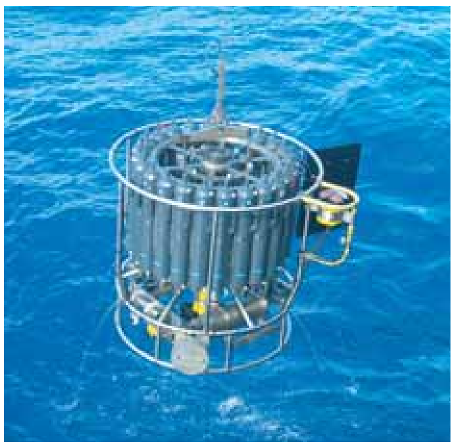
Max-Planck-Institut
für Meteorologie

MAX-PLANCK-GESELLSCHAFT

International Max Planck Research School
on Earth System Modelling

Error Estimation in Geophysical Fluid Dynamics through Learning

Florian Rauser



Berichte zur Erdsystemforschung 97/2011

Reports on Earth System Science

Hinweis

Die Berichte zur Erdsystemforschung werden vom Max-Planck-Institut für Meteorologie in Hamburg in unregelmäßiger Abfolge herausgegeben.

Sie enthalten wissenschaftliche und technische Beiträge, inklusive Dissertationen.

Die Beiträge geben nicht notwendigerweise die Auffassung des Instituts wieder.

Die "Berichte zur Erdsystemforschung" führen die vorherigen Reihen "Reports" und "Examensarbeiten" weiter.

Notice

The Reports on Earth System Science are published by the Max Planck Institute for Meteorology in Hamburg. They appear in irregular intervals.

They contain scientific and technical contributions, including Ph. D. theses.

The Reports do not necessarily reflect the opinion of the Institute.

The "Reports on Earth System Science" continue the former "Reports" and "Examensarbeiten" of the Max Planck Institute.

Anschrift / Address

Max-Planck-Institut für Meteorologie
Bundesstrasse 53
20146 Hamburg
Deutschland

Tel.: +49-(0)40-4 11 73-0
Fax: +49-(0)40-4 11 73-298
Web: www.mpimet.mpg.de

Layout:

Bettina Diallo, PR & Grafik

Titelfotos:
vorne:
Christian Klepp - Jochem Marotzke - Christian Klepp
hinten:
Clotilde Dubois - Christian Klepp - Katsumasa Tanaka

Berichte zur Erdsystemforschung 97/2011

Error Estimation in Geophysical Fluid Dynamics through Learning

Florian Rauser

aus Ratingen

Hamburg 2010

Reports on Earth System Science 97/2011

ISSN 1614-1199

Florian Rauser
Max-Planck-Institut für Meteorologie
Bundesstrasse 53
20146 Hamburg
Germany

