0^2F_1^3+0.000104994\Delta qt\Delta BF_0^2F_1^3+0.000118784ql^2\Delta BF_0^2F_1^3+$$
$$0.000130186ql\Delta BF_0^2F_1^3+1.59801\times10^{-05}\Delta BF_0^2F_1^3+9.62351\times10^{-05}\Delta qt^4F_0^2F_1^3-$$
$$9.33077\times10^{-05}\Delta qt^3F_0^2F_1^3+7.05801\times10^{-05}ql^2\Delta qt^2F_0^2F_1^3-0.000123367ql\Delta qt^2F_0^2F_1^3-$$
$$4.18125\times10^{-05}\Delta qt^2F_0^2F_1^3+3.37138\times10^{-05}ql^2\Delta qtF_0^2F_1^3-0.00010539ql\Delta qtF_0^2F_1^3+$$
$$2.64427\times10^{-05}\Delta qtF_0^2F_1^3+7.23148\times10^{-05}ql^4F_0^2F_1^3-2.92829\times10^{-05}ql^3F_0^2F_1^3-$$
$$7.41997\times10^{-05}ql^2F_0^2F_1^3-4.99711\times10^{-05}qlF_0^2F_1^3-0.000669952F_0^2F_1^3+4.10505\times$$
$$10^{-05}\Delta B^4F_0F_1^3-4.01105\times10^{-05}\Delta B^3F_0F_1^3+3.29234\times10^{-05}\Delta qt^2\Delta B^2F_0F_1^3-$$
$$3.60127\times10^{-05}\Delta qt\Delta B^2F_0F_1^3+5.39697\times10^{-05}ql^2\Delta B^2F_0F_1^3-2.57881\times10^{-05}ql\Delta B^2F_0F_1^3+$$
$$8.49646\times10^{-05}\Delta B^2F_0F_1^3-2.62915\times10^{-05}\Delta qt^2\Delta BF_0F_1^3+3.45203\times10^{-05}\Delta qt\Delta BF_0F_1^3-$$
$$0.000152313ql^2\Delta BF_0F_1^3+6.38562\times10^{-05}ql\Delta BF_0F_1^3+0.000199452\Delta BF_0F_1^3-$$
$$6.65727\times10^{-05}\Delta qt^4F_0F_1^3-7.07483\times10^{-05}\Delta qt^3F_0F_1^3+2.8803\times10^{-05}ql^2\Delta qt^2F_0F_1^3-$$
$$1.60726\times10^{-05}ql\Delta qt^2F_0F_1^3+0.000105885\Delta qt^2F_0F_1^3-4.57483\times10^{-05}ql^2\Delta qtF_0F_1^3-$$
$$3.49363\times10^{-05}ql\Delta qtF_0F_1^3+0.00015091\Delta qtF_0F_1^3+5.66541\times10^{-05}ql^4F_0F_1^3+$$
$$6.13195\times10^{-05}ql^3F_0F_1^3+0.000163147ql^2F_0F_1^3-0.000216094qlF_0F_1^3-0.000134427F_0F_1^3-$$
$$1.78389\times10^{-05}\Delta qt^2\Delta B^4F_1^3+6.56909\times10^{-05}\Delta qt\Delta B^4F_1^3+7.90487\times10^{-05}ql^2\Delta B^4F_1^3-$$
$$2.99083\times10^{-05}ql\Delta B^4F_1^3+0.000150135\Delta B^4F_1^3+0.000152225\Delta qt^2\Delta B^3F_1^3+$$
$$2.96042\times10^{-05}\Delta qt\Delta B^3F_1^3-8.33418\times10^{-05}ql^2\Delta B^3F_1^3-0.000123057ql\Delta B^3F_1^3-$$
$$6.24373\times10^{-05}\Delta B^3F_1^3-9.94049\times10^{-05}\Delta qt^4\Delta B^2F_1^3+7.84926\times10^{-05}\Delta qt^3\Delta B^2F_1^3-$$
$$2.22621\times10^{-05}ql^2\Delta qt^2\Delta B^2F_1^3-0.000172425ql\Delta qt^2\Delta B^2F_1^3-3.03502\times10^{-05}\Delta qt^2\Delta B^2F_1^3-$$
$$3.69663\times10^{-05}ql^2\Delta qt\Delta B^2F_1^3-4.43434\times10^{-05}ql\Delta qt\Delta B^2F_1^3-0.00016961\Delta qt\Delta B^2F_1^3-$$
$$8.15268\times10^{-05}ql^4\Delta B^2F_1^3-0.000205504ql^3\Delta B^2F_1^3+1.80597\times10^{-06}ql^2\Delta B^2F_1^3-$$
$$2.17032\times10^{-05}ql\Delta B^2F_1^3-1.83361\times10^{-05}\Delta B^2F_1^3+0.000246901\Delta qt^4\Delta BF_1^3-$$
$$0.000226736\Delta qt^3\Delta BF_1^3+2.10068\times10^{-05}ql^2\Delta qt^2\Delta BF_1^3+0.000116653ql\Delta qt^2\Delta BF_1^3-$$
$$6.99171\times10^{-05}\Delta qt^2\Delta BF_1^3+0.000195373ql^2\Delta qt\Delta BF_1^3-0.00011137ql\Delta qt\Delta BF_1^3+$$
$$0.000367237\Delta qt\Delta BF_1^3+5.92025\times10^{-05}ql^4\Delta BF_1^3+0.000279384ql^3\Delta BF_1^3+$$
$$8.70485\times10^{-05}ql^2\Delta BF_1^3+0.000206923ql\Delta BF_1^3-0.000176883\Delta BF_1^3+1.29175\times$$
$$10^{-05}ql^2\Delta qt^4F_1^3-0.000206296ql\Delta qt^4F_1^3-0.000115907\Delta qt^4F_1^3-2.40036\times10^{-05}ql^2\Delta qt^3F_1^3+$$
$$0.000349768ql\Delta qt^3F_1^3+0.000131203\Delta qt^3F_1^3+3.17266\times10^{-06}ql^4\Delta qt^2F_1^3-5.43128\times$$
$$10^{-05}ql^3\Delta qt^2F_1^3+5.13174\times10^{-05}ql^2\Delta qt^2F_1^3-5.66524\times10^{-05}ql\Delta qt^2F_1^3-0.000185618\Delta qt^2F_1^3+$$
$$1.20611\times10^{-07}ql^4\Delta qtF_1^3-0.00010896ql^3\Delta qtF_1^3-0.000152344ql^2\Delta qtF_1^3-0.000189359ql\Delta qtF_1^3-$$
$$0.000381399\Delta qtF_1^3+2.58893\times10^{-05}ql^4F_1^3-0.000249907ql^3F_1^3-0.00028737ql^2F_1^3+$$
$$0.000348225qlF_1^3-0.000657733F_1^3-0.000121838F_0^8F_1^2+8.77709\times10^{-06}F_0^7F_1^2+$$
$$0.00015899F_0^6F_1^2+0.000255041F_0^5F_1^2+0.000169269\Delta B^4F_0^4F_1^2-0.000237281\Delta B^3F_0^4F_1^2-$$
$$4.66645\times10^{-05}\Delta qt^2\Delta B^2F_0^4F_1^2-2.16462\times10^{-05}\Delta qt\Delta B^2F_0^4F_1^2-0.000133305ql^2\Delta B^2F_0^4F_1^2-$$

$$0.000165228 q l \Delta B^2 F_0^4 F_1^2 - 6.27828 \times 10^{-05} \Delta B^2 F_0^4 F_1^2 + 0.000105813 \Delta q t^2 \Delta B F_0^4 F_1^2 -$$
$$2.31362 \times 10^{-05} \Delta q t \Delta B F_0^4 F_1^2 + 0.000273403 q l^2 \Delta B F_0^4 F_1^2 + 0.000315133 q l \Delta B F_0^4 F_1^2 +$$
$$0.000356514 \Delta B F_0^4 F_1^2 + 7.62969 \times 10^{-07} \Delta q t^4 F_0^4 F_1^2 + 8.44508 \times 10^{-05} \Delta q t^3 F_0^4 F_1^2 +$$
$$1.42768 \times 10^{-05} q l^2 \Delta q t^2 F_0^4 F_1^2 + 3.97332 \times 10^{-05} q l \Delta q t^2 F_0^4 F_1^2 - 6.18074 \times 10^{-05} \Delta q t^2 F_0^4 F_1^2 -$$
$$3.56496 \times 10^{-07} q l^2 \Delta q t F_0^4 F_1^2 + 1.27102 \times 10^{-05} q l \Delta q t F_0^4 F_1^2 - 3.72301 \times 10^{-05} \Delta q t F_0^4 F_1^2 -$$
$$0.000107288 q l^4 F_0^4 F_1^2 - 0.000106392 q l^3 F_0^4 F_1^2 - 7.34143 \times 10^{-05} q l^2 F_0^4 F_1^2 + 9.83312 \times$$
$$10^{-05} q l F_0^4 F_1^2 - 9.6862 \times 10^{-05} F_0^4 F_1^2 + 0.00016983 \Delta B^4 F_0^3 F_1^2 - 0.000217641 \Delta B^3 F_0^3 F_1^2 +$$
$$6.09667 \times 10^{-05} \Delta q t^2 \Delta B^2 F_0^3 F_1^2 - 6.13422 \times 10^{-05} \Delta q t \Delta B^2 F_0^3 F_1^2 + 2.59872 \times$$
$$10^{-05} q l^2 \Delta B^2 F_0^3 F_1^2 - 1.73976 \times 10^{-05} q l \Delta B^2 F_0^3 F_1^2 + 4.30672 \times 10^{-05} \Delta B^2 F_0^3 F_1^2 +$$
$$0.000137084 \Delta q t^2 \Delta B F_0^3 F_1^2 - 6.27609 \times 10^{-05} \Delta q t \Delta B F_0^3 F_1^2 + 3.92408 \times 10^{-05} q l^2 \Delta B F_0^3 F_1^2 -$$
$$0.000195556 q l \Delta B F_0^3 F_1^2 + 5.13535 \times 10^{-05} \Delta B F_0^3 F_1^2 + 5.75733 \times 10^{-05} \Delta q t^4 F_0^3 F_1^2 -$$
$$4.96946 \times 10^{-05} \Delta q t^3 F_0^3 F_1^2 + 4.68278 \times 10^{-05} q l^2 \Delta q t^2 F_0^3 F_1^2 - 4.7552 \times 10^{-05} q l \Delta q t^2 F_0^3 F_1^2 -$$
$$0.00012187 \Delta q t^2 F_0^3 F_1^2 - 1.44381 \times 10^{-05} q l^2 \Delta q t F_0^3 F_1^2 - 4.03516 \times 10^{-05} q l \Delta q t F_0^3 F_1^2 +$$
$$4.18787 \times 10^{-05} \Delta q t F_0^3 F_1^2 + 1.6288 \times 10^{-06} q l^4 F_0^3 F_1^2 - 5.93827 \times 10^{-05} q l^3 F_0^3 F_1^2 +$$
$$4.23112 \times 10^{-05} q l^2 F_0^3 F_1^2 + 0.000231862 q l F_0^3 F_1^2 - 0.00034142 F_0^3 F_1^2 - 2.64615 \times$$
$$10^{-05} \Delta q t^2 \Delta B^4 F_0^2 F_1^2 - 5.23457 \times 10^{-06} \Delta q t \Delta B^4 F_0^2 F_1^2 + 2.19109 \times 10^{-05} q l^2 \Delta B^4 F_0^2 F_1^2 +$$
$$0.000122768 q l \Delta B^4 F_0^2 F_1^2 + 4.63107 \times 10^{-05} \Delta B^4 F_0^2 F_1^2 + 6.3245 \times 10^{-06} \Delta q t^2 \Delta B^3 F_0^2 F_1^2 +$$
$$7.04478 \times 10^{-05} \Delta q t \Delta B^3 F_0^2 F_1^2 - 8.96021 \times 10^{-06} q l^2 \Delta B^3 F_0^2 F_1^2 - 8.83598 \times 10^{-05} q l \Delta B^3 F_0^2 F_1^2 +$$
$$8.08909 \times 10^{-06} \Delta B^3 F_0^2 F_1^2 - 0.000114654 \Delta q t^4 \Delta B^2 F_0^2 F_1^2 + 5.2334 \times 10^{-05} \Delta q t^3 \Delta B^2 F_0^2 F_1^2 -$$
$$6.62823 \times 10^{-07} q l^2 \Delta q t^2 \Delta B^2 F_0^2 F_1^2 + 6.95329 \times 10^{-05} q l \Delta q t^2 \Delta B^2 F_0^2 F_1^2 + 0.000107862 \Delta q t^2 \Delta B^2 F_0^2 F_1^2 +$$
$$1.62381 \times 10^{-05} q l^2 \Delta q t \Delta B^2 F_0^2 F_1^2 - 0.000112755 q l \Delta q t \Delta B^2 F_0^2 F_1^2 - 0.000125239 \Delta q t \Delta B^2 F_0^2 F_1^2 -$$
$$0.000103046 q l^4 \Delta B^2 F_0^2 F_1^2 - 0.000192937 q l^3 \Delta B^2 F_0^2 F_1^2 + 6.4396 \times 10^{-05} q l^2 \Delta B^2 F_0^2 F_1^2 +$$
$$0.000231342 q l \Delta B^2 F_0^2 F_1^2 + 0.00014115 \Delta B^2 F_0^2 F_1^2 + 0.000146735 \Delta q t^4 \Delta B F_0^2 F_1^2 -$$
$$8.08454 \times 10^{-05} \Delta q t^3 \Delta B F_0^2 F_1^2 - 3.38041 \times 10^{-05} q l^2 \Delta q t^2 \Delta B F_0^2 F_1^2 - 0.000118602 q l \Delta q t^2 \Delta B F_0^2 F_1^2 -$$
$$0.000106159 \Delta q t^2 \Delta B F_0^2 F_1^2 + 4.20169 \times 10^{-05} q l^2 \Delta q t \Delta B F_0^2 F_1^2 + 0.000364971 q l \Delta q t \Delta B F_0^2 F_1^2 +$$
$$0.000122075 \Delta q t \Delta B F_0^2 F_1^2 + 7.08716 \times 10^{-05} q l^4 \Delta B F_0^2 F_1^2 + 5.03772 \times 10^{-05} q l^3 \Delta B F_0^2 F_1^2 -$$
$$8.67514 \times 10^{-05} q l^2 \Delta B F_0^2 F_1^2 - 0.000223081 q l \Delta B F_0^2 F_1^2 - 0.000240037 \Delta B F_0^2 F_1^2 +$$
$$1.04412 \times 10^{-05} q l^2 \Delta q t^4 F_0^2 F_1^2 + 7.90225 \times 10^{-05} q l \Delta q t^4 F_0^2 F_1^2 - 3.04973 \times 10^{-05} \Delta q t^4 F_0^2 F_1^2 +$$
$$4.27925 \times 10^{-05} q l^2 \Delta q t^3 F_0^2 F_1^2 - 0.000133724 q l \Delta q t^3 F_0^2 F_1^2 - 6.79021 \times 10^{-05} \Delta q t^3 F_0^2 F_1^2 -$$
$$3.43323 \times 10^{-05} q l^4 \Delta q t^2 F_0^2 F_1^2 + 9.71427 \times 10^{-06} q l^3 \Delta q t^2 F_0^2 F_1^2 + 5.28779 \times 10^{-05} q l^2 \Delta q t^2 F_0^2 F_1^2 +$$
$$8.14509 \times 10^{-05} q l \Delta q t^2 F_0^2 F_1^2 + 6.97032 \times 10^{-05} \Delta q t^2 F_0^2 F_1^2 + 3.47821 \times 10^{-05} q l^4 \Delta q t F_0^2 F_1^2 -$$
$$5.10603 \times 10^{-05} q l^3 \Delta q t F_0^2 F_1^2 - 6.66358 \times 10^{-05} q l^2 \Delta q t F_0^2 F_1^2 + 3.96392 \times 10^{-05} q l \Delta q t F_0^2 F_1^2 +$$
$$4.44552 \times 10^{-05} \Delta q t F_0^2 F_1^2 - 2.27357 \times 10^{-05} q l^4 F_0^2 F_1^2 - 5.14853 \times 10^{-05} q l^3 F_0^2 F_1^2 +$$
$$3.93579 \times 10^{-05} q l^2 F_0^2 F_1^2 + 6.82368 \times 10^{-05} q l F_0^2 F_1^2 - 0.000328531 F_0^2 F_1^2 - 9.29926 \times$$
$$10^{-05} \Delta q t^2 \Delta B^4 F_0 F_1^2 + 5.44205 \times 10^{-05} \Delta q t \Delta B^4 F_0 F_1^2 - 4.93499 \times 10^{-05} q l^2 \Delta B^4 F_0 F_1^2 +$$
$$2.73115 \times 10^{-05} q l \Delta B^4 F_0 F_1^2 - 0.000155036 \Delta B^4 F_0 F_1^2 + 0.000186217 \Delta q t^2 \Delta B^3 F_0 F_1^2 -$$

$$6.62698\times10^{-05}\Delta qt\Delta B^3F_0F_1^2+0.000239268ql^2\Delta B^3F_0F_1^2+0.000210954ql\Delta B^3F_0F_1^2+$$
$$0.000368971\Delta B^3F_0F_1^2+8.84146\times10^{-06}\Delta qt^4\Delta B^2F_0F_1^2-6.48924\times10^{-06}\Delta qt^3\Delta B^2F_0F_1^2+$$
$$7.47537\times10^{-05}ql^2\Delta qt^2\Delta B^2F_0F_1^2+0.000158353ql\Delta qt^2\Delta B^2F_0F_1^2+3.03866\times$$
$$10^{-05}\Delta qt^2\Delta B^2F_0F_1^2-0.000122159ql^2\Delta qt\Delta B^2F_0F_1^2-0.00022464ql\Delta qt\Delta B^2F_0F_1^2-$$
$$7.73525\times10^{-05}\Delta qt\Delta B^2F_0F_1^2-0.000200683ql^4\Delta B^2F_0F_1^2-0.00017794ql^3\Delta B^2F_0F_1^2-$$
$$5.04534\times10^{-05}ql^2\Delta B^2F_0F_1^2+6.52408\times10^{-05}ql\Delta B^2F_0F_1^2+4.67204\times10^{-05}\Delta B^2F_0F_1^2-$$
$$0.000122953\Delta qt^4\Delta BF_0F_1^2+2.76101\times10^{-05}\Delta qt^3\Delta BF_0F_1^2-0.000239373ql^2\Delta qt^2\Delta BF_0F_1^2-$$
$$0.000327688ql\Delta qt^2\Delta BF_0F_1^2-0.000359607\Delta qt^2\Delta BF_0F_1^2+0.00034913ql^2\Delta qt\Delta BF_0F_1^2+$$
$$0.000422447ql\Delta qt\Delta BF_0F_1^2+0.000222093\Delta qt\Delta BF_0F_1^2+9.49411\times10^{-05}ql^4\Delta BF_0F_1^2+$$
$$0.000664447ql^3\Delta BF_0F_1^2-0.000128791ql^2\Delta BF_0F_1^2-0.000533126ql\Delta BF_0F_1^2-$$
$$9.21262\times10^{-05}\Delta BF_0F_1^2-4.87505\times10^{-06}ql^2\Delta qt^4F_0F_1^2+0.00010852ql\Delta qt^4F_0F_1^2+$$
$$8.1965\times10^{-05}\Delta qt^4F_0F_1^2+1.44671\times10^{-05}ql^2\Delta qt^3F_0F_1^2-0.0001048ql\Delta qt^3F_0F_1^2-$$
$$9.03886\times10^{-05}\Delta qt^3F_0F_1^2-1.68239\times10^{-05}ql^4\Delta qt^2F_0F_1^2+3.89257\times10^{-05}ql^3\Delta qt^2F_0F_1^2+$$
$$5.22268\times10^{-05}ql^2\Delta qt^2F_0F_1^2+0.000117069ql\Delta qt^2F_0F_1^2+0.000198843\Delta qt^2F_0F_1^2+$$
$$1.20763\times10^{-05}ql^4\Delta qtF_0F_1^2-2.71846\times10^{-05}ql^3\Delta qtF_0F_1^2-8.97118\times10^{-05}ql^2\Delta qtF_0F_1^2+$$
$$7.43231\times10^{-05}ql\Delta qtF_0F_1^2+1.24876\times10^{-06}\Delta qtF_0F_1^2-0.000128108ql^4F_0F_1^2-$$
$$8.4927\times10^{-05}ql^3F_0F_1^2+3.30144\times10^{-05}ql^2F_0F_1^2-7.65939\times10^{-05}qlF_0F_1^2-$$
$$7.65103\times10^{-05}F_0F_1^2-0.000162142\Delta B^8F_1^2-7.48053\times10^{-05}\Delta B^7F_1^2+0.000258772\Delta B^6F_1^2-$$
$$0.000163947\Delta B^5F_1^2+2.84473\times10^{-06}\Delta qt^4\Delta B^4F_1^2+0.000142094\Delta qt^3\Delta B^4F_1^2+$$
$$1.44878\times10^{-05}ql^2\Delta qt^2\Delta B^4F_1^2+2.10344\times10^{-05}ql\Delta qt^2\Delta B^4F_1^2-2.29685\times$$
$$10^{-05}\Delta qt^2\Delta B^4F_1^2+3.48528\times10^{-05}ql^2\Delta qt\Delta B^4F_1^2+0.000113454ql\Delta qt\Delta B^4F_1^2-$$
$$0.000158144\Delta qt\Delta B^4F_1^2+1.0249\times10^{-05}ql^4\Delta B^4F_1^2-5.98253\times10^{-05}ql^3\Delta B^4F_1^2-$$
$$5.83724\times10^{-05}ql^2\Delta B^4F_1^2+5.36195\times10^{-05}ql\Delta B^4F_1^2-0.00013164\Delta B^4F_1^2-$$
$$0.000105298\Delta qt^4\Delta B^3F_1^2+7.20359\times10^{-06}\Delta qt^3\Delta B^3F_1^2-0.000127111ql^2\Delta qt^2\Delta B^3F_1^2-$$
$$0.000104829ql\Delta qt^2\Delta B^3F_1^2+1.55525\times10^{-05}\Delta qt^2\Delta B^3F_1^2-9.98905\times10^{-06}ql^2\Delta qt\Delta B^3F_1^2-$$
$$0.00018437ql\Delta qt\Delta B^3F_1^2+0.000120882\Delta qt\Delta B^3F_1^2-9.08097\times10^{-05}ql^4\Delta B^3F_1^2+$$
$$3.48355\times10^{-05}ql^3\Delta B^3F_1^2+0.000225532ql^2\Delta B^3F_1^2+0.000219175ql\Delta B^3F_1^2+$$
$$0.00056996\Delta B^3F_1^2-0.000158954ql^2\Delta qt^4\Delta B^2F_1^2-0.000160863ql\Delta qt^4\Delta B^2F_1^2-$$
$$6.9186\times10^{-05}\Delta qt^4\Delta B^2F_1^2-3.06209\times10^{-05}ql^2\Delta qt^3\Delta B^2F_1^2-3.89542\times10^{-05}ql\Delta qt^3\Delta B^2F_1^2-$$
$$5.36983\times10^{-05}\Delta qt^3\Delta B^2F_1^2-2.23236\times10^{-05}ql^4\Delta qt^2\Delta B^2F_1^2-0.00014924ql^3\Delta qt^2\Delta B^2F_1^2+$$
$$0.000128686ql^2\Delta qt^2\Delta B^2F_1^2+0.000253855ql\Delta qt^2\Delta B^2F_1^2+4.64516\times10^{-05}\Delta qt^2\Delta B^2F_1^2-$$
$$6.83619\times10^{-05}ql^4\Delta qt\Delta B^2F_1^2+0.000223792ql^3\Delta qt\Delta B^2F_1^2+9.73707\times10^{-05}ql^2\Delta qt\Delta B^2F_1^2-$$
$$4.5034\times10^{-05}ql\Delta qt\Delta B^2F_1^2+0.000155764\Delta qt\Delta B^2F_1^2+4.45382\times10^{-05}ql^4\Delta B^2F_1^2-$$
$$0.000324639ql^3\Delta B^2F_1^2+5.63534\times10^{-05}ql^2\Delta B^2F_1^2+0.000331915ql\Delta B^2F_1^2-$$
$$0.000596463\Delta B^2F_1^2+0.000230089ql^2\Delta qt^4\Delta BF_1^2+0.00028546ql\Delta qt^4\Delta BF_1^2+$$
$$0.000219407\Delta qt^4\Delta BF_1^2+4.74702\times10^{-06}ql^2\Delta qt^3\Delta BF_1^2-0.000144416ql\Delta qt^3\Delta BF_1^2+$$