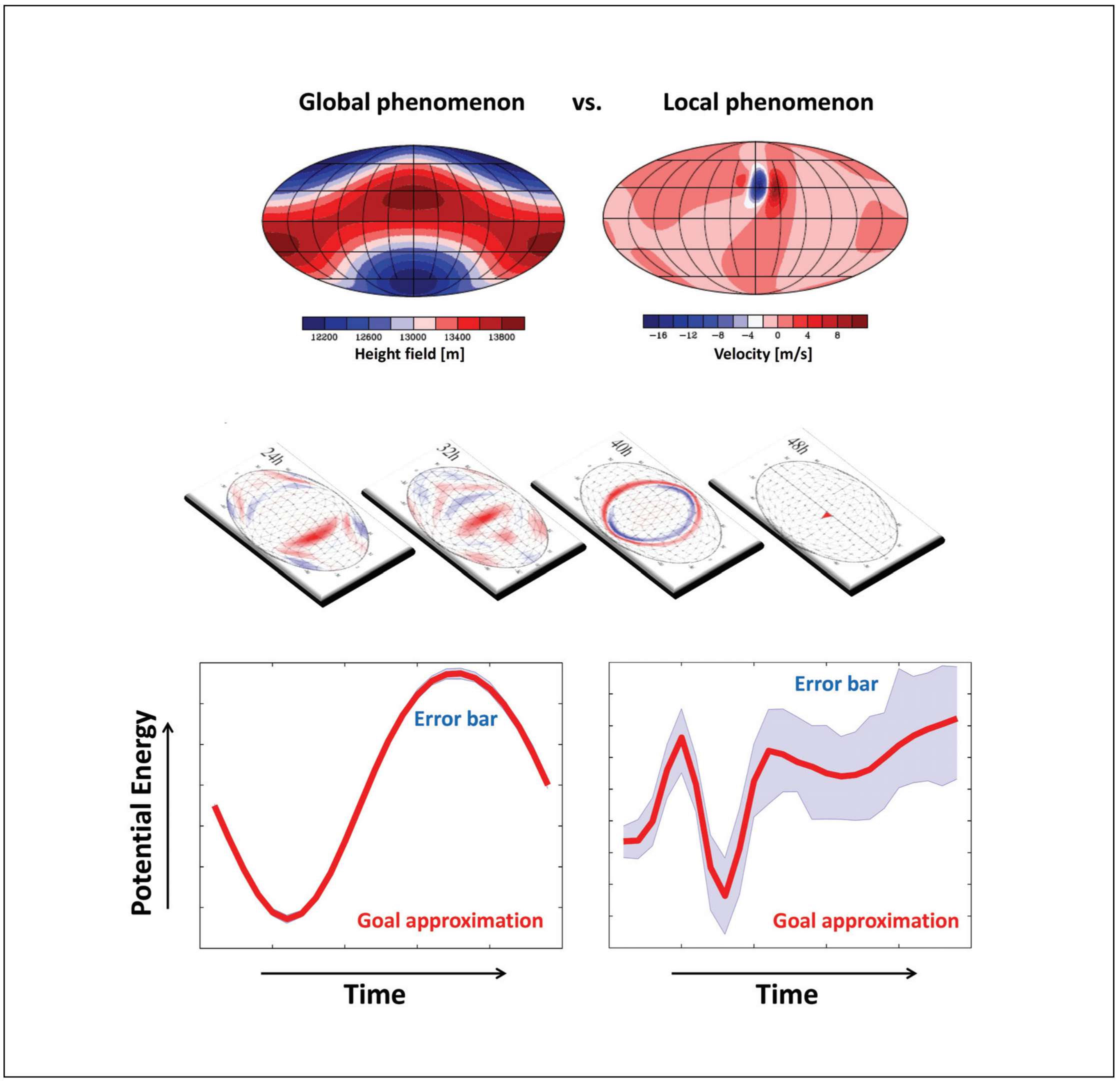
Als Dissertation angenommen
vom Department Geowissenschaften der Universität Hamburg

auf Grund der Gutachten von
Prof. Dr. Jochem Marotzke
und
Dr. Peter Korn

Hamburg, den 30. November 2010
Prof. Dr. Jürgen Oßenbrügge
Leiter des Departments für Geowissenschaften

ISSN 1614-1199

Error Estimation in Geophysical Fluid Dynamics through Learning



Florian Rauser

Hamburg 2010

Abstract

Current models of Geophysical Fluid Dynamics (GFD) lack the capability to quantify computationally induced errors. To address this issue, we present a new approach for numerical uncertainty quantification in GFD models: goal error estimation through learning.

We estimate the error in important physical quantities – so-called goals – as a weighted sum of local model errors. Our algorithm divides this goal error estimation into three phases. In phase one, we select a mathematical description of local model errors, either a deterministic functional of the solution or a stochastic process. In phase two, a learning algorithm adapts the selected mathematical description to the numerical experiment under consideration by determining the free parameters of the mathematical description. The learning algorithm analyzes a series of short numerical simulations on different resolutions. In phase three, goal errors are estimated using the learned parameters of the local error description. The deterministic description produces a goal error estimate that can be used to correct the original goal approximation. The stochastic description produces a goal error estimate ensemble that can be used to construct error bounds for the original goal approximation. The goal error ensemble is generated from a single model forward evaluation. The weights that are required for both approaches are the sensitivities of the goal with respect to local model errors. These sensitivities are calculated automatically with an Algorithmic Differentiation tool applied to the model's source code.

We evaluate both algorithms within ICOSWM, a numerical model for the shallow water equations on the sphere, and implement an Algorithmic Differentiation framework that calculates any required goal sensitivity. With our deterministic approach, we are the first to estimate time-dependent goal approximation errors for the spherical shallow water equations. With our stochastic approach, we are the first to estimate an ensemble of goal approximation errors from only one forward solution of the model. We combine our local error learning algorithm with stochastic physics and initial condition ensemble techniques and compare the results of both forward ensembles and our a posteriori ensemble. For our test cases, we see that an a posteriori ensemble – derived from a single model solution – delivers comparable results as a stochastic physics ensemble that requires multiple model solutions. We suggest the extension of our method to total model error and discuss the general nature of local model errors.

The algorithm proposed in this thesis bridges the gap between deterministic numerical methods and stochastic ensemble methods. It is generally applicable, easy to use, and simple compared to classical goal error estimation methods. Goal error estimation through learning is a first step towards automatic error bars for GFD models.

Contents

Chapter 1

Introduction

The Earth System Sciences attempt to describe and understand Earth as a combination of interrelating systems. The main focus is to understand the emerging interactions between subsystems such as atmosphere, hydrosphere, lithosphere, and biosphere. The Earth System Sciences rely heavily on computational models because it is impossible to measure all relevant physical quantities on all scales and the complexity of most of the Earth's subsystems often prevents analytical analysis. One central component of Earth System Modelling is Geophysical Fluid Dynamics (GFD), the science of the circulation of atmosphere and ocean (e.g., Pedlosky 1982). Geophysical Fluid Dynamics differ from Computational Fluid Dynamics (CFD) through the inclusion of rotational effects and a variety of effects due to Earth's geometry and scale (Charney et al. 1950). We use the term "GFD model" for any computational model that yields approximated solutions for the state of atmosphere or ocean.
Computational modelling applies numerical algorithms to solve problems that cannot be solved analytically. In this thesis, we develop a new method for GFD models that helps to combat one major problem of computational models: the reliability of numerical outputs.

A natural part of scientific thinking is that the uncertainty in the magnitude of a measured quantity is crucial to determine the physical significance of the measurement itself. The use of standard error bars and confidence intervals is widely accepted as a prerequisite to accept measured data as being representative for a real physical process. The same principle holds for all numerical experiments. The uncertainty in the magnitude of a simulated quantity is crucial to determine the physical significance of the simulation itself. A numerical method should be able to attach an error bar to a given numerical output for a physical quantity of interest.
Computational fluid dynamics is an excellent successfull example: the industrial need to get reliable estimates of flow and drag for simulations of aerofoils has led to a multitude of methods to enable aerodynamics computational models to assess and control the error in important physical quantities (Giles et al. 2004). The development of GFD models, however, lags behind. Even though GFD models are important for decision