$$5.53758\times10^{-05}\Delta qt^3\Delta BF_1^2-0.000110311ql^4\Delta qt^2\Delta BF_1^2+0.000147492ql^3\Delta qt^2\Delta BF_1^2-$$
$$0.000108043ql^2\Delta qt^2\Delta BF_1^2-0.000357618ql\Delta qt^2\Delta BF_1^2-7.14666\times10^{-05}\Delta qt^2\Delta BF_1^2+$$
$$0.000150651ql^4\Delta qt\Delta BF_1^2-0.000568838ql^3\Delta qt\Delta BF_1^2-0.000170911ql^2\Delta qt\Delta BF_1^2+$$
$$0.000563304ql\Delta qt\Delta BF_1^2-0.000324137\Delta qt\Delta BF_1^2+6.91435\times10^{-05}ql^4\Delta BF_1^2+$$
$$0.000490095ql^3\Delta BF_1^2-0.000235171ql^2\Delta BF_1^2-0.000735715ql\Delta BF_1^2-0.000118487\Delta BF_1^2-$$
$$4.15061\times10^{-05}\Delta qt^8F_1^2-0.000285343\Delta qt^7F_1^2+1.10668\times10^{-05}\Delta qt^6F_1^2-1.9295\times$$
$$10^{-05}\Delta qt^5F_1^2-8.11377\times10^{-05}ql^4\Delta qt^4F_1^2-1.73548\times10^{-05}ql^3\Delta qt^4F_1^2-8.89081\times$$
$$10^{-05}ql^2\Delta qt^4F_1^2-8.22208\times10^{-05}ql\Delta qt^4F_1^2-1.32427\times10^{-05}\Delta qt^4F_1^2+6.99442\times$$
$$10^{-05}ql^4\Delta qt^3F_1^2+7.67991\times10^{-05}ql^3\Delta qt^3F_1^2-3.70912\times10^{-05}ql^2\Delta qt^3F_1^2-$$
$$2.67831\times10^{-05}ql\Delta qt^3F_1^2-0.000277742\Delta qt^3F_1^2+2.60015\times10^{-05}ql^4\Delta qt^2F_1^2-$$
$$0.000116825ql^3\Delta qt^2F_1^2+3.116\times10^{-05}ql^2\Delta qt^2F_1^2+0.000133931ql\Delta qt^2F_1^2-0.000183342\Delta qt^2F_1^2-$$
$$0.000226457ql^4\Delta qtF_1^2+1.50103\times10^{-05}ql^3\Delta qtF_1^2+0.000105986ql^2\Delta qtF_1^2+2.67706\times$$
$$10^{-05}ql\Delta qtF_1^2+0.00092892\Delta qtF_1^2-0.000156346ql^8F_1^2+0.000153573ql^7F_1^2-$$
$$1.63514\times10^{-05}ql^6F_1^2-0.000139956ql^5F_1^2+9.2718\times10^{-05}ql^4F_1^2-0.000103746ql^3F_1^2-$$
$$0.000205011ql^2F_1^2-0.000133197qlF_1^2-9.0313\times10^{-05}F_1^2+3.28112\times10^{-05}F_0^8F_1-$$
$$0.00018575F_0^7F_1-2.15514\times10^{-05}F_0^6F_1-0.000159175F_0^5F_1+9.45646\times10^{-05}\Delta B^4F_0^4F_1-$$
$$9.13192\times10^{-05}\Delta B^3F_0^4F_1+7.74136\times10^{-06}\Delta qt^2\Delta B^2F_0^4F_1+9.89049\times10^{-05}\Delta qt\Delta B^2F_0^4F_1+$$
$$0.000113713ql^2\Delta B^2F_0^4F_1-3.43267\times10^{-05}ql\Delta B^2F_0^4F_1+0.000112845\Delta B^2F_0^4F_1-$$
$$2.11148\times10^{-05}\Delta qt^2\Delta BF_0^4F_1-0.000195424\Delta qt\Delta BF_0^4F_1-0.00021585ql^2\Delta BF_0^4F_1+$$
$$7.05046\times10^{-05}ql\Delta BF_0^4F_1+1.50606\times10^{-05}\Delta BF_0^4F_1-0.00011667\Delta qt^4F_0^4F_1-$$
$$4.35157\times10^{-05}\Delta qt^3F_0^4F_1+3.87146\times10^{-05}ql^2\Delta qt^2F_0^4F_1+3.64361\times10^{-05}ql\Delta qt^2F_0^4F_1+$$
$$0.000123706\Delta qt^2F_0^4F_1+1.39113\times10^{-05}ql^2\Delta qtF_0^4F_1+0.000128297ql\Delta qtF_0^4F_1+$$
$$0.000173729\Delta qtF_0^4F_1-0.000117962ql^4F_0^4F_1-2.39464\times10^{-05}ql^3F_0^4F_1+0.000239404ql^2F_0^4F_1+$$
$$4.5918\times10^{-05}qlF_0^4F_1+0.000270268F_0^4F_1+0.000160038\Delta B^4F_0^3F_1-0.000200513\Delta B^3F_0^3F_1-$$
$$8.62838\times10^{-05}\Delta qt^2\Delta B^2F_0^3F_1+0.000184039\Delta qt\Delta B^2F_0^3F_1-4.72721\times10^{-05}ql^2\Delta B^2F_0^3F_1+$$
$$8.87596\times10^{-05}ql\Delta B^2F_0^3F_1+1.66153\times10^{-05}\Delta B^2F_0^3F_1+0.00029302\Delta qt^2\Delta BF_0^3F_1-$$
$$0.000357636\Delta qt\Delta BF_0^3F_1+0.000240265ql^2\Delta BF_0^3F_1-0.000121319ql\Delta BF_0^3F_1+$$
$$0.000547985\Delta BF_0^3F_1-0.000119222\Delta qt^4F_0^3F_1-5.43741\times10^{-05}\Delta qt^3F_0^3F_1+$$
$$9.2672\times10^{-05}ql^2\Delta qt^2F_0^3F_1+8.2427\times10^{-05}ql\Delta qt^2F_0^3F_1+3.68642\times10^{-05}\Delta qt^2F_0^3F_1-$$
$$7.55501\times10^{-05}ql^2\Delta qtF_0^3F_1-0.000169636ql\Delta qtF_0^3F_1+0.00027631\Delta qtF_0^3F_1+$$
$$9.36292\times10^{-05}ql^4F_0^3F_1+2.02719\times10^{-05}ql^3F_0^3F_1+3.14469\times10^{-05}ql^2F_0^3F_1+$$
$$2.08655\times10^{-05}qlF_0^3F_1-4.99327\times10^{-05}F_0^3F_1-3.18824\times10^{-05}\Delta qt^2\Delta B^4F_0^2F_1-$$
$$0.000131163\Delta qt\Delta B^4F_0^2F_1-9.07491\times10^{-05}ql^2\Delta B^4F_0^2F_1-1.60762\times10^{-05}ql\Delta B^4F_0^2F_1-$$
$$0.000126715\Delta B^4F_0^2F_1+6.17264\times10^{-05}\Delta qt^2\Delta B^3F_0^2F_1+0.000141882\Delta qt\Delta B^3F_0^2F_1+$$
$$0.000162537ql^2\Delta B^3F_0^2F_1+2.88022\times10^{-05}ql\Delta B^3F_0^2F_1+3.23423\times10^{-05}\Delta B^3F_0^2F_1+$$
$$5.06039\times10^{-05}\Delta qt^4\Delta B^2F_0^2F_1+0.000208613\Delta qt^3\Delta B^2F_0^2F_1-3.87982\times10^{-05}ql^2\Delta qt^2\Delta B^2F_0^2F_1+$$

$$4.3587\times10^{-05}ql\Delta qt^2\Delta B^2F_0^2F_1-7.82437\times10^{-05}\Delta qt^2\Delta B^2F_0^2F_1+0.000220401ql^2\Delta qt\Delta B^2F_0^2F_1+$$
$$0.000240731ql\Delta qt\Delta B^2F_0^2F_1-0.00010524\Delta qt\Delta B^2F_0^2F_1+3.08921\times10^{-05}ql^4\Delta B^2F_0^2F_1+$$
$$0.000122379ql^3\Delta B^2F_0^2F_1-7.89069\times10^{-05}ql^2\Delta B^2F_0^2F_1+5.20763\times10^{-05}ql\Delta B^2F_0^2F_1-$$
$$6.41122\times10^{-05}\Delta B^2F_0^2F_1-8.02309\times10^{-05}\Delta qt^4\Delta BF_0^2F_1-0.000346998\Delta qt^3\Delta BF_0^2F_1+$$
$$2.29055\times10^{-05}ql^2\Delta qt^2\Delta BF_0^2F_1-2.74053\times10^{-05}ql\Delta qt^2\Delta BF_0^2F_1-4.25421\times$$
$$10^{-05}\Delta qt^2\Delta BF_0^2F_1-0.000420327ql^2\Delta qt\Delta BF_0^2F_1-0.000205355ql\Delta qt\Delta BF_0^2F_1+$$
$$0.00010954\Delta qt\Delta BF_0^2F_1-5.04491\times10^{-05}ql^4\Delta BF_0^2F_1-0.000203544ql^3\Delta BF_0^2F_1-$$
$$0.000131878ql^2\Delta BF_0^2F_1-5.14511\times10^{-05}ql\Delta BF_0^2F_1-1.86549\times10^{-05}\Delta BF_0^2F_1+$$
$$1.71732\times10^{-05}ql^2\Delta qt^4F_0^2F_1+4.71936\times10^{-05}ql\Delta qt^4F_0^2F_1-3.29518\times10^{-06}\Delta qt^4F_0^2F_1-$$
$$1.83957\times10^{-05}ql^2\Delta qt^3F_0^2F_1-0.000262011ql\Delta qt^3F_0^2F_1+0.000162924\Delta qt^3F_0^2F_1-$$
$$2.52318\times10^{-05}ql^4\Delta qt^2F_0^2F_1+2.48294\times10^{-05}ql^3\Delta qt^2F_0^2F_1-3.934\times10^{-05}ql^2\Delta qt^2F_0^2F_1+$$
$$0.000140906ql\Delta qt^2F_0^2F_1+0.000106111\Delta qt^2F_0^2F_1-0.000137668ql^4\Delta qtF_0^2F_1-$$
$$6.28547\times10^{-06}ql^3\Delta qtF_0^2F_1+0.000182232ql^2\Delta qtF_0^2F_1+0.0002927ql\Delta qtF_0^2F_1+$$
$$3.16572\times10^{-05}\Delta qtF_0^2F_1-3.89656\times10^{-05}ql^4F_0^2F_1+0.000162008ql^3F_0^2F_1+$$
$$1.92084\times10^{-05}ql^2F_0^2F_1-8.94125\times10^{-05}qlF_0^2F_1+0.000259068F_0^2F_1-0.000218287\Delta qt^2\Delta B^4F_0F_1+$$
$$6.41221\times10^{-05}\Delta qt\Delta B^4F_0F_1-0.000159444ql^2\Delta B^4F_0F_1-0.00013521ql\Delta B^4F_0F_1-$$
$$0.000407793\Delta B^4F_0F_1+0.000413488\Delta qt^2\Delta B^3F_0F_1-0.000320299\Delta qt\Delta B^3F_0F_1+$$
$$0.000291237ql^2\Delta B^3F_0F_1+0.000201619ql\Delta B^3F_0F_1+0.000609896\Delta B^3F_0F_1+$$
$$0.000112906\Delta qt^4\Delta B^2F_0F_1+0.000238157\Delta qt^3\Delta B^2F_0F_1-3.62297\times10^{-05}ql^2\Delta qt^2\Delta B^2F_0F_1-$$
$$0.000218879ql\Delta qt^2\Delta B^2F_0F_1-0.000154714\Delta qt^2\Delta B^2F_0F_1-3.16025\times10^{-05}ql^2\Delta qt\Delta B^2F_0F_1+$$
$$0.000202951ql\Delta qt\Delta B^2F_0F_1-0.000189078\Delta qt\Delta B^2F_0F_1+7.99244\times10^{-05}ql^4\Delta B^2F_0F_1-$$
$$1.02053\times10^{-05}ql^3\Delta B^2F_0F_1-0.000137334ql^2\Delta B^2F_0F_1-0.00027156ql\Delta B^2F_0F_1-$$
$$0.000270915\Delta B^2F_0F_1+0.000128121\Delta qt^4\Delta BF_0F_1-0.00039538\Delta qt^3\Delta BF_0F_1-$$
$$0.000156346ql^2\Delta qt^2\Delta BF_0F_1+0.000276776ql\Delta qt^2\Delta BF_0F_1-0.000314352\Delta qt^2\Delta BF_0F_1+$$
$$0.00020509ql^2\Delta qt\Delta BF_0F_1-0.000170786ql\Delta qt\Delta BF_0F_1+0.000726576\Delta qt\Delta BF_0F_1+$$
$$1.36336\times10^{-05}ql^4\Delta BF_0F_1-1.42757\times10^{-05}ql^3\Delta BF_0F_1-0.000207951ql^2\Delta BF_0F_1+$$
$$0.000434053ql\Delta BF_0F_1-0.000897152\Delta BF_0F_1+5.64186\times10^{-05}ql^2\Delta qt^4F_0F_1-$$
$$0.000117079ql\Delta qt^4F_0F_1+0.000112256\Delta qt^4F_0F_1+1.51001\times10^{-05}ql^2\Delta qt^3F_0F_1-$$
$$9.87671\times10^{-05}ql\Delta qt^3F_0F_1+0.000203676\Delta qt^3F_0F_1+7.84638\times10^{-05}ql^4\Delta qt^2F_0F_1+$$
$$0.000135717ql^3\Delta qt^2F_0F_1+7.4455\times10^{-06}ql^2\Delta qt^2F_0F_1-0.000171409ql\Delta qt^2F_0F_1+$$
$$3.58926\times10^{-05}\Delta qt^2F_0F_1-0.000128145ql^4\Delta qtF_0F_1-0.000130074ql^3\Delta qtF_0F_1+$$
$$0.000126553ql^2\Delta qtF_0F_1+0.000693288ql\Delta qtF_0F_1-0.000121373\Delta qtF_0F_1-0.000107458ql^4F_0F_1+$$
$$0.000168636ql^3F_0F_1-0.000119686ql^2F_0F_1-0.000232811qlF_0F_1+0.000701499F_0F_1-$$
$$0.000156075\Delta B^8F_1+9.01551\times10^{-05}\Delta B^7F_1-8.11741\times10^{-05}\Delta B^6F_1+6.79178\times$$
$$10^{-07}\Delta B^5F_1-3.14007\times10^{-05}\Delta qt^4\Delta B^4F_1+0.000105685\Delta qt^3\Delta B^4F_1-3.85079\times$$
$$10^{-05}ql^2\Delta qt^2\Delta B^4F_1-0.000109857ql\Delta qt^2\Delta B^4F_1-7.27349\times10^{-05}\Delta qt^2\Delta B^4F_1-$$

$3.07487\times10^{-05}ql^2\Delta qt\Delta B^4F_1+0.000218922ql\Delta qt\Delta B^4F_1-0.000234977\Delta qt\Delta B^4F_1-$
$6.58122\times10^{-05}ql^4\Delta B^4F_1-0.000280622ql^3\Delta B^4F_1-0.000235263ql^2\Delta B^4F_1+$
$0.000149974ql\Delta B^4F_1-6.18216\times10^{-05}\Delta B^4F_1+5.21471\times10^{-05}\Delta qt^4\Delta B^3F_1-$
$8.52997\times10^{-05}\Delta qt^3\Delta B^3F_1+0.000139895ql^2\Delta qt^2\Delta B^3F_1+0.000212534ql\Delta qt^2\Delta B^3F_1+$
$1.09501\times10^{-05}\Delta qt^2\Delta B^3F_1+6.02265\times10^{-05}ql^2\Delta qt\Delta B^3F_1-0.000165682ql\Delta qt\Delta B^3F_1+$
$0.000233075\Delta qt\Delta B^3F_1+6.74264\times10^{-05}ql^4\Delta B^3F_1+0.000497595ql^3\Delta B^3F_1+$
$0.000405494ql^2\Delta B^3F_1-0.00028194ql\Delta B^3F_1-0.00026199\Delta B^3F_1+0.000155423ql^2\Delta qt^4\Delta B^2F_1+$
$2.87742\times10^{-05}ql\Delta qt^4\Delta B^2F_1+0.000181945\Delta qt^4\Delta B^2F_1-0.000103839ql^2\Delta qt^3\Delta B^2F_1-$
$0.000198929ql\Delta qt^3\Delta B^2F_1-7.28204\times10^{-06}\Delta qt^3\Delta B^2F_1+8.66813\times10^{-05}ql^4\Delta qt^2\Delta B^2F_1-$
$0.000138752ql^3\Delta qt^2\Delta B^2F_1-0.000114288ql^2\Delta qt^2\Delta B^2F_1+0.00013099ql\Delta qt^2\Delta B^2F_1-$
$0.000159735\Delta qt^2\Delta B^2F_1-0.000172378ql^4\Delta qt\Delta B^2F_1+0.000487712ql^3\Delta qt\Delta B^2F_1+$
$0.000217166ql^2\Delta qt\Delta B^2F_1-0.000160228ql\Delta qt\Delta B^2F_1+7.72373\times10^{-07}\Delta qt\Delta B^2F_1+$
$0.000133319ql^4\Delta B^2F_1-0.000205367ql^3\Delta B^2F_1-0.000564629ql^2\Delta B^2F_1+0.000810376ql\Delta B^2F_1+$
$0.000558867\Delta B^2F_1-0.000192652ql^2\Delta qt^4\Delta BF_1+0.000174833ql\Delta qt^4\Delta BF_1-$
$0.000263602\Delta qt^4\Delta BF_1+7.31126\times10^{-05}ql^2\Delta qt^3\Delta BF_1-0.000178115ql\Delta qt^3\Delta BF_1+$
$6.77581\times10^{-05}\Delta qt^3\Delta BF_1-0.000211338ql^4\Delta qt^2\Delta BF_1+0.000220988ql^3\Delta qt^2\Delta BF_1-$
$8.903\times10^{-05}ql^2\Delta qt^2\Delta BF_1-0.00038295ql\Delta qt^2\Delta BF_1+0.000236667\Delta qt^2\Delta BF_1+$
$0.000231592ql^4\Delta qt\Delta BF_1-0.00100414ql^3\Delta qt\Delta BF_1-0.000543998ql^2\Delta qt\Delta BF_1+$
$0.00123902ql\Delta qt\Delta BF_1+5.13378\times10^{-05}\Delta qt\Delta BF_1-0.0002966ql^4\Delta BF_1+0.000141384ql^3\Delta BF_1+$
$0.000656474ql^2\Delta BF_1-0.00191251ql\Delta BF_1-0.00138252\Delta BF_1+3.03159\times10^{-05}\Delta qt^8F_1+$
$0.000111867\Delta qt^7F_1+0.000190144\Delta qt^6F_1+7.18311\times10^{-05}\Delta qt^5F_1+1.3048\times$
$10^{-06}ql^4\Delta qt^4F_1+1.25722\times10^{-06}ql^3\Delta qt^4F_1+0.000123831ql^2\Delta qt^4F_1+0.00012782ql\Delta qt^4F_1+$
$0.000244611\Delta qt^4F_1+1.6379\times10^{-05}ql^4\Delta qt^3F_1+3.61185\times10^{-05}ql^3\Delta qt^3F_1-$
$5.48585\times10^{-05}ql^2\Delta qt^3F_1-0.000386054ql\Delta qt^3F_1-0.000209316\Delta qt^3F_1+8.41949\times$
$10^{-05}ql^4\Delta qt^2F_1-4.2575\times10^{-05}ql^3\Delta qt^2F_1-5.87096\times10^{-05}ql^2\Delta qt^2F_1+0.000181408ql\Delta qt^2F_1-$
$0.000238766\Delta qt^2F_1-0.000106127ql^4\Delta qtF_1+0.000683062ql^3\Delta qtF_1+0.000488954ql^2\Delta qtF_1-$
$0.000378626ql\Delta qtF_1-2.72315\times10^{-05}\Delta qtF_1-0.000251916ql^8F_1-0.00037798ql^7F_1-$
$0.000150533ql^6F_1-0.000108607ql^5F_1+0.00018764ql^4F_1+3.11339\times10^{-05}ql^3F_1-$
$0.000104654ql^2F_1+0.00201383qlF_1+0.00190027F_1-5.2741\times10^{-05}\Delta B^2F_0^8-$
$8.59695\times10^{-05}\Delta BF_0^8-0.000197199\Delta qt^2F_0^8+3.53831\times10^{-05}\Delta qtF_0^8-0.000117695ql^2F_0^8+$
$0.000123664qlF_0^8-0.000275129F_0^8-4.73354\times10^{-05}\Delta B^2F_0^7+8.99336\times10^{-05}\Delta BF_0^7-$
$0.000106721\Delta qt^2F_0^7+3.06345\times10^{-05}\Delta qtF_0^7-9.67874\times10^{-05}ql^2F_0^7+1.04212\times$
$10^{-05}qlF_0^7-0.000172208F_0^7+6.92044\times10^{-05}\Delta B^2F_0^6+0.000128112\Delta BF_0^6+$
$0.000155362\Delta qt^2F_0^6-4.52479\times10^{-05}\Delta qtF_0^6+2.36421\times10^{-05}ql^2F_0^6-0.000164638qlF_0^6+$
$0.000205378F_0^6-0.000114148\Delta B^2F_0^5+0.00039244\Delta BF_0^5+6.0453\times10^{-05}\Delta qt^2F_0^5+$
$0.000159965\Delta qtF_0^5+5.93101\times10^{-05}ql^2F_0^5-6.63619\times10^{-05}qlF_0^5+9.65608\times$

$10^{-05}F_0^5+2.80767\times10^{-05}\Delta qt^2\Delta B^4F_0^4+8.48678\times10^{-05}\Delta qt\Delta B^4F_0^4+0.000110729ql^2\Delta B^4F_0^4+$
$4.4007\times10^{-05}ql\Delta B^4F_0^4+0.000118024\Delta B^4F_0^4-0.00010386\Delta qt^2\Delta B^3F_0^4-0.000147058\Delta qt\Delta B^3F_0^4-$
$0.000334432ql^2\Delta B^3F_0^4-0.000350058ql\Delta B^3F_0^4-0.000275009\Delta B^3F_0^4-0.000195587\Delta qt^4\Delta B^2F_0^4+$
$0.000199319\Delta qt^3\Delta B^2F_0^4-5.60329\times10^{-05}ql^2\Delta qt^2\Delta B^2F_0^4-1.22659\times10^{-05}ql\Delta qt^2\Delta B^2F_0^4-$
$3.40556\times10^{-05}\Delta qt^2\Delta B^2F_0^4-3.31099\times10^{-05}ql^2\Delta qt\Delta B^2F_0^4-4.24792\times10^{-06}ql\Delta qt\Delta B^2F_0^4-$
$4.24681\times10^{-05}\Delta qt\Delta B^2F_0^4-0.000330994ql^4\Delta B^2F_0^4-0.00010208ql^3\Delta B^2F_0^4+$
$2.8828\times10^{-05}ql^2\Delta B^2F_0^4+0.000113781ql\Delta B^2F_0^4+0.000431608\Delta B^2F_0^4+0.000380366\Delta qt^4\Delta BF_0^4-$
$0.000252748\Delta qt^3\Delta BF_0^4+0.000210234ql^2\Delta qt^2\Delta BF_0^4+0.00024608ql\Delta qt^2\Delta BF_0^4+$
$0.000259178\Delta qt^2\Delta BF_0^4+9.36046\times10^{-05}ql^2\Delta qt\Delta BF_0^4+7.35902\times10^{-05}ql\Delta qt\Delta BF_0^4+$
$0.000169009\Delta qt\Delta BF_0^4+0.000510243ql^4\Delta BF_0^4+0.000202012ql^3\Delta BF_0^4+0.000282753ql^2\Delta BF_0^4+$
$0.00025375ql\Delta BF_0^4-0.000462602\Delta BF_0^4-3.54033\times10^{-05}ql^2\Delta qt^4F_0^4-9.91655\times$
$10^{-05}ql\Delta qt^4F_0^4-0.000173555\Delta qt^4F_0^4+6.25744\times10^{-06}ql^2\Delta qt^3F_0^4-0.000165532ql\Delta qt^3F_0^4+$
$0.000119911\Delta qt^3F_0^4-6.99456\times10^{-05}ql^4\Delta qt^2F_0^4-0.000171518ql^3\Delta qt^2F_0^4-$
$9.64406\times10^{-05}ql^2\Delta qt^2F_0^4+6.82583\times10^{-05}ql\Delta qt^2F_0^4-8.02396\times10^{-05}\Delta qt^2F_0^4-$
$8.52277\times10^{-05}ql^4\Delta qtF_0^4-6.19954\times10^{-05}ql^3\Delta qtF_0^4-4.74703\times10^{-05}ql^2\Delta qtF_0^4+$
$0.000212418ql\Delta qtF_0^4+4.05012\times10^{-05}\Delta qtF_0^4-0.000371236ql^4F_0^4-0.000151582ql^3F_0^4-$
$0.00016836ql^2F_0^4+3.24065\times10^{-05}qlF_0^4+0.000223197F_0^4+9.16361\times10^{-05}\Delta qt^2\Delta B^4F_0^3-$
$6.2406\times10^{-05}\Delta qt\Delta B^4F_0^3+9.32136\times10^{-05}ql^2\Delta B^4F_0^3+0.000178129ql\Delta B^4F_0^3+$
$0.00028045\Delta B^4F_0^3-0.00013123\Delta qt^2\Delta B^3F_0^3+0.000125876\Delta qt\Delta B^3F_0^3-0.00026754ql^2\Delta B^3F_0^3-$
$0.000385776ql\Delta B^3F_0^3-0.000354458\Delta B^3F_0^3-3.08316\times10^{-05}\Delta qt^4\Delta B^2F_0^3+$
$0.000151987\Delta qt^3\Delta B^2F_0^3-6.73673\times10^{-05}ql^2\Delta qt^2\Delta B^2F_0^3-0.00018676ql\Delta qt^2\Delta B^2F_0^3-$
$7.66353\times10^{-05}\Delta qt^2\Delta B^2F_0^3+0.000280317ql^2\Delta qt\Delta B^2F_0^3+0.000372886ql\Delta qt\Delta B^2F_0^3+$
$8.10831\times10^{-05}\Delta qt\Delta B^2F_0^3+1.35933\times10^{-05}ql^4\Delta B^2F_0^3-0.000302851ql^3\Delta B^2F_0^3-$
$2.46262\times10^{-05}ql^2\Delta B^2F_0^3+0.000206734ql\Delta B^2F_0^3+5.30058\times10^{-05}\Delta B^2F_0^3+$
$9.5078\times10^{-05}\Delta qt^4\Delta BF_0^3-0.000363218\Delta qt^3\Delta BF_0^3+0.000215909ql^2\Delta qt^2\Delta BF_0^3+$
$0.000398379ql\Delta qt^2\Delta BF_0^3+0.000344011\Delta qt^2\Delta BF_0^3-0.000658579ql^2\Delta qt\Delta BF_0^3-$
$0.0010462ql\Delta qt\Delta BF_0^3-0.000410076\Delta qt\Delta BF_0^3+0.000207101ql^4\Delta BF_0^3+0.000247613ql^3\Delta BF_0^3+$
$0.000441942ql^2\Delta BF_0^3+0.00035709ql\Delta BF_0^3-0.00032412\Delta BF_0^3-3.81342\times10^{-05}ql^2\Delta qt^4F_0^3-$
$0.000140432ql\Delta qt^4F_0^3-1.587\times10^{-05}\Delta qt^4F_0^3-0.000126538ql^2\Delta qt^3F_0^3-4.5656\times$
$10^{-05}ql\Delta qt^3F_0^3+0.00010373\Delta qt^3F_0^3+5.59828\times10^{-05}ql^4\Delta qt^2F_0^3-0.000121365ql^3\Delta qt^2F_0^3-$
$0.00012522ql^2\Delta qt^2F_0^3-0.000167773ql\Delta qt^2F_0^3-0.000354973\Delta qt^2F_0^3-1.67172\times$
$10^{-05}ql^4\Delta qtF_0^3-7.08416\times10^{-05}ql^3\Delta qtF_0^3+0.000393808ql^2\Delta qtF_0^3+0.000584161ql\Delta qtF_0^3-$
$5.58875\times10^{-05}\Delta qtF_0^3+0.000185513ql^4F_0^3-0.000240618ql^3F_0^3-0.000310779ql^2F_0^3+$
$6.84401\times10^{-05}qlF_0^3-0.000606536F_0^3-6.43734\times10^{-05}\Delta B^8F_0^2-8.81199\times$
$10^{-05}\Delta B^7F_0^2+0.000185225\Delta B^6F_0^2-3.45622\times10^{-06}\Delta B^5F_0^2-3.79683\times10^{-05}\Delta qt^4\Delta B^4F_0^2+$
$0.000140422\Delta qt^3\Delta B^4F_0^2-3.16393\times10^{-05}ql^2\Delta qt^2\Delta B^4F_0^2+3.72363\times10^{-06}ql\Delta qt^2\Delta B^4F_0^2+$

$$6.80165\times10^{-05}\Delta qt^2\Delta B^4F_0^2+1.21128\times10^{-05}ql^2\Delta qt\Delta B^4F_0^2+0.000135801ql\Delta qt\Delta B^4F_0^2-$$
$$0.000206551\Delta qt\Delta B^4F_0^2+0.000107807ql^4\Delta B^4F_0^2+9.38098\times10^{-05}ql^3\Delta B^4F_0^2-$$
$$7.669\times10^{-05}ql^2\Delta B^4F_0^2-0.000257618ql\Delta B^4F_0^2+6.66935\times10^{-06}\Delta B^4F_0^2-$$
$$7.2317\times10^{-05}\Delta qt^4\Delta B^3F_0^2+1.3357\times10^{-05}\Delta qt^3\Delta B^3F_0^2+7.01314\times10^{-05}ql^2\Delta qt^2\Delta B^3F_0^2+$$
$$8.83862\times10^{-05}ql\Delta qt^2\Delta B^3F_0^2-4.44549\times10^{-05}\Delta qt^2\Delta B^3F_0^2+1.4784\times10^{-05}ql^2\Delta qt\Delta B^3F_0^2-$$
$$0.000374102ql\Delta qt\Delta B^3F_0^2+1.41095\times10^{-05}\Delta qt\Delta B^3F_0^2-0.000257699ql^4\Delta B^3F_0^2-$$
$$0.000110376ql^3\Delta B^3F_0^2+0.000144298ql^2\Delta B^3F_0^2+0.0004948ql\Delta B^3F_0^2+0.000265006\Delta B^3F_0^2-$$
$$0.000191655ql^2\Delta qt^4\Delta B^2F_0^2-0.00019759ql\Delta qt^4\Delta B^2F_0^2-6.62182\times10^{-05}\Delta qt^4\Delta B^2F_0^2+$$
$$4.47738\times10^{-05}ql^2\Delta qt^3\Delta B^2F_0^2+0.000100199ql\Delta qt^3\Delta B^2F_0^2-9.33408\times10^{-06}\Delta qt^3\Delta B^2F_0^2-$$
$$8.43787\times10^{-05}ql^4\Delta qt^2\Delta B^2F_0^2-0.000195871ql^3\Delta qt^2\Delta B^2F_0^2-5.16969\times10^{-05}ql^2\Delta qt^2\Delta B^2F_0^2+$$
$$4.70792\times10^{-05}ql\Delta qt^2\Delta B^2F_0^2-1.80742\times10^{-05}\Delta qt^2\Delta B^2F_0^2-6.21239\times10^{-05}ql^4\Delta qt\Delta B^2F_0^2+$$
$$7.5995\times10^{-05}ql^3\Delta qt\Delta B^2F_0^2+1.64598\times10^{-05}ql^2\Delta qt\Delta B^2F_0^2+4.38201\times10^{-05}ql\Delta qt\Delta B^2F_0^2+$$
$$0.00010047\Delta qt\Delta B^2F_0^2+5.89757\times10^{-05}ql^4\Delta B^2F_0^2-0.000192384ql^3\Delta B^2F_0^2-$$
$$3.42368\times10^{-05}ql^2\Delta B^2F_0^2+0.000225081ql\Delta B^2F_0^2-0.00029669\Delta B^2F_0^2+0.000209786ql^2\Delta qt^4\Delta BF_0^2+$$
$$0.000314427ql\Delta qt^4\Delta BF_0^2+0.0001844\Delta qt^4\Delta BF_0^2-8.63157\times10^{-05}ql^2\Delta qt^3\Delta BF_0^2-$$
$$0.000478685ql\Delta qt^3\Delta BF_0^2-8.65534\times10^{-05}\Delta qt^3\Delta BF_0^2+0.000208802ql^4\Delta qt^2\Delta BF_0^2+$$
$$0.000363977ql^3\Delta qt^2\Delta BF_0^2+3.54891\times10^{-05}ql^2\Delta qt^2\Delta BF_0^2-0.000206315ql\Delta qt^2\Delta BF_0^2+$$
$$0.000108691\Delta qt^2\Delta BF_0^2-6.48825\times10^{-05}ql^4\Delta qt\Delta BF_0^2-0.000631046ql^3\Delta qt\Delta BF_0^2-$$
$$0.000142871ql^2\Delta qt\Delta BF_0^2+0.000933083ql\Delta qt\Delta BF_0^2+3.00213\times10^{-05}\Delta qt\Delta BF_0^2+$$
$$5.46617\times10^{-05}ql^4\Delta BF_0^2+0.000562701ql^3\Delta BF_0^2+7.46434\times10^{-06}ql^2\Delta BF_0^2-$$
$$0.000911618ql\Delta BF_0^2-1.10541\times10^{-05}\Delta BF_0^2-3.27416\times10^{-05}\Delta qt^8F_0^2-0.000205275\Delta qt^7F_0^2+$$
$$3.38702\times10^{-05}\Delta qt^6F_0^2+2.30659\times10^{-05}\Delta qt^5F_0^2-6.29047\times10^{-05}ql^4\Delta qt^4F_0^2+$$
$$2.27231\times10^{-06}ql^3\Delta qt^4F_0^2-9.30304\times10^{-05}ql^2\Delta qt^4F_0^2-0.000114734ql\Delta qt^4F_0^2+$$
$$0.000122444\Delta qt^4F_0^2+3.86402\times10^{-05}ql^4\Delta qt^3F_0^2-2.44687\times10^{-05}ql^3\Delta qt^3F_0^2-$$
$$2.88259\times10^{-05}ql^2\Delta qt^3F_0^2+0.000199945ql\Delta qt^3F_0^2-0.000182864\Delta qt^3F_0^2-8.21286\times$$
$$10^{-05}ql^4\Delta qt^2F_0^2-0.000298691ql^3\Delta qt^2F_0^2-3.49421\times10^{-05}ql^2\Delta qt^2F_0^2+0.000155835ql\Delta qt^2F_0^2-$$
$$0.000242198\Delta qt^2F_0^2-7.52979\times10^{-05}ql^4\Delta qtF_0^2+0.00019522ql^3\Delta qtF_0^2+2.90558\times$$
$$10^{-05}ql^2\Delta qtF_0^2-0.000275578ql\Delta qtF_0^2+0.000482911\Delta qtF_0^2-0.000219277ql^8F_0^2+$$
$$5.87329\times10^{-05}ql^7F_0^2+9.78857\times10^{-05}ql^6F_0^2+0.000100481ql^5F_0^2+0.000133769ql^4F_0^2-$$
$$0.00059146ql^3F_0^2-4.39654\times10^{-05}ql^2F_0^2+0.000474691qlF_0^2-0.000505332F_0^2+$$
$$0.000150863\Delta B^8F_0-0.000273683\Delta B^7F_0-3.88641\times10^{-06}\Delta B^6F_0+0.000221198\Delta B^5F_0-$$
$$0.000106047\Delta qt^4\Delta B^4F_0+0.0001403\Delta qt^3\Delta B^4F_0-3.93433\times10^{-05}ql^2\Delta qt^2\Delta B^4F_0-$$
$$0.000111045ql\Delta qt^2\Delta B^4F_0-0.000214988\Delta qt^2\Delta B^4F_0+0.000194676ql^2\Delta qt\Delta B^4F_0+$$
$$0.000333815ql\Delta qt\Delta B^4F_0+0.000135294\Delta qt\Delta B^4F_0+4.9781\times10^{-05}ql^4\Delta B^4F_0+$$
$$5.97634\times10^{-05}ql^3\Delta B^4F_0-0.00020065ql^2\Delta B^4F_0-0.000490302ql\Delta B^4F_0-0.000263378\Delta B^4F_0+$$
$$0.000172101\Delta qt^4\Delta B^3F_0-0.000135381\Delta qt^3\Delta B^3F_0+0.000291057ql^2\Delta qt^2\Delta B^3F_0+$$

$$0.000393678 ql \Delta qt^2 \Delta B^3 F_0 + 0.000401412 \Delta qt^2 \Delta B^3 F_0 - 0.000564196 ql^2 \Delta qt \Delta B^3 F_0 -$$
$$0.000902961 ql \Delta qt \Delta B^3 F_0 - 0.00044465 \Delta qt \Delta B^3 F_0 + 8.82359 \times 10^{-05} ql^4 \Delta B^3 F_0 -$$
$$0.000527192 ql^3 \Delta B^3 F_0 + 0.00051971 ql^2 \Delta B^3 F_0 + 0.00137498 ql \Delta B^3 F_0 - 0.000301827 \Delta B^3 F_0 -$$
$$1.37882 \times 10^{-05} ql^2 \Delta qt^4 \Delta B^2 F_0 - 1.56792 \times 10^{-05} ql \Delta qt^4 \Delta B^2 F_0 - 5.96197 \times$$
$$10^{-07} \Delta qt^4 \Delta B^2 F_0 + 1.2456 \times 10^{-05} ql^2 \Delta qt^3 \Delta B^2 F_0 + 0.000218688 ql \Delta qt^3 \Delta B^2 F_0 +$$
$$4.39431 \times 10^{-05} \Delta qt^3 \Delta B^2 F_0 - 0.000222369 ql^4 \Delta qt^2 \Delta B^2 F_0 - 0.000255366 ql^3 \Delta qt^2 \Delta B^2 F_0 -$$
$$7.05612 \times 10^{-05} ql^2 \Delta qt^2 \Delta B^2 F_0 + 0.000355401 ql \Delta qt^2 \Delta B^2 F_0 + 0.000310851 \Delta qt^2 \Delta B^2 F_0 +$$
$$0.000211264 ql^4 \Delta qt \Delta B^2 F_0 + 0.000354655 ql^3 \Delta qt \Delta B^2 F_0 + 1.38368 \times 10^{-05} ql^2 \Delta qt \Delta B^2 F_0 -$$
$$0.000221827 ql \Delta qt \Delta B^2 F_0 - 0.000159002 \Delta qt \Delta B^2 F_0 - 0.000275615 ql^4 \Delta B^2 F_0 +$$
$$0.000269838 ql^3 \Delta B^2 F_0 - 0.000182685 ql^2 \Delta B^2 F_0 - 0.000103795 ql \Delta B^2 F_0 + 0.0012259 \Delta B^2 F_0 -$$
$$2.66927 \times 10^{-05} ql^2 \Delta qt^4 \Delta B F_0 + 0.000162987 ql \Delta qt^4 \Delta B F_0 - 0.00022601 \Delta qt^4 \Delta B F_0 -$$
$$0.00013087 ql^2 \Delta qt^3 \Delta B F_0 - 0.00058152 ql \Delta qt^3 \Delta B F_0 + 1.9347 \times 10^{-05} \Delta qt^3 \Delta B F_0 +$$
$$0.000189613 ql^4 \Delta qt^2 \Delta B F_0 + 0.000803963 ql^3 \Delta qt^2 \Delta B F_0 - 0.000231672 ql^2 \Delta qt^2 \Delta B F_0 -$$
$$0.00140208 ql \Delta qt^2 \Delta B F_0 - 0.000864435 \Delta qt^2 \Delta B F_0 - 0.000456354 ql^4 \Delta qt \Delta B F_0 -$$
$$0.000784042 ql^3 \Delta qt \Delta B F_0 + 0.000913012 ql^2 \Delta qt \Delta B F_0 + 0.00185222 ql \Delta qt \Delta B F_0 +$$
$$0.000648262 \Delta qt \Delta B F_0 + 0.000220221 ql^4 \Delta B F_0 + 0.000309872 ql^3 \Delta B F_0 - 0.000229923 ql^2 \Delta B F_0 -$$
$$0.00176254 ql \Delta B F_0 - 0.00247321 \Delta B F_0 - 3.86444 \times 10^{-05} \Delta qt^8 F_0 - 0.000205448 \Delta qt^7 F_0 +$$
$$4.18135 \times 10^{-05} \Delta qt^6 F_0 + 0.000210994 \Delta qt^5 F_0 - 2.60768 \times 10^{-05} ql^4 \Delta qt^4 F_0 -$$
$$1.87467 \times 10^{-05} ql^3 \Delta qt^4 F_0 + 1.79599 \times 10^{-05} ql^2 \Delta qt^4 F_0 + 0.000116717 ql \Delta qt^4 F_0 +$$
$$6.04722 \times 10^{-05} \Delta qt^4 F_0 + 0.000249272 ql^4 \Delta qt^3 F_0 + 0.00021727 ql^3 \Delta qt^3 F_0 + 0.000152048 ql^2 \Delta qt^3 F_0 +$$
$$4.68862 \times 10^{-05} ql \Delta qt^3 F_0 - 0.000448782 \Delta qt^3 F_0 - 0.000223736 ql^4 \Delta qt^2 F_0 - 8.71044 \times$$
$$10^{-05} ql^3 \Delta qt^2 F_0 + 9.65038 \times 10^{-05} ql^2 \Delta qt^2 F_0 + 0.000404013 ql \Delta qt^2 F_0 + 0.00066015 \Delta qt^2 F_0 +$$
$$4.90151 \times 10^{-05} ql^4 \Delta qt F_0 + 0.000186076 ql^3 \Delta qt F_0 - 0.000379504 ql^2 \Delta qt F_0 - 0.000428542 ql \Delta qt F_0 +$$
$$0.000322418 \Delta qt F_0 - 0.000103433 ql^8 F_0 - 0.000192317 ql^7 F_0 - 7.84763 \times 10^{-05} ql^6 F_0 -$$
$$2.61942 \times 10^{-05} ql^5 F_0 - 0.00053312 ql^4 F_0 + 0.000395559 ql^3 F_0 - 0.000120493 ql^2 F_0 +$$
$$0.00167644 ql F_0 + 0.00454505 F_0 - 6.55797 \times 10^{-05} \Delta qt^2 \Delta B^8 + 5.18006 \times 10^{-05} \Delta qt \Delta B^8 -$$
$$0.000183888 ql^2 \Delta B^8 - 6.66352 \times 10^{-05} ql \Delta B^8 - 0.000287395 \Delta B^8 - 0.000262145 \Delta qt^2 \Delta B^7 -$$
$$5.57782 \times 10^{-05} \Delta qt \Delta B^7 + 8.0444 \times 10^{-05} ql^2 \Delta B^7 + 0.000151114 ql \Delta B^7 - 0.000165533 \Delta B^7 +$$
$$0.000404586 \Delta qt^2 \Delta B^6 - 0.000117837 \Delta qt \Delta B^6 + 2.65762 \times 10^{-05} ql^2 \Delta B^6 - 0.000248674 ql \Delta B^6 +$$
$$0.000484541 \Delta B^6 - 0.000185335 \Delta qt^2 \Delta B^5 + 1.75698 \times 10^{-06} \Delta qt \Delta B^5 - 7.2966 \times$$
$$10^{-05} ql^2 \Delta B^5 - 0.000426014 ql \Delta B^5 - 0.000608746 \Delta B^5 + 4.71269 \times 10^{-05} ql^2 \Delta qt^4 \Delta B^4 +$$
$$9.90105 \times 10^{-05} ql \Delta qt^4 \Delta B^4 - 3.57382 \times 10^{-05} \Delta qt^4 \Delta B^4 + 0.000129917 ql^2 \Delta qt^3 \Delta B^4 -$$
$$0.000177134 ql \Delta qt^3 \Delta B^4 + 0.000230597 \Delta qt^3 \Delta B^4 - 4.53723 \times 10^{-06} ql^4 \Delta qt^2 \Delta B^4 +$$
$$2.58749 \times 10^{-05} ql^3 \Delta qt^2 \Delta B^4 - 4.1452 \times 10^{-06} ql^2 \Delta qt^2 \Delta B^4 - 9.17309 \times 10^{-05} ql \Delta qt^2 \Delta B^4 +$$
$$3.15775 \times 10^{-05} \Delta qt^2 \Delta B^4 + 0.00015826 ql^4 \Delta qt \Delta B^4 - 3.29288 \times 10^{-05} ql^3 \Delta qt \Delta B^4 -$$
$$4.96333 \times 10^{-05} ql^2 \Delta qt \Delta B^4 + 0.000173263 ql \Delta qt \Delta B^4 - 0.000613023 \Delta qt \Delta B^4 -$$

$$6.73349\times10^{-05}ql^4\Delta B^4-0.000111285ql^3\Delta B^4-0.000344691ql^2\Delta B^4+0.000569809ql\Delta B^4+$$
$$0.000989295\Delta B^4-0.000245507ql^2\Delta qt^4\Delta B^3-0.000297999ql\Delta qt^4\Delta B^3-0.000109302\Delta qt^4\Delta B^3-$$
$$0.000127ql^2\Delta qt^3\Delta B^3+0.000244249ql\Delta qt^3\Delta B^3-0.000120389\Delta qt^3\Delta B^3-0.000124313ql^4\Delta qt^2\Delta B^3-$$
$$0.000247203ql^3\Delta qt^2\Delta B^3+0.000166817ql^2\Delta qt^2\Delta B^3+0.000560476ql\Delta qt^2\Delta B^3+$$
$$0.00037811\Delta qt^2\Delta B^3-0.000190002ql^4\Delta qt\Delta B^3+7.92679\times10^{-05}ql^3\Delta qt\Delta B^3-$$
$$9.08993\times10^{-05}ql^2\Delta qt\Delta B^3-4.59519\times10^{-05}ql\Delta qt\Delta B^3+0.000912115\Delta qt\Delta B^3-$$
$$7.99321\times10^{-05}ql^4\Delta B^3+0.000153816ql^3\Delta B^3+0.000921984ql^2\Delta B^3-0.00117288ql\Delta B^3-$$
$$0.00192004\Delta B^3-0.000179318\Delta qt^8\Delta B^2+0.000160337\Delta qt^7\Delta B^2+6.63862\times$$
$$10^{-05}\Delta qt^6\Delta B^2-4.04301\times10^{-05}\Delta qt^5\Delta B^2-0.000254464ql^4\Delta qt^4\Delta B^2-9.55125\times$$
$$10^{-05}ql^3\Delta qt^4\Delta B^2-3.02451\times10^{-05}ql^2\Delta qt^4\Delta B^2+7.69339\times10^{-05}ql\Delta qt^4\Delta B^2+$$
$$0.000382109\Delta qt^4\Delta B^2+0.000265741ql^4\Delta qt^3\Delta B^2+7.35708\times10^{-05}ql^3\Delta qt^3\Delta B^2-$$
$$2.17618\times10^{-05}ql^2\Delta qt^3\Delta B^2-0.000125089ql\Delta qt^3\Delta B^2-0.000491448\Delta qt^3\Delta B^2-$$
$$2.75182\times10^{-05}ql^4\Delta qt^2\Delta B^2-0.000228685ql^3\Delta qt^2\Delta B^2-0.00011998ql^2\Delta qt^2\Delta B^2-$$
$$0.000125177ql\Delta qt^2\Delta B^2-0.000643597\Delta qt^2\Delta B^2-0.00023594ql^4\Delta qt\Delta B^2+0.000270591ql^3\Delta qt\Delta B^2-$$
$$0.000148361ql^2\Delta qt\Delta B^2-0.00161477ql\Delta qt\Delta B^2-0.00089148\Delta qt\Delta B^2-0.000141134ql^8\Delta B^2+$$
$$0.00040993ql^7\Delta B^2+0.00026276ql^6\Delta B^2+6.78085\times10^{-05}ql^5\Delta B^2+0.000596241ql^4\Delta B^2-$$
$$0.000542758ql^3\Delta B^2-0.000263051ql^2\Delta B^2+0.00533575ql\Delta B^2+0.00503697\Delta B^2+$$
$$0.000223761\Delta qt^8\Delta B-0.00039347\Delta qt^7\Delta B-0.000166305\Delta qt^6\Delta B-3.24971\times$$
$$10^{-05}\Delta qt^5\Delta B+0.000473171ql^4\Delta qt^4\Delta B+0.000284001ql^3\Delta qt^4\Delta B+0.000235958ql^2\Delta qt^4\Delta B+$$
$$0.000124124ql\Delta qt^4\Delta B-0.000465947\Delta qt^4\Delta B-0.000618466ql^4\Delta qt^3\Delta B-0.000236678ql^3\Delta qt^3\Delta B+$$
$$8.18646\times10^{-05}ql^2\Delta qt^3\Delta B+0.000151977ql\Delta qt^3\Delta B+0.00121855\Delta qt^3\Delta B+$$
$$0.000122174ql^4\Delta qt^2\Delta B+0.00042485ql^3\Delta qt^2\Delta B+5.63849\times10^{-05}ql^2\Delta qt^2\Delta B-$$
$$5.43893\times10^{-05}ql\Delta qt^2\Delta B+0.000757395\Delta qt^2\Delta B+0.000746547ql^4\Delta qt\Delta B-$$
$$0.000639995ql^3\Delta qt\Delta B+0.000654805ql^2\Delta qt\Delta B+0.00437121ql\Delta qt\Delta B+0.0018448\Delta qt\Delta B-$$
$$0.000106196ql^8\Delta B-0.000710075ql^7\Delta B-0.000430784ql^6\Delta B-0.000168713ql^5\Delta B-$$
$$0.000908575ql^4\Delta B+0.000638193ql^3\Delta B-0.000471842ql^2\Delta B-0.0114916ql\Delta B-$$
$$0.0114296\Delta B-5.22747\times10^{-05}ql^2\Delta qt^8+1.45783\times10^{-05}ql\Delta qt^8-0.00033418\Delta qt^8-$$
$$0.000154631ql^2\Delta qt^7+0.000151886ql\Delta qt^7-0.000125299\Delta qt^7-4.89834\times10^{-05}ql^2\Delta qt^6-$$
$$7.13843\times10^{-05}ql\Delta qt^6+4.43747\times10^{-05}\Delta qt^6-5.48264\times10^{-05}ql^2\Delta qt^5-0.000305599ql\Delta qt^5-$$
$$8.50055\times10^{-05}\Delta qt^5-0.000277301ql^4\Delta qt^4-0.000131579ql^3\Delta qt^4+1.90158\times$$
$$10^{-05}ql^2\Delta qt^4+0.000116326ql\Delta qt^4+0.000414899\Delta qt^4+0.000286375ql^4\Delta qt^3+$$
$$3.09619\times10^{-05}ql^3\Delta qt^3-0.000119755ql^2\Delta qt^3+0.000383241ql\Delta qt^3-0.000669632\Delta qt^3-$$
$$0.000220448ql^8\Delta qt^2+5.23092\times10^{-05}ql^7\Delta qt^2+5.12563\times10^{-06}ql^6\Delta qt^2+9.62054\times$$
$$10^{-05}ql^5\Delta qt^2+0.00011172ql^4\Delta qt^2-0.000147831ql^3\Delta qt^2+6.02448\times10^{-05}ql^2\Delta qt^2-$$
$$0.000289859ql\Delta qt^2+1.67262\times10^{-05}\Delta qt^2+5.80634\times10^{-05}ql^8\Delta qt+3.82173\times$$
$$10^{-06}ql^7\Delta qt+2.43975\times10^{-05}ql^6\Delta qt+1.1973\times10^{-05}ql^5\Delta qt-0.000178314ql^4\Delta qt+$$

$0.000703461ql^3\Delta qt - 0.00025309ql^2\Delta qt - 0.00450494ql\Delta qt - 0.00228695\Delta qt -$
$0.000514976ql^8 + 0.000276427ql^7 + 0.000149689ql^6 + 0.000169212ql^5 + 0.000617858ql^4 -$
$0.000476485ql^3 + 0.000465312ql^2 + 0.0102646ql + 0.0125835.$

C.2 Standard deviation

The polynomial to calculate standard deviation, in Chebyshev form, is:

$-5.76469 \times 10^{-05}F_0^2F_1^8 - 3.89611 \times 10^{-05}F_0F_1^8 - 7.39867 \times 10^{-05}\Delta B^2F_1^8 +$
$0.000104341\Delta BF_1^8 - 1.34491 \times 10^{-05}\Delta qt^2F_1^8 - 2.45106 \times 10^{-05}\Delta qtF_1^8 - 4.21925 \times$
$10^{-05}ql^2F_1^8 - 3.09858 \times 10^{-05}qlF_1^8 - 0.000133626F_1^8 - 5.8007 \times 10^{-05}F_0^2F_1^7 +$
$4.19637 \times 10^{-06}F_0F_1^7 - 0.000229788\Delta B^2F_1^7 + 0.00037428\Delta BF_1^7 - 1.69008 \times$
$10^{-06}\Delta qt^2F_1^7 - 2.00511 \times 10^{-05}\Delta qtF_1^7 - 5.53338 \times 10^{-05}ql^2F_1^7 - 6.36116 \times$
$10^{-05}qlF_1^7 - 0.000297763F_1^7 - 3.64445 \times 10^{-05}F_0^2F_1^6 - 5.88191 \times 10^{-05}F_0F_1^6 -$
$0.000233064\Delta B^2F_1^6 + 0.000356055\Delta BF_1^6 - 4.13626 \times 10^{-05}\Delta qt^2F_1^6 + 3.37682 \times$
$10^{-05}\Delta qtF_1^6 - 3.01362 \times 10^{-05}ql^2F_1^6 - 3.34204 \times 10^{-05}qlF_1^6 - 0.000314655F_1^6 -$
$2.62656 \times 10^{-06}F_0^2F_1^5 - 4.37038 \times 10^{-05}F_0F_1^5 - 0.0001363\Delta B^2F_1^5 + 0.000282323\Delta BF_1^5 +$
$1.01158 \times 10^{-05}\Delta qt^2F_1^5 + 1.04444 \times 10^{-05}\Delta qtF_1^5 - 1.13789 \times 10^{-05}ql^2F_1^5 -$
$3.63485 \times 10^{-05}qlF_1^5 - 0.000153781F_1^5 + 2.89317 \times 10^{-05}\Delta B^2F_0^4F_1^4 - 1.69983 \times$
$10^{-06}\Delta BF_0^4F_1^4 + 2.24029 \times 10^{-05}\Delta qt^2F_0^4F_1^4 + 3.44753 \times 10^{-06}\Delta qtF_0^4F_1^4 + 2.31986 \times$
$10^{-05}ql^2F_0^4F_1^4 - 4.07579 \times 10^{-06}qlF_0^4F_1^4 + 2.99419 \times 10^{-05}F_0^4F_1^4 + 8.2897 \times$
$10^{-05}\Delta B^2F_0^3F_1^4 - 0.000155472\Delta BF_0^3F_1^4 + 5.79929 \times 10^{-06}\Delta qt^2F_0^3F_1^4 - 5.21192 \times$
$10^{-06}\Delta qtF_0^3F_1^4 - 1.71817 \times 10^{-05}ql^2F_0^3F_1^4 - 3.97892 \times 10^{-05}qlF_0^3F_1^4 + 7.47462 \times$
$10^{-05}F_0^3F_1^4 - 4.98067 \times 10^{-05}\Delta B^4F_0^2F_1^4 + 0.000119923\Delta B^3F_0^2F_1^4 + 0.000119159\Delta qt^2\Delta B^2F_0^2F_1^4 -$
$8.9729 \times 10^{-05}\Delta qt\Delta B^2F_0^2F_1^4 + 0.000114084ql^2\Delta B^2F_0^2F_1^4 + 0.000107627ql\Delta B^2F_0^2F_1^4 +$
$0.000112904\Delta B^2F_0^2F_1^4 - 0.00019822\Delta qt^2\Delta BF_0^2F_1^4 + 0.000183178\Delta qt\Delta BF_0^2F_1^4 -$
$0.000194048ql^2\Delta BF_0^2F_1^4 - 0.000193268ql\Delta BF_0^2F_1^4 - 0.00031928\Delta BF_0^2F_1^4 - 8.01973 \times$
$10^{-06}\Delta qt^4F_0^2F_1^4 + 1.12547 \times 10^{-06}\Delta qt^3F_0^2F_1^4 + 2.17789 \times 10^{-05}ql^2\Delta qt^2F_0^2F_1^4 +$
$2.26444 \times 10^{-05}ql\Delta qt^2F_0^2F_1^4 + 0.000115546\Delta qt^2F_0^2F_1^4 - 1.13119 \times 10^{-05}ql^2\Delta qtF_0^2F_1^4 -$
$2.29865 \times 10^{-05}ql\Delta qtF_0^2F_1^4 - 0.000104523\Delta qtF_0^2F_1^4 + 1.40009 \times 10^{-05}ql^4F_0^2F_1^4 -$
$4.53669 \times 10^{-06}ql^3F_0^2F_1^4 + 0.000119684ql^2F_0^2F_1^4 + 0.000125452qlF_0^2F_1^4 + 0.000161452F_0^2F_1^4 +$
$4.73879 \times 10^{-05}\Delta B^4F_0F_1^4 - 8.46909 \times 10^{-05}\Delta B^3F_0F_1^4 - 5.60842 \times 10^{-05}\Delta qt^2\Delta B^2F_0F_1^4 +$
$4.65246 \times 10^{-05}\Delta qt\Delta B^2F_0F_1^4 - 4.75893 \times 10^{-05}ql^2\Delta B^2F_0F_1^4 + 2.31728 \times 10^{-05}ql\Delta B^2F_0F_1^4 -$
$7.55121 \times 10^{-05}\Delta B^2F_0F_1^4 + 0.000107801\Delta qt^2\Delta BF_0F_1^4 - 5.68105 \times 10^{-05}\Delta qt\Delta BF_0F_1^4 +$
$8.23757 \times 10^{-05}ql^2\Delta BF_0F_1^4 - 3.88383 \times 10^{-05}ql\Delta BF_0F_1^4 + 0.00023781\Delta BF_0F_1^4 -$
$6.81094 \times 10^{-07}\Delta qt^4F_0F_1^4 - 4.49427 \times 10^{-06}\Delta qt^3F_0F_1^4 - 7.00067 \times 10^{-06}ql^2\Delta qt^2F_0F_1^4 -$

$$1.91376\times10^{-06}ql\Delta qt^2F_0F_1^4-6.60033\times10^{-05}\Delta qt^2F_0F_1^4+2.68784\times10^{-05}ql^2\Delta qtF_0F_1^4+$$
$$1.3892\times10^{-05}ql\Delta qtF_0F_1^4+3.92558\times10^{-05}\Delta qtF_0F_1^4+2.3144\times10^{-05}ql^4F_0F_1^4+$$
$$4.20577\times10^{-05}ql^3F_0F_1^4-3.2294\times10^{-05}ql^2F_0F_1^4+1.46954\times10^{-05}qlF_0F_1^4-$$
$$0.000147216F_0F_1^4-0.000103825\Delta qt^2\Delta B^4F_1^4+8.88301\times10^{-05}\Delta qt\Delta B^4F_1^4-$$
$$0.000146482ql^2\Delta B^4F_1^4-0.00016978ql\Delta B^4F_1^4-0.000180819\Delta B^4F_1^4+0.000241453\Delta qt^2\Delta B^3F_1^4-$$
$$0.000210793\Delta qt\Delta B^3F_1^4+0.000297357ql^2\Delta B^3F_1^4+0.000322228ql\Delta B^3F_1^4+0.000394106\Delta B^3F_1^4-$$
$$1.00379\times10^{-05}\Delta qt^4\Delta B^2F_1^4+0.00010449\Delta qt^3\Delta B^2F_1^4+0.000243199ql^2\Delta qt^2\Delta B^2F_1^4+$$
$$0.000256516ql\Delta qt^2\Delta B^2F_1^4+0.000126609\Delta qt^2\Delta B^2F_1^4-0.000189583ql^2\Delta qt\Delta B^2F_1^4-$$
$$0.00020335ql\Delta qt\Delta B^2F_1^4-0.000180279\Delta qt\Delta B^2F_1^4+5.83686\times10^{-06}ql^4\Delta B^2F_1^4-$$
$$0.000150291ql^3\Delta B^2F_1^4+9.37868\times10^{-05}ql^2\Delta B^2F_1^4+0.000255514ql\Delta B^2F_1^4+$$
$$9.25027\times10^{-05}\Delta B^2F_1^4+2.85915\times10^{-05}\Delta qt^4\Delta BF_1^4-0.000195801\Delta qt^3\Delta BF_1^4-$$
$$0.000419747ql^2\Delta qt^2\Delta BF_1^4-0.000469685ql\Delta qt^2\Delta BF_1^4-0.000477625\Delta qt^2\Delta BF_1^4+$$
$$0.000352061ql^2\Delta qt\Delta BF_1^4+0.000366499ql\Delta qt\Delta BF_1^4+0.000548793\Delta qt\Delta BF_1^4+$$
$$1.23689\times10^{-05}ql^4\Delta BF_1^4+0.000245219ql^3\Delta BF_1^4-0.000468609ql^2\Delta BF_1^4-0.000754259ql\Delta BF_1^4-$$
$$0.000622186\Delta BF_1^4-1.08378\times10^{-05}ql^2\Delta qt^4F_1^4-1.1667\times10^{-05}ql\Delta qt^4F_1^4-$$
$$1.87618\times10^{-05}\Delta qt^4F_1^4+3.03889\times10^{-05}ql^2\Delta qt^3F_1^4+4.84518\times10^{-05}ql\Delta qt^3F_1^4+$$
$$0.000123679\Delta qt^3F_1^4-2.27291\times10^{-05}ql^4\Delta qt^2F_1^4-7.34813\times10^{-05}ql^3\Delta qt^2F_1^4+$$
$$0.000214888ql^2\Delta qt^2F_1^4+0.000319654ql\Delta qt^2F_1^4+0.000243999\Delta qt^2F_1^4+6.73551\times$$
$$10^{-05}ql^4\Delta qtF_1^4+0.000130464ql^3\Delta qtF_1^4-0.00017571ql^2\Delta qtF_1^4-0.00032765ql\Delta qtF_1^4-$$
$$0.00035074\Delta qtF_1^4-2.11268\times10^{-05}ql^4F_1^4-0.000192869ql^3F_1^4+0.000225788ql^2F_1^4+$$
$$0.00047947qlF_1^4+0.000303259F_1^4-0.000112725\Delta B^2F_0^4F_1^3+0.000230768\Delta BF_0^4F_1^3+$$
$$1.19847\times10^{-05}\Delta qt^2F_0^4F_1^3-7.50594\times10^{-06}\Delta qtF_0^4F_1^3-2.15291\times10^{-06}ql^2F_0^4F_1^3-$$
$$1.29415\times10^{-05}qlF_0^4F_1^3-0.000105954F_0^4F_1^3+4.30101\times10^{-05}\Delta B^2F_0^3F_1^3-5.96998\times$$
$$10^{-05}\Delta BF_0^3F_1^3-2.00194\times10^{-06}\Delta qt^2F_0^3F_1^3-2.37989\times10^{-05}\Delta qtF_0^3F_1^3+5.11902\times$$
$$10^{-06}ql^2F_0^3F_1^3-2.36719\times10^{-05}qlF_0^3F_1^3+2.30318\times10^{-05}F_0^3F_1^3-8.64048\times$$
$$10^{-05}\Delta B^4F_0^2F_1^3+0.000150126\Delta B^3F_0^2F_1^3+7.66338\times10^{-05}\Delta qt^2\Delta B^2F_0^2F_1^3-$$
$$0.000141505\Delta qt\Delta B^2F_0^2F_1^3-6.57478\times10^{-05}ql^2\Delta B^2F_0^2F_1^3-0.00011625ql\Delta B^2F_0^2F_1^3-$$
$$3.6588\times10^{-06}\Delta B^2F_0^2F_1^3-0.000144707\Delta qt^2\Delta BF_0^2F_1^3+0.000348073\Delta qt\Delta BF_0^2F_1^3+$$
$$0.000199491ql^2\Delta BF_0^2F_1^3+0.000334648ql\Delta BF_0^2F_1^3-0.000110101\Delta BF_0^2F_1^3+1.98006\times$$
$$10^{-06}\Delta qt^4F_0^2F_1^3-1.8515\times10^{-05}\Delta qt^3F_0^2F_1^3+2.49915\times10^{-05}ql^2\Delta qt^2F_0^2F_1^3+$$
$$4.64904\times10^{-05}ql\Delta qt^2F_0^2F_1^3+9.0962\times10^{-05}\Delta qt^2F_0^2F_1^3-1.50123\times10^{-05}ql^2\Delta qtF_0^2F_1^3-$$
$$5.78981\times10^{-05}ql\Delta qtF_0^2F_1^3-0.000158057\Delta qtF_0^2F_1^3+5.97031\times10^{-06}ql^4F_0^2F_1^3-$$
$$2.56641\times10^{-05}ql^3F_0^2F_1^3-8.16303\times10^{-05}ql^2F_0^2F_1^3-0.000109923qlF_0^2F_1^3+$$
$$7.68032\times10^{-05}F_0^2F_1^3-2.76954\times10^{-05}\Delta B^4F_0F_1^3+3.76842\times10^{-05}\Delta B^3F_0F_1^3-$$
$$7.25241\times10^{-05}\Delta qt^2\Delta B^2F_0F_1^3+0.000102924\Delta qt\Delta B^2F_0F_1^3-0.000109279ql^2\Delta B^2F_0F_1^3-$$
$$4.86152\times10^{-05}ql\Delta B^2F_0F_1^3-0.000111447\Delta B^2F_0F_1^3+0.000138732\Delta qt^2\Delta BF_0F_1^3-$$

$$0.000130896\Delta qt\Delta BF_0F_1^3+0.000205551ql^2\Delta BF_0F_1^3+0.000126043ql\Delta BF_0F_1^3+$$
$$0.000174382\Delta BF_0F_1^3+2.17577\times10^{-06}\Delta qt^4F_0F_1^3-5.37963\times10^{-05}\Delta qt^3F_0F_1^3-$$
$$2.48376\times10^{-05}ql^2\Delta qt^2F_0F_1^3-1.32765\times10^{-05}ql\Delta qt^2F_0F_1^3-7.33025\times10^{-05}\Delta qt^2F_0F_1^3+$$
$$4.87724\times10^{-05}ql^2\Delta qtF_0F_1^3+4.09322\times10^{-06}ql\Delta qtF_0F_1^3+0.000151477\Delta qtF_0F_1^3+$$
$$5.33067\times10^{-05}ql^4F_0F_1^3+3.97755\times10^{-05}ql^3F_0F_1^3-0.000100764ql^2F_0F_1^3-$$
$$5.86837\times10^{-05}qlF_0F_1^3-7.74816\times10^{-05}F_0F_1^3-0.000165254\Delta qt^2\Delta B^4F_1^3+$$
$$0.00026794\Delta qt\Delta B^4F_1^3-4.46074\times10^{-06}ql^2\Delta B^4F_1^3+0.000110607ql\Delta B^4F_1^3-$$
$$0.000156162\Delta B^4F_1^3+0.00028193\Delta qt^2\Delta B^3F_1^3-0.000559391\Delta qt\Delta B^3F_1^3-1.46892\times$$
$$10^{-05}ql^2\Delta B^3F_1^3-0.000187452ql\Delta B^3F_1^3+0.00028609\Delta B^3F_1^3-2.64347\times10^{-05}\Delta qt^4\Delta B^2F_1^3+$$
$$0.0002548\Delta qt^3\Delta B^2F_1^3+7.36621\times10^{-07}ql^2\Delta qt^2\Delta B^2F_1^3-0.0001399ql\Delta qt^2\Delta B^2F_1^3-$$
$$5.46446\times10^{-05}\Delta qt^2\Delta B^2F_1^3-0.000106866ql^2\Delta qt\Delta B^2F_1^3+0.000210771ql\Delta qt\Delta B^2F_1^3-$$
$$9.00607\times10^{-05}\Delta qt\Delta B^2F_1^3-8.61131\times10^{-05}ql^4\Delta B^2F_1^3-5.99095\times10^{-05}ql^3\Delta B^2F_1^3+$$
$$7.01851\times10^{-05}ql^2\Delta B^2F_1^3+9.95155\times10^{-06}ql\Delta B^2F_1^3+0.000300636\Delta B^2F_1^3+$$
$$5.96142\times10^{-05}\Delta qt^4\Delta BF_1^3-0.00055252\Delta qt^3\Delta BF_1^3+6.09619\times10^{-05}ql^2\Delta qt^2\Delta BF_1^3+$$
$$0.000376467ql\Delta qt^2\Delta BF_1^3-0.000129848\Delta qt^2\Delta BF_1^3+0.000182136ql^2\Delta qt\Delta BF_1^3-$$
$$0.000545151ql\Delta qt\Delta BF_1^3+0.000755482\Delta qt\Delta BF_1^3+0.000152578ql^4\Delta BF_1^3+8.3229\times$$
$$10^{-05}ql^3\Delta BF_1^3-9.92058\times10^{-05}ql^2\Delta BF_1^3+0.000280434ql\Delta BF_1^3-0.000841122\Delta BF_1^3+$$
$$2.72143\times10^{-05}ql^2\Delta qt^4F_1^3+4.98824\times10^{-05}ql\Delta qt^4F_1^3-1.78659\times10^{-06}\Delta qt^4F_1^3-$$
$$7.89065\times10^{-07}ql^2\Delta qt^3F_1^3+1.10874\times10^{-06}ql\Delta qt^3F_1^3+0.000238183\Delta qt^3F_1^3-$$
$$1.36173\times10^{-05}ql^4\Delta qt^2F_1^3-2.71723\times10^{-05}ql^3\Delta qt^2F_1^3-2.52167\times10^{-05}ql^2\Delta qt^2F_1^3-$$
$$0.000126285ql\Delta qt^2F_1^3+0.000107352\Delta qt^2F_1^3+2.8543\times10^{-05}ql^4\Delta qtF_1^3+4.84556\times$$
$$10^{-05}ql^3\Delta qtF_1^3-5.48642\times10^{-05}ql^2\Delta qtF_1^3+0.000221284ql\Delta qtF_1^3-0.000356182\Delta qtF_1^3-$$
$$7.96448\times10^{-05}ql^4F_1^3-8.34201\times10^{-05}ql^3F_1^3+5.74629\times10^{-05}ql^2F_1^3-8.81896\times$$
$$10^{-05}qlF_1^3+0.000432273F_1^3-6.00017\times10^{-06}F_0^8F_1^2-5.10529\times10^{-05}F_0^7F_1^2-$$
$$1.12344\times10^{-05}F_0^6F_1^2+4.40897\times10^{-05}F_0^5F_1^2-7.39376\times10^{-05}\Delta B^4F_0^4F_1^2+$$
$$0.000124835\Delta B^3F_0^4F_1^2+7.03105\times10^{-05}\Delta qt^2\Delta B^2F_0^4F_1^2+9.58224\times10^{-05}\Delta qt\Delta B^2F_0^4F_1^2+$$
$$9.69884\times10^{-05}ql^2\Delta B^2F_0^4F_1^2+5.70915\times10^{-05}ql\Delta B^2F_0^4F_1^2+2.72407\times10^{-05}\Delta B^2F_0^4F_1^2-$$
$$0.000153909\Delta qt^2\Delta BF_0^4F_1^2-0.000221854\Delta qt\Delta BF_0^4F_1^2-0.00017874ql^2\Delta BF_0^4F_1^2-$$
$$8.39998\times10^{-05}ql\Delta BF_0^4F_1^2-0.000169359\Delta BF_0^4F_1^2-2.38183\times10^{-05}\Delta qt^4F_0^4F_1^2+$$
$$8.02268\times10^{-06}\Delta qt^3F_0^4F_1^2-1.91217\times10^{-06}ql^2\Delta qt^2F_0^4F_1^2+9.3785\times10^{-06}ql\Delta qt^2F_0^4F_1^2+$$
$$8.45418\times10^{-05}\Delta qt^2F_0^4F_1^2+1.25427\times10^{-05}ql^2\Delta qtF_0^4F_1^2+2.72563\times10^{-05}ql\Delta qtF_0^4F_1^2+$$
$$0.00010735\Delta qtF_0^4F_1^2-1.65118\times10^{-05}ql^4F_0^4F_1^2-1.77239\times10^{-05}ql^3F_0^4F_1^2+$$
$$8.83181\times10^{-05}ql^2F_0^4F_1^2+5.71181\times10^{-05}qlF_0^4F_1^2+0.000127123F_0^4F_1^2+4.938\times$$
$$10^{-05}\Delta B^4F_0^3F_1^2-4.69313\times10^{-05}\Delta B^3F_0^3F_1^2+0.000101038\Delta qt^2\Delta B^2F_0^3F_1^2-$$
$$3.70786\times10^{-05}\Delta qt\Delta B^2F_0^3F_1^2+0.000107524ql^2\Delta B^2F_0^3F_1^2+0.000196713ql\Delta B^2F_0^3F_1^2+$$
$$0.000190006\Delta B^2F_0^3F_1^2-0.000188085\Delta qt^2\Delta BF_0^3F_1^2+0.000101463\Delta qt\Delta BF_0^3F_1^2-$$

$$0.000157633ql^2\Delta BF_0^3F_1^2-0.000334751ql\Delta BF_0^3F_1^2-0.000305593\Delta BF_0^3F_1^2+9.0045\times 10^{-06}\Delta qt^4F_0^3F_1^2-3.66425\times 10^{-05}\Delta qt^3F_0^3F_1^2+1.25336\times 10^{-05}ql^2\Delta qt^2F_0^3F_1^2+5.97158\times 10^{-06}ql\Delta qt^2F_0^3F_1^2+9.87478\times 10^{-05}\Delta qt^2F_0^3F_1^2+4.40551\times 10^{-05}ql^2\Delta qtF_0^3F_1^2+3.38601\times 10^{-05}ql\Delta qtF_0^3F_1^2-3.88198\times 10^{-06}\Delta qtF_0^3F_1^2+1.31722\times 10^{-06}ql^4F_0^3F_1^2-1.27679\times 10^{-05}ql^3F_0^3F_1^2+7.03166\times 10^{-05}ql^2F_0^3F_1^2+0.000155704qlF_0^3F_1^2+6.16812\times 10^{-05}F_0^3F_1^2+6.63975\times 10^{-06}\Delta qt^2\Delta B^4F_0^2F_1^2-4.15122\times 10^{-05}\Delta qt\Delta B^4F_0^2F_1^2-1.35915\times 10^{-05}ql^2\Delta B^4F_0^2F_1^2+4.35281\times 10^{-05}ql\Delta B^4F_0^2F_1^2+6.68203\times 10^{-05}\Delta B^4F_0^2F_1^2-7.71876\times 10^{-07}\Delta qt^2\Delta B^3F_0^2F_1^2+4.767\times 10^{-05}\Delta qt\Delta B^3F_0^2F_1^2+3.66547\times 10^{-06}ql^2\Delta B^3F_0^2F_1^2-0.000103452ql\Delta B^3F_0^2F_1^2-0.000114927\Delta B^3F_0^2F_1^2+8.96348\times 10^{-06}\Delta qt^4\Delta B^2F_0^2F_1^2-9.51754\times 10^{-06}\Delta qt^3\Delta B^2F_0^2F_1^2+5.30129\times 10^{-05}ql^2\Delta qt^2\Delta B^2F_0^2F_1^2+3.54438\times 10^{-05}ql\Delta qt^2\Delta B^2F_0^2F_1^2-1.93481\times 10^{-05}\Delta qt^2\Delta B^2F_0^2F_1^2+2.04048\times 10^{-05}ql^2\Delta qt\Delta B^2F_0^2F_1^2+4.29538\times 10^{-05}ql\Delta qt\Delta B^2F_0^2F_1^2+4.27621\times 10^{-05}\Delta qt\Delta B^2F_0^2F_1^2+0.000149539ql^4\Delta B^2F_0^2F_1^2+0.000155964ql^3\Delta B^2F_0^2F_1^2+1.46549\times 10^{-05}ql^2\Delta B^2F_0^2F_1^2-5.68354\times 10^{-05}ql\Delta B^2F_0^2F_1^2-7.32845\times 10^{-05}\Delta B^2F_0^2F_1^2-1.90452\times 10^{-05}\Delta qt^4\Delta BF_0^2F_1^2+3.94115\times 10^{-05}\Delta qt^3\Delta BF_0^2F_1^2-8.02427\times 10^{-05}ql^2\Delta qt^2\Delta BF_0^2F_1^2-7.37721\times 10^{-05}ql\Delta qt^2\Delta BF_0^2F_1^2+1.12103\times 10^{-05}\Delta qt^2\Delta BF_0^2F_1^2-3.8735\times 10^{-05}ql^2\Delta qt\Delta BF_0^2F_1^2-2.39321\times 10^{-05}ql\Delta qt\Delta BF_0^2F_1^2-9.10616\times 10^{-05}\Delta qt\Delta BF_0^2F_1^2-0.000216173ql^4\Delta BF_0^2F_1^2-0.000249581ql^3\Delta BF_0^2F_1^2-5.38073\times 10^{-06}ql^2\Delta BF_0^2F_1^2+0.000184959ql\Delta BF_0^2F_1^2+0.000185117\Delta BF_0^2F_1^2-2.13228\times 10^{-05}ql^2\Delta qt^4F_0^2F_1^2-3.68224\times 10^{-05}ql\Delta qt^4F_0^2F_1^2-9.03437\times 10^{-06}\Delta qt^4F_0^2F_1^2+9.12645\times 10^{-06}ql^2\Delta qt^3F_0^2F_1^2+1.16339\times 10^{-06}ql\Delta qt^3F_0^2F_1^2-1.17928\times 10^{-05}\Delta qt^3F_0^2F_1^2+6.75515\times 10^{-06}ql^4\Delta qt^2F_0^2F_1^2+1.78241\times 10^{-05}ql^3\Delta qt^2F_0^2F_1^2+5.0981\times 10^{-05}ql^2\Delta qt^2F_0^2F_1^2+2.45568\times 10^{-05}ql\Delta qt^2F_0^2F_1^2-2.30351\times 10^{-05}\Delta qt^2F_0^2F_1^2+2.13075\times 10^{-05}ql^4\Delta qtF_0^2F_1^2+3.77771\times 10^{-05}ql^3\Delta qtF_0^2F_1^2+8.44331\times 10^{-06}ql^2\Delta qtF_0^2F_1^2-3.10013\times 10^{-05}ql\Delta qtF_0^2F_1^2+3.27517\times 10^{-05}\Delta qtF_0^2F_1^2+0.000110203ql^4F_0^2F_1^2+9.00685\times 10^{-05}ql^3F_0^2F_1^2+2.01354\times 10^{-05}ql^2F_0^2F_1^2-4.08082\times 10^{-05}qlF_0^2F_1^2-5.04908\times 10^{-05}F_0^2F_1^2-4.6425\times 10^{-05}\Delta qt^2\Delta B^4F_0F_1^2+6.89871\times 10^{-05}\Delta qt\Delta B^4F_0F_1^2-3.74358\times 10^{-05}ql^2\Delta B^4F_0F_1^2+1.00601\times 10^{-05}ql\Delta B^4F_0F_1^2-4.706\times 10^{-05}\Delta B^4F_0F_1^2+5.37133\times 10^{-05}\Delta qt^2\Delta B^3F_0F_1^2-0.00013773\Delta qt\Delta B^3F_0F_1^2+2.92133\times 10^{-05}ql^2\Delta B^3F_0F_1^2-6.17798\times 10^{-05}ql\Delta B^3F_0F_1^2+5.12322\times 10^{-06}\Delta B^3F_0F_1^2-3.86047\times 10^{-05}\Delta qt^4\Delta B^2F_0F_1^2+6.58636\times 10^{-05}\Delta qt^3\Delta B^2F_0F_1^2+1.55177\times 10^{-05}ql^2\Delta qt^2\Delta B^2F_0F_1^2-1.08656\times 10^{-05}ql\Delta qt^2\Delta B^2F_0F_1^2-0.000128509\Delta qt^2\Delta B^2F_0F_1^2-5.44536\times 10^{-05}ql^2\Delta qt\Delta B^2F_0F_1^2-6.18809\times 10^{-05}ql\Delta qt\Delta B^2F_0F_1^2-2.94296\times 10^{-05}\Delta qt\Delta B^2F_0F_1^2+9.42776\times 10^{-05}ql^4\Delta B^2F_0F_1^2+0.000126869ql^3\Delta B^2F_0F_1^2-0.000123538ql^2\Delta B^2F_0F_1^2-0.000225819ql\Delta B^2F_0F_1^2-0.000137716\Delta B^2F_0F_1^2+7.63514\times 10^{-05}\Delta qt^4\Delta BF_0F_1^2-0.000119837\Delta qt^3\Delta BF_0F_1^2-4.05512\times 10^{-05}ql^2\Delta qt^2\Delta BF_0F_1^2-1.63395\times 10^{-05}ql\Delta qt^2\Delta BF_0F_1^2+0.000161077\Delta qt^2\Delta BF_0F_1^2+8.16923\times 10^{-05}ql^2\Delta qt\Delta BF_0F_1^2+7.07603\times 10^{-05}ql\Delta qt\Delta BF_0F_1^2+0.000136822\Delta qt\Delta BF_0F_1^2-0.000125454ql^4\Delta BF_0F_1^2-$$

$$6.30147\times10^{-05}ql^3\Delta BF_0F_1^2+0.000171577ql^2\Delta BF_0F_1^2+0.000298366ql\Delta BF_0F_1^2+$$
$$0.00020993\Delta BF_0F_1^2-1.44512\times10^{-05}ql^2\Delta qt^4F_0F_1^2-2.72802\times10^{-05}ql\Delta qt^4F_0F_1^2-$$
$$6.56849\times10^{-05}\Delta qt^4F_0F_1^2+3.43607\times10^{-05}ql^2\Delta qt^3F_0F_1^2+3.15113\times10^{-05}ql\Delta qt^3F_0F_1^2+$$
$$0.000111032\Delta qt^3F_0F_1^2+4.34532\times10^{-05}ql^4\Delta qt^2F_0F_1^2+7.598\times10^{-05}ql^3\Delta qt^2F_0F_1^2+$$
$$1.32403\times10^{-05}ql^2\Delta qt^2F_0F_1^2-6.78339\times10^{-05}ql\Delta qt^2F_0F_1^2-0.000106541\Delta qt^2F_0F_1^2-$$
$$4.61062\times10^{-06}ql^4\Delta qtF_0F_1^2-4.8493\times10^{-06}ql^3\Delta qtF_0F_1^2-0.0001017ql^2\Delta qtF_0F_1^2-$$
$$7.76209\times10^{-05}ql\Delta qtF_0F_1^2-0.000122537\Delta qtF_0F_1^2+4.81436\times10^{-05}ql^4F_0F_1^2+$$
$$9.69434\times10^{-05}ql^3F_0F_1^2-4.40228\times10^{-05}ql^2F_0F_1^2-0.000160646qlF_0F_1^2+8.69544\times$$
$$10^{-05}F_0F_1^2-0.000151808\Delta B^8F_1^2+0.000238187\Delta B^7F_1^2-0.000213419\Delta B^6F_1^2+$$
$$0.000180416\Delta B^5F_1^2-0.000127263\Delta qt^4\Delta B^4F_1^2+7.50207\times10^{-05}\Delta qt^3\Delta B^4F_1^2-$$
$$2.97268\times10^{-05}ql^2\Delta qt^2\Delta B^4F_1^2-4.11489\times10^{-05}ql\Delta qt^2\Delta B^4F_1^2-6.199\times10^{-06}\Delta qt^2\Delta B^4F_1^2-$$
$$3.09166\times10^{-05}ql^2\Delta qt\Delta B^4F_1^2+7.18602\times10^{-05}ql\Delta qt\Delta B^4F_1^2-1.86212\times10^{-05}\Delta qt\Delta B^4F_1^2-$$
$$0.000109595ql^4\Delta B^4F_1^2-0.000117783ql^3\Delta B^4F_1^2+1.27733\times10^{-05}ql^2\Delta B^4F_1^2+$$
$$8.91213\times10^{-05}ql\Delta B^4F_1^2+0.000272771\Delta B^4F_1^2+0.000184761\Delta qt^4\Delta B^3F_1^2-$$
$$9.5255\times10^{-05}\Delta qt^3\Delta B^3F_1^2+6.38554\times10^{-05}ql^2\Delta qt^2\Delta B^3F_1^2+7.53871\times10^{-05}ql\Delta qt^2\Delta B^3F_1^2+$$
$$1.18721\times10^{-05}\Delta qt^2\Delta B^3F_1^2+6.34067\times10^{-05}ql^2\Delta qt\Delta B^3F_1^2-0.000107734ql\Delta qt\Delta B^3F_1^2-$$
$$1.75994\times10^{-05}\Delta qt\Delta B^3F_1^2+0.000174004ql^4\Delta B^3F_1^2+0.000191173ql^3\Delta B^3F_1^2+$$
$$1.97136\times10^{-06}ql^2\Delta B^3F_1^2-0.000120736ql\Delta B^3F_1^2-0.000592701\Delta B^3F_1^2+3.53489\times$$
$$10^{-05}ql^2\Delta qt^4\Delta B^2F_1^2-3.04906\times10^{-05}ql\Delta qt^4\Delta B^2F_1^2-5.52061\times10^{-05}\Delta qt^4\Delta B^2F_1^2-$$
$$6.19934\times10^{-05}ql^2\Delta qt^3\Delta B^2F_1^2-8.69956\times10^{-05}ql\Delta qt^3\Delta B^2F_1^2-3.48558\times$$
$$10^{-05}\Delta qt^3\Delta B^2F_1^2+7.99359\times10^{-05}ql^4\Delta qt^2\Delta B^2F_1^2+2.89237\times10^{-05}ql^3\Delta qt^2\Delta B^2F_1^2-$$
$$6.90947\times10^{-05}ql^2\Delta qt^2\Delta B^2F_1^2-8.7271\times10^{-05}ql\Delta qt^2\Delta B^2F_1^2-0.000137881\Delta qt^2\Delta B^2F_1^2-$$
$$1.47688\times10^{-05}ql^4\Delta qt\Delta B^2F_1^2-1.42527\times10^{-05}ql^3\Delta qt\Delta B^2F_1^2+5.14902\times10^{-05}ql^2\Delta qt\Delta B^2F_1^2+$$
$$0.000173414ql\Delta qt\Delta B^2F_1^2+3.44095\times10^{-05}\Delta qt\Delta B^2F_1^2+0.000100039ql^4\Delta B^2F_1^2+$$
$$6.93886\times10^{-05}ql^3\Delta B^2F_1^2-0.000178138ql^2\Delta B^2F_1^2-0.000164851ql\Delta B^2F_1^2+$$
$$0.000345716\Delta B^2F_1^2-0.000127239ql^2\Delta qt^4\Delta BF_1^2-8.6136\times10^{-06}ql\Delta qt^4\Delta BF_1^2-$$
$$8.82707\times10^{-05}\Delta qt^4\Delta BF_1^2+0.00015074ql^2\Delta qt^3\Delta BF_1^2+0.00014701ql\Delta qt^3\Delta BF_1^2+$$
$$0.000131609\Delta qt^3\Delta BF_1^2-0.00017745ql^4\Delta qt^2\Delta BF_1^2-8.61673\times10^{-05}ql^3\Delta qt^2\Delta BF_1^2+$$
$$5.47662\times10^{-05}ql^2\Delta qt^2\Delta BF_1^2+0.000115976ql\Delta qt^2\Delta BF_1^2+0.000302735\Delta qt^2\Delta BF_1^2+$$
$$4.94757\times10^{-05}ql^4\Delta qt\Delta BF_1^2+9.23497\times10^{-05}ql^3\Delta qt\Delta BF_1^2-0.000200678ql^2\Delta qt\Delta BF_1^2-$$
$$0.000345312ql\Delta qt\Delta BF_1^2-6.68\times10^{-05}\Delta qt\Delta BF_1^2-0.000313987ql^4\Delta BF_1^2-0.000329731ql^3\Delta BF_1^2+$$
$$0.000390626ql^2\Delta BF_1^2+0.00048888ql\Delta BF_1^2-2.9003\times10^{-05}\Delta BF_1^2+1.39729\times$$
$$10^{-05}\Delta qt^8F_1^2+3.29753\times10^{-05}\Delta qt^7F_1^2-9.65342\times10^{-06}\Delta qt^6F_1^2+4.68611\times$$
$$10^{-05}\Delta qt^5F_1^2+2.12504\times10^{-05}ql^4\Delta qt^4F_1^2+4.47983\times10^{-05}ql^3\Delta qt^4F_1^2+6.64852\times$$
$$10^{-05}ql^2\Delta qt^4F_1^2-5.84197\times10^{-05}ql\Delta qt^4F_1^2+5.76088\times10^{-05}\Delta qt^4F_1^2-8.4735\times$$
$$10^{-06}ql^4\Delta qt^3F_1^2-4.40377\times10^{-05}ql^3\Delta qt^3F_1^2-7.67116\times10^{-05}ql^2\Delta qt^3F_1^2-$$

$3.45176\times10^{-05}ql\Delta qt^3F_1^2-0.000127441\Delta qt^3F_1^2+0.000131941ql^4\Delta qt^2F_1^2+0.000109882ql^3\Delta qt^2F_1^2-$
$2.71333\times10^{-05}ql^2\Delta qt^2F_1^2-0.000130701ql\Delta qt^2F_1^2-0.000150228\Delta qt^2F_1^2-3.62287\times$
$10^{-05}ql^4\Delta qtF_1^2-2.73987\times10^{-05}ql^3\Delta qtF_1^2+9.83754\times10^{-05}ql^2\Delta qtF_1^2+0.000132666ql\Delta qtF_1^2+$
$6.93173\times10^{-06}\Delta qtF_1^2-1.40242\times10^{-05}ql^8F_1^2-2.05407\times10^{-05}ql^7F_1^2-1.73905\times$
$10^{-05}ql^6F_1^2-7.88961\times10^{-06}ql^5F_1^2+0.000189078ql^4F_1^2+0.000168875ql^3F_1^2-$
$0.000152757ql^2F_1^2-0.000151503qlF_1^2+0.000133345F_1^2-3.5169\times10^{-05}F_0^8F_1-$
$5.27289\times10^{-05}F_0^7F_1-2.46949\times10^{-05}F_0^6F_1+1.74729\times10^{-05}F_0^5F_1-8.77761\times$
$10^{-05}\Delta B^4F_0^4F_1+0.000187464\Delta B^3F_0^4F_1+0.000109731\Delta qt^2\Delta B^2F_0^4F_1+6.15154\times$
$10^{-05}\Delta qt\Delta B^2F_0^4F_1+8.39725\times10^{-05}ql^2\Delta B^2F_0^4F_1-3.41783\times10^{-05}ql\Delta B^2F_0^4F_1+$
$0.00011191\Delta B^2F_0^4F_1-0.000218555\Delta qt^2\Delta BF_0^4F_1-0.000120978\Delta qt\Delta BF_0^4F_1-$
$0.000153123ql^2\Delta BF_0^4F_1+8.65783\times10^{-05}ql\Delta BF_0^4F_1-0.000415767\Delta BF_0^4F_1+$
$7.95461\times10^{-06}\Delta qt^4F_0^4F_1-1.74824\times10^{-05}\Delta qt^3F_0^4F_1+8.90357\times10^{-06}ql^2\Delta qt^2F_0^4F_1+$
$3.5184\times10^{-05}ql\Delta qt^2F_0^4F_1+0.000121067\Delta qt^2F_0^4F_1+7.51351\times10^{-06}ql^2\Delta qtF_0^4F_1-$
$4.00578\times10^{-07}ql\Delta qtF_0^4F_1+8.01007\times10^{-05}\Delta qtF_0^4F_1+1.84684\times10^{-06}ql^4F_0^4F_1-$
$9.69957\times10^{-06}ql^3F_0^4F_1+7.80804\times10^{-05}ql^2F_0^4F_1-1.33081\times10^{-05}qlF_0^4F_1+$
$0.0001899F_0^4F_1-0.000103078\Delta B^4F_0^3F_1+0.000179919\Delta B^3F_0^3F_1+0.000141566\Delta qt^2\Delta B^2F_0^3F_1-$
$6.15759\times10^{-05}\Delta qt\Delta B^2F_0^3F_1-0.000101417ql^2\Delta B^2F_0^3F_1-0.000265454ql\Delta B^2F_0^3F_1-$
$6.39756\times10^{-05}\Delta B^2F_0^3F_1-0.000265384\Delta qt^2\Delta BF_0^3F_1+9.32241\times10^{-05}\Delta qt\Delta BF_0^3F_1+$
$0.0001657ql^2\Delta BF_0^3F_1+0.00050131ql\Delta BF_0^3F_1-5.81225\times10^{-05}\Delta BF_0^3F_1-1.15255\times$
$10^{-05}\Delta qt^4F_0^3F_1+1.12829\times10^{-05}\Delta qt^3F_0^3F_1-7.4801\times10^{-06}ql^2\Delta qt^2F_0^3F_1-$
$2.94688\times10^{-05}ql\Delta qt^2F_0^3F_1+0.000116953\Delta qt^2F_0^3F_1+3.04566\times10^{-08}ql^2\Delta qtF_0^3F_1+$
$1.24884\times10^{-05}ql\Delta qtF_0^3F_1-3.50688\times10^{-05}\Delta qtF_0^3F_1+4.51186\times10^{-05}ql^4F_0^3F_1+$
$5.04593\times10^{-05}ql^3F_0^3F_1-0.000107782ql^2F_0^3F_1-0.000314774qlF_0^3F_1-1.27786\times$
$10^{-05}F_0^3F_1+7.16604\times10^{-05}\Delta qt^2\Delta B^4F_0^2F_1-6.46939\times10^{-06}\Delta qt\Delta B^4F_0^2F_1-$
$2.23642\times10^{-05}ql^2\Delta B^4F_0^2F_1-2.33841\times10^{-05}ql\Delta B^4F_0^2F_1+0.000166376\Delta B^4F_0^2F_1-$
$0.000119893\Delta qt^2\Delta B^3F_0^2F_1+7.05358\times10^{-05}\Delta qt\Delta B^3F_0^2F_1+5.97323\times10^{-05}ql^2\Delta B^3F_0^2F_1+$
$6.4654\times10^{-05}ql\Delta B^3F_0^2F_1-0.000275338\Delta B^3F_0^2F_1+4.99081\times10^{-05}\Delta qt^4\Delta B^2F_0^2F_1-$
$5.69968\times10^{-06}\Delta qt^3\Delta B^2F_0^2F_1+6.25813\times10^{-05}ql^2\Delta qt^2\Delta B^2F_0^2F_1+0.000152979ql\Delta qt^2\Delta B^2F_0^2F_1+$
$6.39535\times10^{-06}\Delta qt^2\Delta B^2F_0^2F_1-5.49875\times10^{-05}ql^2\Delta qt\Delta B^2F_0^2F_1-0.000184281ql\Delta qt\Delta B^2F_0^2F_1+$
$3.65435\times10^{-05}\Delta qt\Delta B^2F_0^2F_1+2.4859\times10^{-05}ql^4\Delta B^2F_0^2F_1-3.36861\times10^{-05}ql^3\Delta B^2F_0^2F_1-$
$1.31981\times10^{-05}ql^2\Delta B^2F_0^2F_1+0.000191837ql\Delta B^2F_0^2F_1+4.2835\times10^{-05}\Delta B^2F_0^2F_1-$
$5.8421\times10^{-05}\Delta qt^4\Delta BF_0^2F_1+1.95455\times10^{-05}\Delta qt^3\Delta BF_0^2F_1-0.000102321ql^2\Delta qt^2\Delta BF_0^2F_1-$
$0.000285706ql\Delta qt^2\Delta BF_0^2F_1+9.9757\times10^{-05}\Delta qt^2\Delta BF_0^2F_1+8.05187\times10^{-05}ql^2\Delta qt\Delta BF_0^2F_1+$
$0.000331718ql\Delta qt\Delta BF_0^2F_1-0.000217565\Delta qt\Delta BF_0^2F_1-2.1565\times10^{-05}ql^4\Delta BF_0^2F_1+$
$9.51804\times10^{-05}ql^3\Delta BF_0^2F_1-9.29156\times10^{-05}ql^2\Delta BF_0^2F_1-0.00056263ql\Delta BF_0^2F_1+$
$9.77337\times10^{-05}\Delta BF_0^2F_1+1.32344\times10^{-05}ql^2\Delta qt^4F_0^2F_1+5.64165\times10^{-06}ql\Delta qt^4F_0^2F_1+$

$$3.99421\times10^{-05}\Delta qt^4F_0^2F_1-1.80998\times10^{-05}ql^2\Delta qt^3F_0^2F_1-1.83329\times10^{-05}ql\Delta qt^3F_0^2F_1+$$
$$1.8486\times10^{-05}\Delta qt^3F_0^2F_1+6.27644\times10^{-06}ql^4\Delta qt^2F_0^2F_1-6.70267\times10^{-07}ql^3\Delta qt^2F_0^2F_1+$$
$$1.84026\times10^{-05}ql^2\Delta qt^2F_0^2F_1+6.73084\times10^{-05}ql\Delta qt^2F_0^2F_1-0.000108041\Delta qt^2F_0^2F_1+$$
$$4.8004\times10^{-06}ql^4\Delta qtF_0^2F_1+3.92614\times10^{-06}ql^3\Delta qtF_0^2F_1-1.75672\times10^{-05}ql^2\Delta qtF_0^2F_1-$$
$$9.68016\times10^{-05}ql\Delta qtF_0^2F_1+5.78999\times10^{-05}\Delta qtF_0^2F_1-7.71623\times10^{-06}ql^4F_0^2F_1-$$
$$2.59304\times10^{-05}ql^3F_0^2F_1+6.83555\times10^{-06}ql^2F_0^2F_1+0.00019574qlF_0^2F_1-1.89065\times$$
$$10^{-05}F_0^2F_1+6.29128\times10^{-05}\Delta qt^2\Delta B^4F_0F_1+9.11128\times10^{-05}\Delta qt\Delta B^4F_0F_1+$$
$$4.52428\times10^{-06}ql^2\Delta B^4F_0F_1-0.000101304ql\Delta B^4F_0F_1+9.12231\times10^{-05}\Delta B^4F_0F_1-$$
$$0.000121047\Delta qt^2\Delta B^3F_0F_1-0.000211872\Delta qt\Delta B^3F_0F_1+7.15882\times10^{-05}ql^2\Delta B^3F_0F_1+$$
$$0.000282272ql\Delta B^3F_0F_1-0.000120001\Delta B^3F_0F_1-2.80255\times10^{-05}\Delta qt^4\Delta B^2F_0F_1+$$
$$0.000228661\Delta qt^3\Delta B^2F_0F_1+5.45455\times10^{-05}ql^2\Delta qt^2\Delta B^2F_0F_1+0.000150902ql\Delta qt^2\Delta B^2F_0F_1+$$
$$3.6399\times10^{-05}\Delta qt^2\Delta B^2F_0F_1-6.22955\times10^{-05}ql^2\Delta qt\Delta B^2F_0F_1+5.34875\times$$
$$10^{-05}ql\Delta qt\Delta B^2F_0F_1-0.000139122\Delta qt\Delta B^2F_0F_1-6.51772\times10^{-05}ql^4\Delta B^2F_0F_1-$$
$$0.000203277ql^3\Delta B^2F_0F_1+2.8571\times10^{-05}ql^2\Delta B^2F_0F_1+0.000380572ql\Delta B^2F_0F_1+$$
$$0.000257706\Delta B^2F_0F_1+5.10627\times10^{-05}\Delta qt^4\Delta BF_0F_1-0.0003942\Delta qt^3\Delta BF_0F_1-$$
$$8.03196\times10^{-05}ql^2\Delta qt^2\Delta BF_0F_1-0.000257499ql\Delta qt^2\Delta BF_0F_1+5.99611\times10^{-05}\Delta qt^2\Delta BF_0F_1+$$
$$7.11628\times10^{-05}ql^2\Delta qt\Delta BF_0F_1-0.000122687ql\Delta qt\Delta BF_0F_1+0.000432996\Delta qt\Delta BF_0F_1+$$
$$0.000234889ql^4\Delta BF_0F_1+0.000562925ql^3\Delta BF_0F_1-9.4112\times10^{-05}ql^2\Delta BF_0F_1-$$
$$0.00122475ql\Delta BF_0F_1-0.000508352\Delta BF_0F_1+1.32169\times10^{-05}ql^2\Delta qt^4F_0F_1+$$
$$3.8545\times10^{-05}ql\Delta qt^4F_0F_1+5.866\times10^{-06}\Delta qt^4F_0F_1+8.19191\times10^{-06}ql^2\Delta qt^3F_0F_1-$$
$$3.10819\times10^{-05}ql\Delta qt^3F_0F_1+0.000243708\Delta qt^3F_0F_1+4.2533\times10^{-05}ql^4\Delta qt^2F_0F_1+$$
$$5.6448\times10^{-05}ql^3\Delta qt^2F_0F_1+4.80401\times10^{-05}ql^2\Delta qt^2F_0F_1+6.9606\times10^{-05}ql\Delta qt^2F_0F_1-$$
$$7.34389\times10^{-05}\Delta qt^2F_0F_1-1.72457\times10^{-05}ql^4\Delta qtF_0F_1-7.59203\times10^{-06}ql^3\Delta qtF_0F_1-$$
$$8.0112\times10^{-05}ql^2\Delta qtF_0F_1+6.85074\times10^{-05}ql\Delta qtF_0F_1-0.000287157\Delta qtF_0F_1-$$
$$0.000208256ql^4F_0F_1-0.000348886ql^3F_0F_1-6.9362\times10^{-06}ql^2F_0F_1+0.00058656qlF_0F_1+$$
$$0.000340666F_0F_1-6.12831\times10^{-05}\Delta B^8F_1+6.78023\times10^{-05}\Delta B^7F_1-6.20806\times$$
$$10^{-05}\Delta B^6F_1+7.60566\times10^{-05}\Delta B^5F_1-8.88478\times10^{-05}\Delta qt^4\Delta B^4F_1+7.78373\times$$
$$10^{-05}\Delta qt^3\Delta B^4F_1-9.97485\times10^{-06}ql^2\Delta qt^2\Delta B^4F_1-8.20026\times10^{-05}ql\Delta qt^2\Delta B^4F_1+$$
$$0.00018576\Delta qt^2\Delta B^4F_1+0.00011095ql^2\Delta qt\Delta B^4F_1+0.000285602ql\Delta qt\Delta B^4F_1-$$
$$0.00019247\Delta qt\Delta B^4F_1-0.000170502ql^4\Delta B^4F_1-0.000232357ql^3\Delta B^4F_1+6.95224\times$$
$$10^{-06}ql^2\Delta B^4F_1+1.89197\times10^{-05}ql\Delta B^4F_1+0.000484534\Delta B^4F_1+0.000133415\Delta qt^4\Delta B^3F_1-$$
$$0.000202266\Delta qt^3\Delta B^3F_1+5.49969\times10^{-05}ql^2\Delta qt^2\Delta B^3F_1+0.000217682ql\Delta qt^2\Delta B^3F_1-$$
$$0.000275093\Delta qt^2\Delta B^3F_1-0.000150893ql^2\Delta qt\Delta B^3F_1-0.000514514ql\Delta qt\Delta B^3F_1+$$
$$0.000497184\Delta qt\Delta B^3F_1+0.000217502ql^4\Delta B^3F_1+0.000343407ql^3\Delta B^3F_1+8.14537\times$$
$$10^{-05}ql^2\Delta B^3F_1+3.08574\times10^{-05}ql\Delta B^3F_1-0.000918119\Delta B^3F_1+9.98512\times$$
$$10^{-05}ql^2\Delta qt^4\Delta B^2F_1+8.1919\times10^{-05}ql\Delta qt^4\Delta B^2F_1+6.14909\times10^{-05}\Delta qt^4\Delta B^2F_1-$$

$$2.60232\times10^{-05}ql^2\Delta qt^3\Delta B^2F_1+0.000140965ql\Delta qt^3\Delta B^2F_1-0.000107517\Delta qt^3\Delta B^2F_1+$$
$$4.89624\times10^{-05}ql^4\Delta qt^2\Delta B^2F_1-7.40854\times10^{-06}ql^3\Delta qt^2\Delta B^2F_1-5.22169\times$$
$$10^{-05}ql^2\Delta qt^2\Delta B^2F_1+8.0263\times10^{-05}ql\Delta qt^2\Delta B^2F_1-0.000105011\Delta qt^2\Delta B^2F_1-$$
$$7.92278\times10^{-05}ql^4\Delta qt\Delta B^2F_1+1.68685\times10^{-06}ql^3\Delta qt\Delta B^2F_1+0.000200897ql^2\Delta qt\Delta B^2F_1-$$
$$0.000173528ql\Delta qt\Delta B^2F_1+6.51445\times10^{-05}\Delta qt\Delta B^2F_1+4.7205\times10^{-05}ql^4\Delta B^2F_1-$$
$$0.000177542ql^3\Delta B^2F_1-0.000391593ql^2\Delta B^2F_1+0.000273656ql\Delta B^2F_1+0.000355472\Delta B^2F_1-$$
$$0.000182827ql^2\Delta qt^4\Delta BF_1-0.000185947ql\Delta qt^4\Delta BF_1-0.000260731\Delta qt^4\Delta BF_1+$$
$$3.23168\times10^{-05}ql^2\Delta qt^3\Delta BF_1-0.000295294ql\Delta qt^3\Delta BF_1+0.000474732\Delta qt^3\Delta BF_1-$$
$$0.000119541ql^4\Delta qt^2\Delta BF_1-5.19877\times10^{-05}ql^3\Delta qt^2\Delta BF_1-3.29937\times10^{-05}ql^2\Delta qt^2\Delta BF_1-$$
$$0.00045127ql\Delta qt^2\Delta BF_1+0.000443635\Delta qt^2\Delta BF_1+0.000100259ql^4\Delta qt\Delta BF_1-$$
$$9.40643\times10^{-05}ql^3\Delta qt\Delta BF_1-0.000235662ql^2\Delta qt\Delta BF_1+0.00104734ql\Delta qt\Delta BF_1-$$
$$0.000600961\Delta qt\Delta BF_1-0.000203548ql^4\Delta BF_1+0.000190523ql^3\Delta BF_1+0.000657979ql^2\Delta BF_1-$$
$$0.000875447ql\Delta BF_1+6.74919\times10^{-05}\Delta BF_1+1.90256\times10^{-05}\Delta qt^8F_1-1.26127\times$$
$$10^{-05}\Delta qt^7F_1-1.07886\times10^{-05}\Delta qt^6F_1-6.99447\times10^{-05}\Delta qt^5F_1+2.11172\times$$
$$10^{-05}ql^4\Delta qt^4F_1+2.05284\times10^{-05}ql^3\Delta qt^4F_1+8.52002\times10^{-05}ql^2\Delta qt^4F_1+$$
$$3.52399\times10^{-05}ql\Delta qt^4F_1+0.000107394\Delta qt^4F_1-8.63095\times10^{-06}ql^4\Delta qt^3F_1-$$
$$3.61284\times10^{-06}ql^3\Delta qt^3F_1-1.83852\times10^{-05}ql^2\Delta qt^3F_1+0.000123229ql\Delta qt^3F_1-$$
$$9.38645\times10^{-05}\Delta qt^3F_1+0.000104896ql^4\Delta qt^2F_1+9.05198\times10^{-05}ql^3\Delta qt^2F_1-$$
$$1.82671\times10^{-05}ql^2\Delta qt^2F_1+8.1185\times10^{-05}ql\Delta qt^2F_1-0.000320748\Delta qt^2F_1-$$
$$5.159\times10^{-05}ql^4\Delta qtF_1+4.7694\times10^{-05}ql^3\Delta qtF_1+8.01902\times10^{-05}ql^2\Delta qtF_1-$$
$$0.000510545ql\Delta qtF_1+0.000206193\Delta qtF_1-3.25219\times10^{-05}ql^8F_1-7.59334\times$$
$$10^{-05}ql^7F_1-5.64898\times10^{-05}ql^6F_1-6.30394\times10^{-05}ql^5F_1+0.000135085ql^4F_1+$$
$$2.00873\times10^{-05}ql^3F_1-0.000269391ql^2F_1+0.000656058qlF_1+0.000239835F_1+$$
$$0.000119169\Delta B^2F_0^8-0.000262584\Delta BF_0^8+1.79305\times10^{-05}\Delta qt^2F_0^8-3.61173\times$$
$$10^{-05}\Delta qtF_0^8+9.07319\times10^{-06}ql^2F_0^8+1.85164\times10^{-05}qlF_0^8+0.000141945F_0^8-$$
$$0.000151795\Delta B^2F_0^7+0.000223059\Delta BF_0^7-1.90999\times10^{-05}\Delta qt^2F_0^7-5.05524\times$$
$$10^{-05}\Delta qtF_0^7+1.5029\times10^{-05}ql^2F_0^7+5.62673\times10^{-05}qlF_0^7-0.00017814F_0^7-$$
$$9.48909\times10^{-05}\Delta B^2F_0^6+0.000137885\Delta BF_0^6-2.52939\times10^{-05}\Delta qt^2F_0^6+2.31102\times$$
$$10^{-05}\Delta qtF_0^6+1.06967\times10^{-05}ql^2F_0^6+2.20361\times10^{-05}qlF_0^6-0.00011659F_0^6+$$
$$4.35978\times10^{-05}\Delta B^2F_0^5-5.23656\times10^{-05}\Delta BF_0^5+7.6068\times10^{-06}\Delta qt^2F_0^5+$$
$$8.13252\times10^{-06}\Delta qtF_0^5+1.41353\times10^{-05}ql^2F_0^5-2.88743\times10^{-05}qlF_0^5+6.02364\times$$
$$10^{-05}F_0^5-2.20769\times10^{-05}\Delta qt^2\Delta B^4F_0^4-4.416\times10^{-05}\Delta qt\Delta B^4F_0^4-2.57715\times$$
$$10^{-05}ql^2\Delta B^4F_0^4+1.70679\times10^{-05}ql\Delta B^4F_0^4-7.51311\times10^{-05}\Delta B^4F_0^4+6.67336\times$$
$$10^{-05}\Delta qt^2\Delta B^3F_0^4+8.51284\times10^{-05}\Delta qt\Delta B^3F_0^4+5.46832\times10^{-05}ql^2\Delta B^3F_0^4-$$
$$4.89266\times10^{-05}ql\Delta B^3F_0^4+0.000127402\Delta B^3F_0^4+6.05994\times10^{-05}\Delta qt^4\Delta B^2F_0^4-$$
$$2.26646\times10^{-05}\Delta qt^3\Delta B^2F_0^4+5.02645\times10^{-05}ql^2\Delta qt^2\Delta B^2F_0^4-6.20061\times10^{-05}ql\Delta qt^2\Delta B^2F_0^4-$$

$3.02312\times10^{-05}\Delta qt^2\Delta B^2F_0^4+3.40182\times10^{-05}ql^2\Delta qt\Delta B^2F_0^4+7.93059\times10^{-06}ql\Delta qt\Delta B^2F_0^4+$
$0.000132779\Delta qt\Delta B^2F_0^4+0.00013736ql^4\Delta B^2F_0^4+0.000159041ql^3\Delta B^2F_0^4+4.21916\times$
$10^{-05}ql^2\Delta B^2F_0^4-0.000116705ql\Delta B^2F_0^4-0.000165257\Delta B^2F_0^4-0.000125302\Delta qt^4\Delta BF_0^4+$
$6.93028\times10^{-05}\Delta qt^3\Delta BF_0^4-8.24249\times10^{-05}ql^2\Delta qt^2\Delta BF_0^4+0.000128586ql\Delta qt^2\Delta BF_0^4-$
$1.94562\times10^{-05}\Delta qt^2\Delta BF_0^4-4.34321\times10^{-05}ql^2\Delta qt\Delta BF_0^4+3.68372\times10^{-05}ql\Delta qt\Delta BF_0^4-$
$0.000352433\Delta qt\Delta BF_0^4-0.000301997ql^4\Delta BF_0^4-0.000295407ql^3\Delta BF_0^4-0.000163279ql^2\Delta BF_0^4+$
$0.000252983ql\Delta BF_0^4+0.000188551\Delta BF_0^4-1.83432\times10^{-05}ql^2\Delta qt^4F_0^4-9.1372\times$
$10^{-06}ql\Delta qt^4F_0^4+4.54804\times10^{-05}\Delta qt^4F_0^4-2.0141\times10^{-06}ql^2\Delta qt^3F_0^4-1.57855\times$
$10^{-05}ql\Delta qt^3F_0^4-3.78942\times10^{-05}\Delta qt^3F_0^4-2.47408\times10^{-05}ql^4\Delta qt^2F_0^4-3.69296\times$
$10^{-05}ql^3\Delta qt^2F_0^4+4.03359\times10^{-05}ql^2\Delta qt^2F_0^4-3.1069\times10^{-06}ql\Delta qt^2F_0^4+1.53886\times$
$10^{-05}\Delta qt^2F_0^4+4.27444\times10^{-05}ql^4\Delta qtF_0^4+6.47891\times10^{-05}ql^3\Delta qtF_0^4+2.34663\times$
$10^{-05}ql^2\Delta qtF_0^4-5.59834\times10^{-05}ql\Delta qtF_0^4+0.000124126\Delta qtF_0^4+0.000100894ql^4F_0^4+$
$0.000119758ql^3F_0^4+8.08496\times10^{-05}ql^2F_0^4-6.69642\times10^{-05}qlF_0^4-4.60173\times$
$10^{-05}F_0^4+2.94347\times10^{-05}\Delta qt^2\Delta B^4F_0^3+2.68226\times10^{-06}\Delta qt\Delta B^4F_0^3-3.52724\times$
$10^{-05}ql^2\Delta B^4F_0^3-0.000167714ql\Delta B^4F_0^3-7.92834\times10^{-05}\Delta B^4F_0^3+8.78956\times$
$10^{-06}\Delta qt^2\Delta B^3F_0^3-9.37886\times10^{-06}\Delta qt\Delta B^3F_0^3+0.000119855ql^2\Delta B^3F_0^3+0.000350024ql\Delta B^3F_0^3+$
$0.000247825\Delta B^3F_0^3-4.65732\times10^{-05}\Delta qt^4\Delta B^2F_0^3+8.26594\times10^{-05}\Delta qt^3\Delta B^2F_0^3+$
$0.000165907ql^2\Delta qt^2\Delta B^2F_0^3+0.000182231ql\Delta qt^2\Delta B^2F_0^3+0.000144379\Delta qt^2\Delta B^2F_0^3-$
$0.000100458ql^2\Delta qt\Delta B^2F_0^3-0.000141895ql\Delta qt\Delta B^2F_0^3-0.000123725\Delta qt\Delta B^2F_0^3-$
$2.62288\times10^{-05}ql^4\Delta B^2F_0^3-0.00011569ql^3\Delta B^2F_0^3+5.37664\times10^{-05}ql^2\Delta B^2F_0^3+$
$8.69783\times10^{-05}ql\Delta B^2F_0^3-5.09485\times10^{-05}\Delta B^2F_0^3+0.000120645\Delta qt^4\Delta BF_0^3-$
$0.000201975\Delta qt^3\Delta BF_0^3-0.000332619ql^2\Delta qt^2\Delta BF_0^3-0.000370218ql\Delta qt^2\Delta BF_0^3-$
$0.000267529\Delta qt^2\Delta BF_0^3+0.000282526ql^2\Delta qt\Delta BF_0^3+0.000318754ql\Delta qt\Delta BF_0^3+$
$0.000308192\Delta qt\Delta BF_0^3+6.94749\times10^{-05}ql^4\Delta BF_0^3+0.000181788ql^3\Delta BF_0^3-$
$0.0002102ql^2\Delta BF_0^3-0.000441079ql\Delta BF_0^3-0.000206238\Delta BF_0^3+6.81199\times10^{-06}ql^2\Delta qt^4F_0^3+$
$1.45186\times10^{-05}ql\Delta qt^4F_0^3-3.08861\times10^{-05}\Delta qt^4F_0^3-4.19184\times10^{-05}ql^2\Delta qt^3F_0^3-$
$3.15297\times10^{-05}ql\Delta qt^3F_0^3+4.11571\times10^{-05}\Delta qt^3F_0^3-5.62808\times10^{-06}ql^4\Delta qt^2F_0^3+$
$4.43534\times10^{-07}ql^3\Delta qt^2F_0^3+0.000187433ql^2\Delta qt^2F_0^3+0.000206439ql\Delta qt^2F_0^3+$
$9.35848\times10^{-05}\Delta qt^2F_0^3+1.21328\times10^{-05}ql^4\Delta qtF_0^3-1.39585\times10^{-05}ql^3\Delta qtF_0^3-$
$8.07845\times10^{-05}ql^2\Delta qtF_0^3-8.7814\times10^{-05}ql\Delta qtF_0^3-9.6418\times10^{-05}\Delta qtF_0^3-$
$4.45516\times10^{-06}ql^4F_0^3-0.000136432ql^3F_0^3+6.50828\times10^{-05}ql^2F_0^3+0.000295827qlF_0^3-$
$2.61501\times10^{-05}F_0^3-0.000136602\Delta B^8F_0^2+0.000231105\Delta B^7F_0^2-0.000184856\Delta B^6F_0^2+$
$0.000111873\Delta B^5F_0^2-6.04087\times10^{-05}\Delta qt^4\Delta B^4F_0^2+8.83875\times10^{-05}\Delta qt^3\Delta B^4F_0^2-$
$6.75385\times10^{-05}ql^2\Delta qt^2\Delta B^4F_0^2-0.000112207ql\Delta qt^2\Delta B^4F_0^2-3.13722\times10^{-05}\Delta qt^2\Delta B^4F_0^2+$
$1.1286\times10^{-05}ql^2\Delta qt\Delta B^4F_0^2+7.1965\times10^{-05}ql\Delta qt\Delta B^4F_0^2-8.12905\times10^{-05}\Delta qt\Delta B^4F_0^2-$
$9.98402\times10^{-05}ql^4\Delta B^4F_0^2-8.91989\times10^{-05}ql^3\Delta B^4F_0^2+9.77875\times10^{-07}ql^2\Delta B^4F_0^2+$

$4.59138\times10^{-05}ql\Delta B^4F_0^2+0.000197928\Delta B^4F_0^2+8.58672\times10^{-05}\Delta qt^4\Delta B^3F_0^2-$
$0.000118198\Delta qt^3\Delta B^3F_0^2+0.000136444ql^2\Delta qt^2\Delta B^3F_0^2+0.000206315ql\Delta qt^2\Delta B^3F_0^2+$
$4.74605\times10^{-05}\Delta qt^2\Delta B^3F_0^2-4.47845\times10^{-05}ql^2\Delta qt\Delta B^3F_0^2-0.000150556ql\Delta qt\Delta B^3F_0^2+$
$8.22004\times10^{-05}\Delta qt\Delta B^3F_0^2+0.000166741ql^4\Delta B^3F_0^2+0.000187118ql^3\Delta B^3F_0^2+$
$2.49797\times10^{-05}ql^2\Delta B^3F_0^2-7.9011\times10^{-05}ql\Delta B^3F_0^2-0.000446223\Delta B^3F_0^2-$
$2.84996\times10^{-05}ql^2\Delta qt^4\Delta B^2F_0^2-9.78098\times10^{-05}ql\Delta qt^4\Delta B^2F_0^2-6.72524\times$
$10^{-05}\Delta qt^4\Delta B^2F_0^2-1.79515\times10^{-05}ql^2\Delta qt^3\Delta B^2F_0^2+3.41684\times10^{-05}ql\Delta qt^3\Delta B^2F_0^2+$
$4.13326\times10^{-05}\Delta qt^3\Delta B^2F_0^2+9.00317\times10^{-05}ql^4\Delta qt^2\Delta B^2F_0^2+7.68062\times10^{-06}ql^3\Delta qt^2\Delta B^2F_0^2-$
$0.000108253ql^2\Delta qt^2\Delta B^2F_0^2-0.000137299ql\Delta qt^2\Delta B^2F_0^2-0.00020447\Delta qt^2\Delta B^2F_0^2+$
$2.60895\times10^{-05}ql^4\Delta qt\Delta B^2F_0^2+6.84739\times10^{-05}ql^3\Delta qt\Delta B^2F_0^2+5.90438\times10^{-05}ql^2\Delta qt\Delta B^2F_0^2-$
$6.24527\times10^{-06}ql\Delta qt\Delta B^2F_0^2+5.34906\times10^{-05}\Delta qt\Delta B^2F_0^2+7.06788\times10^{-05}ql^4\Delta B^2F_0^2+$
$6.7496\times10^{-05}ql^3\Delta B^2F_0^2-0.000127236ql^2\Delta B^2F_0^2-0.000194475ql\Delta B^2F_0^2+$
$0.000121782\Delta B^2F_0^2-9.18273\times10^{-07}ql^2\Delta qt^4\Delta BF_0^2+0.000113887ql\Delta qt^4\Delta BF_0^2+$
$2.29519\times10^{-05}\Delta qt^4\Delta BF_0^2+5.65224\times10^{-05}ql^2\Delta qt^3\Delta BF_0^2-6.81336\times10^{-05}ql\Delta qt^3\Delta BF_0^2+$
$4.38261\times10^{-05}\Delta qt^3\Delta BF_0^2-0.000141108ql^4\Delta qt^2\Delta BF_0^2-4.71436\times10^{-05}ql^3\Delta qt^2\Delta BF_0^2+$
$4.69794\times10^{-05}ql^2\Delta qt^2\Delta BF_0^2+8.90002\times10^{-05}ql\Delta qt^2\Delta BF_0^2+0.000311354\Delta qt^2\Delta BF_0^2-$
$1.86496\times10^{-05}ql^4\Delta qt\Delta BF_0^2-6.3828\times10^{-05}ql^3\Delta qt\Delta BF_0^2-0.00010836ql^2\Delta qt\Delta BF_0^2+$
$4.72743\times10^{-05}ql\Delta qt\Delta BF_0^2-0.000261422\Delta qt\Delta BF_0^2-0.000307958ql^4\Delta BF_0^2-$
$0.000297386ql^3\Delta BF_0^2+0.000225555ql^2\Delta BF_0^2+0.000505307ql\Delta BF_0^2+0.000279403\Delta BF_0^2-$
$2.98766\times10^{-07}\Delta qt^8F_0^2+4.5324\times10^{-05}\Delta qt^7F_0^2-9.12435\times10^{-06}\Delta qt^6F_0^2+$
$3.17887\times10^{-05}\Delta qt^5F_0^2+3.84826\times10^{-05}ql^4\Delta qt^4F_0^2+6.97279\times10^{-05}ql^3\Delta qt^4F_0^2-$
$1.95818\times10^{-06}ql^2\Delta qt^4F_0^2-0.000134674ql\Delta qt^4F_0^2-3.41783\times10^{-05}\Delta qt^4F_0^2-$
$3.76093\times10^{-05}ql^4\Delta qt^3F_0^2-7.41479\times10^{-05}ql^3\Delta qt^3F_0^2-3.66577\times10^{-05}ql^2\Delta qt^3F_0^2+$
$9.06111\times10^{-05}ql\Delta qt^3F_0^2-4.19557\times10^{-05}\Delta qt^3F_0^2+0.00010684ql^4\Delta qt^2F_0^2+$
$4.70812\times10^{-05}ql^3\Delta qt^2F_0^2-4.82647\times10^{-05}ql^2\Delta qt^2F_0^2-8.89725\times10^{-05}ql\Delta qt^2F_0^2-$
$0.000173702\Delta qt^2F_0^2+2.60711\times10^{-05}ql^4\Delta qtF_0^2+8.83879\times10^{-05}ql^3\Delta qtF_0^2+$
$8.70428\times10^{-05}ql^2\Delta qtF_0^2-3.28308\times10^{-05}ql\Delta qtF_0^2+9.70745\times10^{-05}\Delta qtF_0^2-$
$2.33951\times10^{-05}ql^8F_0^2-4.01894\times10^{-05}ql^7F_0^2-3.48326\times10^{-05}ql^6F_0^2+3.31066\times$
$10^{-06}ql^5F_0^2+0.000129397ql^4F_0^2+0.000106376ql^3F_0^2-0.000104108ql^2F_0^2-0.000225534qlF_0^2-$
$1.89594\times10^{-07}F_0^2-5.5175\times10^{-05}\Delta B^8F_0+0.000138162\Delta B^7F_0-0.000123818\Delta B^6F_0+$
$9.56027\times10^{-05}\Delta B^5F_0-3.12355\times10^{-05}\Delta qt^4\Delta B^4F_0+6.37526\times10^{-05}\Delta qt^3\Delta B^4F_0-$
$7.03608\times10^{-05}ql^2\Delta qt^2\Delta B^4F_0-4.96057\times10^{-05}ql\Delta qt^2\Delta B^4F_0-4.86572\times$
$10^{-05}\Delta qt^2\Delta B^4F_0+6.73559\times10^{-05}ql^2\Delta qt\Delta B^4F_0+9.0776\times10^{-05}ql\Delta qt\Delta B^4F_0+$
$3.16854\times10^{-05}\Delta qt\Delta B^4F_0-1.65908\times10^{-05}ql^4\Delta B^4F_0+2.96691\times10^{-05}ql^3\Delta B^4F_0-$
$1.61566\times10^{-05}ql^2\Delta B^4F_0+4.18396\times10^{-05}ql\Delta B^4F_0+5.10879\times10^{-05}\Delta B^4F_0+$
$5.51278\times10^{-05}\Delta qt^4\Delta B^3F_0-7.74477\times10^{-05}\Delta qt^3\Delta B^3F_0+8.87474\times10^{-05}ql^2\Delta qt^2\Delta B^3F_0+$

$$4.95319\times10^{-05}ql\Delta qt^2\Delta B^3F_0-2.11101\times10^{-05}\Delta qt^2\Delta B^3F_0-0.000154486ql^2\Delta qt\Delta B^3F_0-$$
$$0.000203065ql\Delta qt\Delta B^3F_0-0.000119216\Delta qt\Delta B^3F_0+7.80897\times10^{-05}ql^4\Delta B^3F_0-$$
$$7.66223\times10^{-05}ql^3\Delta B^3F_0+8.85149\times10^{-06}ql^2\Delta B^3F_0-0.000128694ql\Delta B^3F_0-$$
$$0.000460675\Delta B^3F_0-3.0976\times10^{-05}ql^2\Delta qt^4\Delta B^2F_0-6.22394\times10^{-05}ql\Delta qt^4\Delta B^2F_0-$$
$$4.92292\times10^{-05}\Delta qt^4\Delta B^2F_0+1.98759\times10^{-05}ql^2\Delta qt^3\Delta B^2F_0+4.41849\times10^{-05}ql\Delta qt^3\Delta B^2F_0+$$
$$2.81912\times10^{-05}\Delta qt^3\Delta B^2F_0+8.54343\times10^{-06}ql^4\Delta qt^2\Delta B^2F_0-1.85364\times10^{-05}ql^3\Delta qt^2\Delta B^2F_0-$$
$$0.000151006ql^2\Delta qt^2\Delta B^2F_0-8.31121\times10^{-05}ql\Delta qt^2\Delta B^2F_0-2.60112\times10^{-07}\Delta qt^2\Delta B^2F_0-$$
$$3.88125\times10^{-05}ql^4\Delta qt\Delta B^2F_0-9.33724\times10^{-06}ql^3\Delta qt\Delta B^2F_0+9.66478\times10^{-05}ql^2\Delta qt\Delta B^2F_0+$$
$$0.000116496ql\Delta qt\Delta B^2F_0+9.86198\times10^{-06}\Delta qt\Delta B^2F_0+7.8624\times10^{-05}ql^4\Delta B^2F_0+$$
$$0.000192928ql^3\Delta B^2F_0-0.000139473ql^2\Delta B^2F_0-5.76823\times10^{-05}ql\Delta B^2F_0+$$
$$0.000736064\Delta B^2F_0+4.2099\times10^{-05}ql^2\Delta qt^4\Delta BF_0+0.000107927ql\Delta qt^4\Delta BF_0+$$
$$1.62688\times10^{-05}\Delta qt^4\Delta BF_0-3.78417\times10^{-05}ql^2\Delta qt^3\Delta BF_0-8.75487\times10^{-05}ql\Delta qt^3\Delta BF_0+$$
$$8.51877\times10^{-05}\Delta qt^3\Delta BF_0+5.29495\times10^{-06}ql^4\Delta qt^2\Delta BF_0+0.000159818ql^3\Delta qt^2\Delta BF_0+$$
$$0.000203765ql^2\Delta qt^2\Delta BF_0+8.14891\times10^{-06}ql\Delta qt^2\Delta BF_0+6.67153\times10^{-06}\Delta qt^2\Delta BF_0+$$
$$6.75042\times10^{-05}ql^4\Delta qt\Delta BF_0+3.97739\times10^{-05}ql^3\Delta qt\Delta BF_0-5.08859\times10^{-05}ql^2\Delta qt\Delta BF_0+$$
$$3.88591\times10^{-05}ql\Delta qt\Delta BF_0+5.13987\times10^{-05}\Delta qt\Delta BF_0-0.000156305ql^4\Delta BF_0-$$
$$0.000165762ql^3\Delta BF_0+0.000225102ql^2\Delta BF_0-1.16594\times10^{-05}ql\Delta BF_0-0.00109057\Delta BF_0+$$
$$2.53347\times10^{-05}\Delta qt^8F_0-1.46228\times10^{-05}\Delta qt^7F_0-4.18202\times10^{-05}\Delta qt^6F_0+$$
$$2.97133\times10^{-05}\Delta qt^5F_0+1.77008\times10^{-05}ql^4\Delta qt^4F_0+3.12619\times10^{-05}ql^3\Delta qt^4F_0-$$
$$4.81047\times10^{-05}ql^2\Delta qt^4F_0-0.000132107ql\Delta qt^4F_0-7.06084\times10^{-05}\Delta qt^4F_0+$$
$$3.96837\times10^{-06}ql^4\Delta qt^3F_0-2.06081\times10^{-06}ql^3\Delta qt^3F_0+7.17097\times10^{-05}ql^2\Delta qt^3F_0+$$
$$0.00010929ql\Delta qt^3F_0-2.84074\times10^{-06}\Delta qt^3F_0+1.8128\times10^{-05}ql^4\Delta qt^2F_0+$$
$$3.72127\times10^{-07}ql^3\Delta qt^2F_0-8.85339\times10^{-05}ql^2\Delta qt^2F_0-8.59463\times10^{-05}ql\Delta qt^2F_0+$$
$$9.37759\times10^{-05}\Delta qt^2F_0-5.61292\times10^{-05}ql^4\Delta qtF_0-4.985\times10^{-06}ql^3\Delta qtF_0-$$
$$9.89047\times10^{-05}ql^2\Delta qtF_0-0.000199959ql\Delta qtF_0-5.96116\times10^{-05}\Delta qtF_0-4.45111\times$$
$$10^{-05}ql^8F_0-8.56219\times10^{-05}ql^7F_0-3.96396\times10^{-05}ql^6F_0-2.251\times10^{-05}ql^5F_0-$$
$$2.63755\times10^{-05}ql^4F_0+0.000133088ql^3F_0-8.8354\times10^{-05}ql^2F_0+0.000162668qlF_0+$$
$$0.00108563F_0-0.000187655\Delta qt^2\Delta B^8+4.42756\times10^{-06}\Delta qt\Delta B^8-0.00020984ql^2\Delta B^8-$$
$$0.000124061ql\Delta B^8-0.000377489\Delta B^8+0.000324273\Delta qt^2\Delta B^7-2.82965\times10^{-05}\Delta qt\Delta B^7+$$
$$0.000374764ql^2\Delta B^7+0.000231527ql\Delta B^7+0.000630581\Delta B^7-0.000304767\Delta qt^2\Delta B^6+$$
$$5.28674\times10^{-05}\Delta qt\Delta B^6-0.000316534ql^2\Delta B^6-0.000123827ql\Delta B^6-0.000502354\Delta B^6+$$
$$0.000158206\Delta qt^2\Delta B^5+6.92472\times10^{-05}\Delta qt\Delta B^5+0.000211552ql^2\Delta B^5+2.90532\times$$
$$10^{-06}ql\Delta B^5+0.000237562\Delta B^5-8.73023\times10^{-05}ql^2\Delta qt^4\Delta B^4-2.9854\times10^{-05}ql\Delta qt^4\Delta B^4-$$
$$0.00013564\Delta qt^4\Delta B^4+5.97473\times10^{-05}ql^2\Delta qt^3\Delta B^4-1.30861\times10^{-05}ql\Delta qt^3\Delta B^4+$$
$$0.000108405\Delta qt^3\Delta B^4-7.22708\times10^{-05}ql^4\Delta qt^2\Delta B^4-6.30152\times10^{-05}ql^3\Delta qt^2\Delta B^4+$$
$$5.19995\times10^{-05}ql^2\Delta qt^2\Delta B^4+0.000160396ql\Delta qt^2\Delta B^4+0.000248568\Delta qt^2\Delta B^4-$$

$$4.94362\times10^{-05}ql^4\Delta qt\Delta B^4-3.37283\times10^{-05}ql^3\Delta qt\Delta B^4-9.03365\times10^{-05}ql^2\Delta qt\Delta B^4-$$
$$9.5699\times10^{-05}ql\Delta qt\Delta B^4-0.000244158\Delta qt\Delta B^4-0.000160643ql^4\Delta B^4-0.000179985ql^3\Delta B^4+$$
$$0.000287085ql^2\Delta B^4+0.000686505ql\Delta B^4+0.000890953\Delta B^4+0.000108838ql^2\Delta qt^4\Delta B^3-$$
$$1.40072\times10^{-05}ql\Delta qt^4\Delta B^3+0.000159098\Delta qt^4\Delta B^3-7.62879\times10^{-05}ql^2\Delta qt^3\Delta B^3+$$
$$3.65177\times10^{-05}ql\Delta qt^3\Delta B^3-0.000132664\Delta qt^3\Delta B^3+0.000122426ql^4\Delta qt^2\Delta B^3+$$
$$4.57389\times10^{-05}ql^3\Delta qt^2\Delta B^3-0.000104725ql^2\Delta qt^2\Delta B^3-0.000182987ql\Delta qt^2\Delta B^3-$$
$$0.000626181\Delta qt^2\Delta B^3+0.000151726ql^4\Delta qt\Delta B^3+0.00014236ql^3\Delta qt\Delta B^3+0.000147486ql^2\Delta qt\Delta B^3+$$
$$0.000225513ql\Delta qt\Delta B^3+0.000386996\Delta qt\Delta B^3+0.00022752ql^4\Delta B^3+0.000303362ql^3\Delta B^3-$$
$$0.00071398ql^2\Delta B^3-0.00143746ql\Delta B^3-0.00206996\Delta B^3+0.000155001\Delta qt^8\Delta B^2-$$
$$7.67737\times10^{-06}\Delta qt^7\Delta B^2-0.000164042\Delta qt^6\Delta B^2-8.17153\times10^{-05}\Delta qt^5\Delta B^2+$$
$$8.36758\times10^{-05}ql^4\Delta qt^4\Delta B^2+4.89937\times10^{-05}ql^3\Delta qt^4\Delta B^2-1.3771\times10^{-05}ql^2\Delta qt^4\Delta B^2-$$
$$4.7366\times10^{-05}ql\Delta qt^4\Delta B^2-0.000127234\Delta qt^4\Delta B^2-0.000116839ql^4\Delta qt^3\Delta B^2-$$
$$0.000162947ql^3\Delta qt^3\Delta B^2-3.68123\times10^{-05}ql^2\Delta qt^3\Delta B^2+7.24203\times10^{-05}ql\Delta qt^3\Delta B^2+$$
$$7.51081\times10^{-05}\Delta qt^3\Delta B^2+6.84232\times10^{-05}ql^4\Delta qt^2\Delta B^2+7.64265\times10^{-05}ql^3\Delta qt^2\Delta B^2-$$
$$0.000194795ql^2\Delta qt^2\Delta B^2-0.000217048ql\Delta qt^2\Delta B^2+0.000264964\Delta qt^2\Delta B^2-$$
$$3.05621\times10^{-05}ql^4\Delta qt\Delta B^2+5.03643\times10^{-05}ql^3\Delta qt\Delta B^2+9.03406\times10^{-05}ql^2\Delta qt\Delta B^2-$$
$$0.000520433ql\Delta qt\Delta B^2-0.000542198\Delta qt\Delta B^2+0.000130616ql^8\Delta B^2+2.54425\times$$
$$10^{-05}ql^7\Delta B^2-0.000175501ql^6\Delta B^2+2.61642\times10^{-05}ql^5\Delta B^2-0.000151694ql^4\Delta B^2-$$
$$0.000161975ql^3\Delta B^2+0.000427605ql^2\Delta B^2+0.0016374ql\Delta B^2+0.00186388\Delta B^2-$$
$$0.000271865\Delta qt^8\Delta B+6.46175\times10^{-05}\Delta qt^7\Delta B+0.00024116\Delta qt^6\Delta B+0.000240509\Delta qt^5\Delta B-$$
$$0.000131476ql^4\Delta qt^4\Delta B-1.02385\times10^{-05}ql^3\Delta qt^4\Delta B-8.33841\times10^{-05}ql^2\Delta qt^4\Delta B+$$
$$1.67774\times10^{-05}ql\Delta qt^4\Delta B+5.0835\times10^{-05}\Delta qt^4\Delta B+0.000220168ql^4\Delta qt^3\Delta B+$$
$$0.000248984ql^3\Delta qt^3\Delta B+0.000103066ql^2\Delta qt^3\Delta B-0.000181392ql\Delta qt^3\Delta B-$$
$$0.000119662\Delta qt^3\Delta B-0.000203907ql^4\Delta qt^2\Delta B-0.000190458ql^3\Delta qt^2\Delta B+0.000427361ql^2\Delta qt^2\Delta B+$$
$$0.000553154ql\Delta qt^2\Delta B+0.000232744\Delta qt^2\Delta B-0.000159803ql^4\Delta qt\Delta B-0.000274049ql^3\Delta qt\Delta B-$$
$$0.000201409ql^2\Delta qt\Delta B+0.00119639ql\Delta qt\Delta B+0.000805873\Delta qt\Delta B-0.000234194ql^8\Delta B-$$
$$3.49241\times10^{-05}ql^7\Delta B+0.000284516ql^6\Delta B-7.53026\times10^{-05}ql^5\Delta B+0.000151511ql^4\Delta B+$$
$$0.000103479ql^3\Delta B-8.1764\times10^{-05}ql^2\Delta B-0.00229247ql\Delta B-0.00193273\Delta B+$$
$$3.91128\times10^{-05}ql^2\Delta qt^8+4.808\times10^{-05}ql\Delta qt^8+0.000177892\Delta qt^8-8.52208\times$$
$$10^{-06}ql^2\Delta qt^7-7.09359\times10^{-05}ql\Delta qt^7+8.20805\times10^{-06}\Delta qt^7-1.04906\times10^{-05}ql^2\Delta qt^6+$$
$$1.4973\times10^{-05}ql\Delta qt^6-0.000157856\Delta qt^6+6.0172\times10^{-05}ql^2\Delta qt^5+5.04307\times$$
$$10^{-05}ql\Delta qt^5-1.28068\times10^{-05}\Delta qt^5+0.000106321ql^4\Delta qt^4+7.61288\times10^{-05}ql^3\Delta qt^4+$$
$$7.85808\times10^{-05}ql^2\Delta qt^4-6.18722\times10^{-05}ql\Delta qt^4-7.23798\times10^{-06}\Delta qt^4-0.000126451ql^4\Delta qt^3-$$
$$0.000162061ql^3\Delta qt^3-0.000136534ql^2\Delta qt^3+7.28316\times10^{-05}ql\Delta qt^3-7.44136\times$$
$$10^{-05}\Delta qt^3+4.37787\times10^{-05}ql^8\Delta qt^2+6.47746\times10^{-05}ql^7\Delta qt^2-5.47124\times$$
$$10^{-05}ql^6\Delta qt^2-0.000103773ql^5\Delta qt^2+0.000182459ql^4\Delta qt^2+0.000332405ql^3\Delta qt^2-$$

$$0.000155585ql^2\Delta qt^2 - 0.000528351ql\Delta qt^2 - 0.000173342\Delta qt^2 - 9.06463\times10^{-05}ql^8\Delta qt - 9.18625\times10^{-05}ql^7\Delta qt + 4.91132\times10^{-05}ql^6\Delta qt + 0.000153591ql^5\Delta qt + 1.66816\times10^{-06}ql^4\Delta qt - 0.000110246ql^3\Delta qt + 1.67276\times10^{-05}ql^2\Delta qt - 0.00065211ql\Delta qt - 0.000378374\Delta qt + 0.00011546ql^8 - 6.79246\times10^{-06}ql^7 - 0.000228526ql^6 - 4.54201\times10^{-05}ql^5 - 3.29703\times10^{-05}ql^4 + 7.14734\times10^{-05}ql^3 + 0.000106582ql^2 + 0.00152204ql + 0.00178699.$$

Bibliography

[1] Aho, A.V., et.al., **2007**: Compilers: principles, techniques and tools (2^{nd} Ed.) Pearson, Boston.

[2] Albrecht, B.A., **1988**: Observations of marine stratocumulus clouds during FIRE. Bull.Ame.Meteo.Soc. 69:618-626.

[3] Arakawa, A., **2004**: The cumulus parameterization problem: past, present and future. J. Clim., 17:2493-2525.

[4] Arnold, L., **2001**: Hasselmann's program revisited: The analysis of stochasticity in deterministic climate models, in Imkeller, P., (Ed.): Stochastic climate models. Birkhauser.

[5] Arnold, L., **2002**: Random dynamical systems. Springer-Verlag, Berlin.

[6] Bachmann, O., Wang, P.S., Zima E.V., **1994**: Chains of recurrences - a method to expediate the evaluation of closed form functions, in ACM Proceedings of the International conference on symbolic and algebraic computing (ISSAC) 1994. ACM press, Oxford, UK.

[7] Backus, J.W., et.al., **1963**: Revised report on the algorithmic language ALGOL 60. The Computer Journal, 5:349-367.

[8] Bacon, F., **1620**: Novum organum. Translated by Urbach, P., and Gibson, J., (1994). Open court publishing company.

[9] Battles, Z., Trefethen, L.N., **2004**: An extension of Matlab to continuous functions and operators. SIAM Journal of Scientific Computing, 25:1743-1770.

[10] Berz, M., Hoffstatter, G., **1998**: Computation and application of Taylor polynomials with interval remainder bounds. Reliable Computing, 4:83-97.

[11] Bhattacharya, R., Majumdar, M., **2007**: Random dynamical systems: Theory and applications. Cambridge University Press, Cambridge.

[12] Blanchet. B., et.al., **2002**: Design and implementation of a special-purpose static program analyser for safety-critical real-time embedded software. Lecture Notes in Computer Science, 2566:85-108.

[13] Blieberger, J., **2002**: Data-flow frameworks for worst-case execution time analysis. Real Time Systems, 22:183-227.

[14] Bohren, C.F., Huffman, D.R., **1998**: Absorption and scattering of light by small particles. Wiley.

[15] Bony, S., Dufresne, J., **2005**: Marine boundary layer clouds at the heart of tropical cloud feedback uncertainties in climate models. Geo. Res. Let. 32, L20806, doi:10.1029/2005GL023851.

[16] Boyd, J.P., **2001**: Chebyshev and Fourier spectral methods (2^{nd} edition). Dover publications.

[17] Bracciali, A., Ferrari, G., Tuosto, E., **2008**: A symbolic framework for multi-faceted security protocol analysis. International Journal of Information Security, 7:55-84.

[18] Bretherton, C.S., Austin, P., Siems, S.T., **1995**: Cloudiness and marine boundary layer dynamics in the ASTEX Lagrangian experiments. Part II: Cloudiness, drizzle, surface fluxes and entrainment. J.Atmos.Sci., 52:2724-2735.

[19] Bretherton, C.S., Pincus, R., **1995**: Cloudiness and marine boundary layer dynamics of the ASTEX Lagrangian experiments. Part I: Synoptic setting and vertical structure. J.Atmos.Sci., 52:2707-2723.

[20] Brisbarre, N., Joldes, M., **2010**: Chebyshev interpolation polynomial-based tools for rigorous computing. Proceedings of the 35^{th} International Symposium on Symbolic and Algebraic Computation, Munich, 25-28 July 2010.

[21] Buizza, R., Miller, M., Palmer, T.N., **1999**: Stochastic representation of model uncertainties in the ECMWF ensemble prediction system. Q.J.R.Meteorol.Soc., 125:2887-2908.

[22] Bush, W.R., Pincus, J.D., Sielaff, D.J., **2000**: A static analyser for finding dynamic programming errors. Software: Practice and Experience, 30:775-802.

[23] Cadar, C., et.al., **2006**: Automatically generating inputs of death. Proceedings of ACM Conference on Computer and Communications Security, 2006.

[24] Canetti, R., Herzog. J., **2010**: Universally composable symbolic security analysis. Journal of Cryptology, DOI: 10.1007/s00145-009-9055-0.

[25] Cargo, G.T., Shisha, O., **1966**: The Bernstein form of a polynomial. Journal of Research of the National Bureau of Standards - B: Mathematics and Mathematical Physics, 708:79-81.

[26] Chalmers, A.F., **1999**: What is this thing called science? (3^{rd} Ed.) 288pp. Open university press, Milton Keynes.

[27] Chassignet, E.P., Garrao, Z,D., **2001**: Viscosity parameterization and the gulf stream separation. Proceedings of the 'Aha Huliko' a Hawaiian winter workshop, U. of Hawaii, January 2001.

[28] Cheney, E.W., **2000**: Introduction to approximation theory (2^{nd} Ed.) American Mathematical Society.

[29] Clenshaw, C.W., **1962**: Chebyshev series for mathematical functions. Vol.5 of National Physical Laboratory Mathematical Tables, HMSO, London.

[30] Collins, G.E., **1975**: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. Lecture Notes in Computer Science, 33:134-183.

[31] Collins, W.D., **2001**: Parameterization of generalized cloud overlap for radiative calculations in general circulation models. J.Atmos.Sci., 58:3224-3242.

[32] Corless, R.M., et.al., **1999**: Approximate polynomial decomposition. Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC) 1999, 213-219.

[33] Cornelius, H., Lohner, R., **1984**: Computing the range of values of real functions with accuracy higher than second order. Computing, 33:331-347.

[34] Cousot, P., **1996**: Abstract Interpretation. ACM Computing Surveys (CSUR), 28:324-328.

[35] Cousot, P., Cousot, R., **1977**: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. Conference Record of the Fourth ACM Symposium on the Principles of Programming Languages.

[36] Dancy, J., **1985**: Introduction to contemporary epistemology. Blackwell, Oxford.

[37] Dufrence, J.L., Bony, S., **2008**: An assessment of the primary sources of spread of global warming estimates from coupled atmosphere-ocean models. J.Clim., 21:5135-5144.

[38] Emanuel, K.A., **1994**: Atmospheric Convection. Oxford University Press, Oxford.

[39] Epstein, C., Miranker, W.L., Rivlin, T.J., **1982a**: Ultra-arithmetic I: Function data types. Mathematics and Computers in Simulation, 24:1-18.

[40] Epstein, C., Miranker, W.L., Rivlin, T.J., **1982b**: Ultra-arithmetic II: Intervals of polynomials. Mathematics and Computers in Simulation, 24:19-29.

[41] Epstein, E.S., **1969**: Stochastic dynamic prediction. Tellus, 21: 739-759.

[42] Fahringer, T., Scholz, B., **2003**: Advanced symbolic analysis for compilers. Lecture Notes in Computer Science, 2628:1-125.

[43] Floyd, R.W., Biegel, R., **1994**: The language of machines: An introduction to computability and formal languages. W.H. Freeman and Company, 706pp.

[44] Gao, S., et.al., **2004**: Approximate factorization of multivariate polynomials via differential equations. Proceedings of the International Conference on Symbolic and Algebraic Computing (ISSAC) 2004. ACM press, Oxford, UK.

[45] Gathen, J.V., Gutierrez, J., Rubio, R., **2003**: Multivariate polynomial decomposition. Applicable Algebra in Engineering, Communication and Computing (AAECC), 14:11-31.

[46] Goldstein, M., Rougier, J., **2009**: Reified Bayesian modelling and inference for physical systems. J. Statistical Planning and Inference, 139:1221-1239.

[47] Gregory, D., Rowntree, P.R., **1990**: A mass flux convection scheme with representation of cloud ensemble characteristics and stability-dependent closure. Mon. Wea. Rev., 118:1483-1506.

[48] Guillaume, P., Huard, A., **1998**: Generalized multivariate Pade approximants. Journal of Approximation theory, 95:203-214.

[49] Haghighat, M.R., Polychronopoulos, C.D., **1996**: Symbolic analysis for parallelizing compilers. ACM Transactions on Programming Languages and Systems, 18:477-518.

[50] Hasselmann, K., **1976**: Stochastic climate models. Part I: Theory. Tellus, 28: 473-485.

[51] Hartmann, D.L., Short, D.A., **1980**: On the use of Earth radiation budget statistics for studies of clouds and climate. J.Atmos.Sci., 37:1233-1250.

[52] Hinchey, M., et.al., **2008**: Software engineering and formal methods. Communications of the ACM, 51:54-59.

[53] IPCC, **2007**: Climate change 2007: The physical science basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, Cambridge. 996 pp.

[54] ISO/IEC **1998**: ISO/IEC International standard 14882: Programming languages - C++. American National Standards Institute, New York.

[55] Jeffrey, R.C., **1990**: The logic of decision (2^{nd} Edition). University of Chicago Press.

[56] Kay, M., **1986**: Algorithm schemata and data structures in syntactic processing, in Grosz et.al (Eds.): Readings in Natural Language Processing. 35-70, Morgan Kaufamnn, San Francisco.

[57] Klien, R., **2008**: A unified approach to meteorological modelling based on multiple-scales asymptotics. Advances in Geosciences, 15:23-33.

[58] Klein, S.A., Hartmann, D.L., **1993**: The seasonal cycle of low stratiform clouds. J.Clim., 6:1587-1606.

[59] Klemp, J.B., Wilhelmson, R.B., **1978**: The simulation of three-dimensional convective storm dynamics. J.Atmos.Sci., 35:1070-1096.

[60] Knutti, R., **2008**: Oral presentation at a meeting of the Royal Statistical Society. London, 24th June 2008.

[61] Koshyk, J.N., Hamiltom, K., **2001**: The horizontal kinetic energy spectrum and spectral budget simulated by a high-resolution troposphere-stratosphere-mesosophere GCM. J.Atmos.Sci., 58:329-348.

[62] Kraichnan, R.H., **1967**: Inertial ranges in two-dimensional turbulence. Physics of Fluids, 10:1417-1423.

[63] Krishnamourti, T.N., Bounouna, L., **1995**: An introduction to numerical weather prediction techniques. CRC Press.

[64] Kropa, J.C., **1978**: Calculator algorithms. Mathematics Magazine, 51:106-109.

[65] Kyriakopoulos, K., Psarris, K., **2009**: Nonlinear symbolic analysis for advanced program parallelization. IEEE Transactions on Parallel and Distributed Systems, 20:623-640.

[66] Kuhn, T.S., **1962**: The structure of scientific revolutions. 168pp. University of Chicago Press, Chicago.

[67] Larson, V.E., Kotenberg, K.E., Wood, N.B, **2007**: An analytic longwave radiation formula for liquid layer clouds. Mon.Wea.Rev., 135:689-699.

[68] Lasota, A., Mackey, M.C., **1985**: Probabilistic properties of deterministic systems. Cambridge University Press, Cambridge.

[69] Lecerf, G., **2007**: Improved dense multivariate polynomial factorization algorithms. J.Symbolic Computation, 42:477-494.

[70] Lemoine, D.M., **2010**: Climate sensitivity distributions dependence on the possibility that models share biases. J.Clim., 23:4395-4415.

[71] Lilly, D.K., **1968**: Models of cloud-topped mixed layers under a strong inversion. Q.J.R.Meteorol.Soc., 94:292-309.

[72] Lin, Q., Rokne,J.G., **1995**: Methods for bounding the range of a polynomial. Journal of Computational and Applied Mathematics, 58:193-199.

[73] Li, M., Vitanyi, P., **1997**: An Introduction to Kolmogorov complexity and its applications. Springer-Verlag, New York.

[74] Lorenz, E.N., **1963**: Deterministic non-periodic flow. J.Atmos.Sci., 20:130-141.

[75] Lueker, G.S., **1980**: Some techniques for solving recurrences. Computing Surveys, 12:419-436.

[76] Mason, J.C., Handscombe, D., **2003**: Chebyshev polynomials. CRC Press LLC, Florida.

[77] Matos, A.C., **2007**: Multivariate Frobenius-Pade approximants: properties and algorithms. J.Computational and Applied Mathematics, 202:548-572.

[78] Mayo, D., **1996**: Error and the growth of experimental knowledge. 509pp. University of Chicago Press. Chicago.

[79] McDonald, W.F., **1938**: Atlas of climatic charts of the oceans. Washington DC:US Dept of Agriculture, Weather Bureau.

[80] Merrill, J., **2003**: GENERIC and GIMPLE: A new tree representation for entire functions. Proceedings of the GCC Developer's Summit, May 25-27, 2003. Ottawa, Canada. 171-180.

[81] Meyers, S.D., Kelly, B.G., O'Brien, J.J., **1993**: An introduction to wavelet analysis in oceanography and meteorology with application to the dispersion of Yanai waves. Mon.Wea.Rev., 121:2858-2866.

[82] Metropolis, N., Ulam, S., **1949**: The Monte-carlo method. Journal of the American Statistical Association, 44:335-341.

[83] Modersheim, S., Vigano, L., **2009**: The open-source fixed-point model checker for symbolic analysis of security protocols. Lecture Notes in Computer Science, 5705:166-194.

[84] Moeng, C.H., et.al., **1996**: Simulation of a stratocumulus-topped planetary boundary layer: intercomparison among different numerical codes. Bull.Ame.Meteo.Soc., 77:261-278.

[85] Moore, R.E., **1966**: Interval analysis. Prentice-Hall, Englewood Cliffs, NJ.

[86] Mosses, P. D., **1990**: Denotational semantics, in Leeuwen (Ed.): Handbook of Theoretical Computer Science, Vol.B. Elsevier, 575-631.

[87] Nastrom, G.D. Gage, K.S., Jasperson, W.H., **1984**: Kinetic energy spectrum of large and mesoscale atmospheric processes. Nature, 310:36-38.

[88] Nastrom, G.D., Gage, K.S., **1985**: A climatology of atmospheric wavenumber spectra of wind and temperature observed by commercial aircraft. J. Atmos. Sci., 42:950-960.

[89] Nikiforakis, N., **2007**: Lecture course given at the first UJCC-NCAS summer school on climate modelling. 10-21 September, 2007, University of Cambridge, UK.

[90] Oakley, J., O'Hagan, A., **2002**: Bayesian inference for the uncertainty distribution of computer model outputs. Biometrica, 89:769-784.

[91] O'Brien, F., **1993**: The third policeman. Harper-Collins, London, UK.

[92] Palmer, T.N., **2001**: A nonlinear dynamical perspective on model error: A proposal for non-local stochastic-dynamic parameterisation in weather and climate prediction models. Q.J.R.Met.Soc., 127:279-304.

[93] Palmer, T.N., **1999**: A nonlinear dynamical perspective on climate prediction. J. Clim., 12:575-591.

[94] Palmer, T.N., **2006**: Predictability of weather and climate from theory to practice, in Palmer, T.N., Hagedorn, R. (Eds.): Predictability of weather and climate. Cambridge University Press.

[95] Platte, R.B., Trefethen, L.N., **2010**: Chebfun: a new kind of numerical computing, in Progress in Fitt, A.D. et.al. (Eds.): Industrial Mathematics at ECMI 2008, 1083pp., Springer-Verlag, Berlin.

[96] Pop, S., Cohen, A., Silber, G.A., **2005**: Induction variable analysis and delayed abstractions. ACM Transactions on Architecture and Code Optimization, 5:1-30.

[97] Pop, S., et.al., **2006**: The new framework for loop nest optimization in GCC: from prototyping to evaluation. Proceedings of the 12^{th} Workshop on Compilers for Parallel Computers (CPC'06), Spain, January 2006.

[98] Press, W.H., et.al. **2007**: Numerical recipes: The art of scientific computing, 3^{rd} ed. Cambridge University Press, NY.

[99] Rababa, A., **2003**: Transformation of Chebyshev-Bernstein polynomial basis. Computational Methods in Applied Mathematics, 3:608-622.

[100] Rall, L.B., **2006**: Perspectives on automatic differentiation: past, present, future? Lecture Notes in Computational Science and Engineering, 50:1-14.

[101] Randall, D., Khairoutdinov, M., Arakawa, A., Grabowski, W., **2003**: Breaking the cloud parameterization deadlock. Bull.Amer.Meteo.Soc., 84:1547-1564.

[102] Rice, H.G., **1953**: Classes of recursively enumerable sets and their decision problems. Trans.Amer.Math.Soc., 74:358-366.

[103] Seidenberg, A., **1954**: A new decision method for elementary algebra. Annals of Mathematics, 60:365-374.

[104] Shutts, G.J., Palmer, T.N., **2007**: Convective forcing fluctuations in a cloud resolving model: Relevance to the stochastic parameterization problem. J.Clim., 20:187-202.

[105] Skamrock, W.C., Klemp, J.B., **1994**: Efficiency and accuracy of the Klemp-Wilhelmson time-splitting technique. Mon.Wea.Rev., 122:2623-2630.

[106] Smith, R.N.B, **1990**: A scheme for predicting layer clouds and their water content in a general circulation model. Q.J.R.Meteorol.Soc., 116:435-460.

[107] Smith, A.P., **2009**: Fast construction of constant bound functions for sparse polynomials. J.Glob.Optim., 43:445-458.

[108] Smolyak, S.A., **1963**: Quadrature and interpolation formulas for tensor products of certain classes of functions. Soviet Math Doklady (Doklady Akademii Nauk), 4:240-243.

[109] Solomonoff, R., **2005**: Algorithmic probability, AI and NKS. Lecture given at the Midwest NKS conference 2005.

[110] Solomonoff, R.J., **2008**: Three kinds of probabilistic induction: Universal distributions and convergence theorems. The Computer Journal, 51:566-570.

[111] Sommeria, G., Deardorff, J.W., **1977**: Subgrid-scale condensation in models of nonprecipitating clouds. J.Atmos.Sci., 34:344-355.

[112] Stahl, V., **1995**: Interval methods for bounding the range of polynomials and solving systems of nonlinear equations. Phd Thesis, Johannes Kepler University, Linz.

[113] Stevens, B., **2002**: Entrainment in stratocumulus-topped mixed layers. Q.J.R.Meteorol.Soc., 128:2663-2690.

[114] Stevens, B., et.al., **2005**: Evaluation of large-eddy simulations via observations of nocturnal marine stratocumulus. Mon.Wea.Rev., 133:1443-1462.

[115] Stull, R.B., **1988**: An introduction to boundary layer meteorology. Kulwer Academic Publishers, Netherlands.

[116] Tarski, A., **1951**: A decision method for elementary algebra and geometry. University of California Press, California.

[117] Trefethen, L.N., **2007**: Computing numerically with functions instead of numbers. Mathematics in Computer Science, 1:9-19.

[118] Turing A.M., **1936**: On computable numbers, with an application to the entsheidungsproblem. Proc. of the London Mathematical Soc., 2-42:230-265.

[119] Van Engelen, R.A, **2000**: Symbolic evaluation of chains of recurrences for loop optimization. Technical Report TR-000102, Dept. of Computer Science, Florida State University.

[120] Van Engelen, R.A., **2001**: Efficient symbolic analysis for optimizing compilers. Lecture Notes in Computer Science, 2027:118-132.

[121] Van Engelen, R.A., **2004**: The CR# algebra and its application in loop analysis and optimization. Technical Report TR-041223, Dept. of Computer Science, Florida State University.

[122] Wasilkowski, G.W., Wozniakowski, H., **1995**: Explicit cost bounds of algorithms for multivariate tensor product problems. Journal of Complexity, 11:1-56.

[123] Weiss, G.H., **1994**: Aspects and applications of the random walk. North-Holland, Amsterdam.

[124] Werbos, P.J., **2006**: Backwards differentiation in AD and neural nets: past links and new opportunities. Lecture Notes in Computational Science and Engineering, 50:15-34.