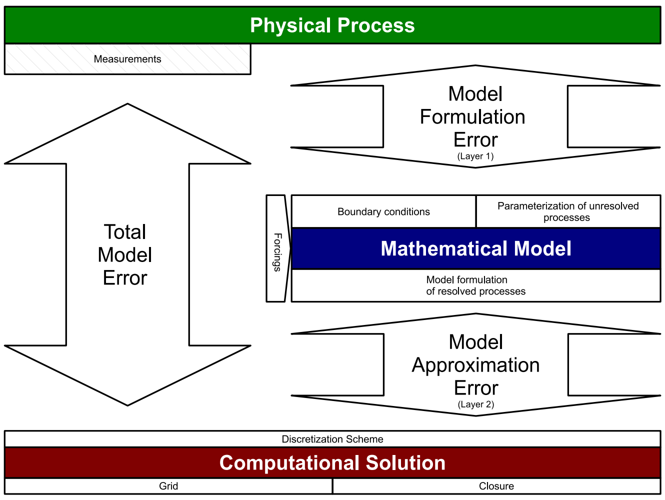


Figure 1.1: A sketch of the two layers of model errors and different error sources.

processes in society (Treut et al. 2007, IPCC 2007) they usually deliver numerical approximations for relevant physical quantities without attached error bars. It seems therefore highly appropriate to demand adequate uncertainty quantification for important physical quantities derived from GFD models.

1.1 The Hierarchy of Model Errors

The problem of uncertainty quantification will not vanish with better models or more computational power. No matter the increase in available computational power, some processes will remain unresolved and the multi-scale nature of GFD will lead to noticeable errors in macroscopic quantities. We need to develop ways to understand the causes of model uncertainty and means to quantify them.
Every model is wrong for different reasons. GFD models are complicated and there is a multitude of error sources that lead to final uncertainty in physical outputs. For a given physical process we find a hierarchy of descriptions of the process: the supposedly real values of the physical process, the measured values of a part of the state of this process, the model values as described with (mathematical) models and the approximated computational solutions. In this sense, total model error is the difference between the approximated computational solution of a model and the measured representation of reality. The evaluation of total model error cannot be completely separated from measurement errors, as reality is only quantifiable with measurements. Nevertheless, to structure the error that occurs during modelling, we can ignore measurement errors (see Figure 1.1). Oden and Prudhomme (2002) define two layers of model errors as 1) model formulation and specification error and 2) model approximation error.

1. **Model formulation and specification errors:** The first error layer incorporates everything that is part of the mathematical formulation and specification process: choice of prognostic variables, governing equations, parameterizations, forcings, boundary conditions, initial conditions.

2. **Model approximation errors:** If we have to use computational models to get an approximate solution of our model, we necessarily introduce errors based on finite degrees of freedom. Finite degrees of freedom imply unresolved scales which have to be parameterized. The choice of grid, discretization scheme and resolution can introduce additional errors. On top of that, computational models always face the problem of round-off error for real-valued numbers.

Deterministic chaos is another important concept which strongly contributes to the fact that every model is wrong. Chaotic systems are defined by their ability to allow small finite perturbations to grow exponentially. This means that all attempts to reduce model formulation, specification and approximation errors have only limited effects because the remaining small errors still grow exponentially.

1.2 Error Estimation and Optimization in GFD

We give a brief overview on current progress in GFD modelling with a focus on uncertainty quantification.

- A strong worldwide effort to build "next-generation" dynamical cores for GFD models tries to reduce the number of error sources in the model approximation layer (e.g., Bonaventura and Ringler 2005).

- The number of computational degrees of freedom has been steadily growing due to increased available computational power (Dongarra et al. 2010).

- The explicit inclusion of many subsystems into the GFD models has shifted many uncertainty sources from external forcings to internal parameterizations (e.g, Brovkin et al. 2009).

- Model intercomparison projects of all types have been able to quantify general model uncertainty (AMIP/PMIP/SMIP/APE/CMIP(e.g., Meehl et al. 2000)).

- Models try to estimate parameterization uncertainty through the inclusion of stochastic parameterizations into models (Buizza et al. 1999; Majda and Stechmann 2009).

- Data assimilation techniques are used to decrease specification and formulation error sources by fitting model outputs to data (Kalnay et al. 2007).

- New mathematical methods are applied to GFD problems to augment existing-model systems, e.g. low order modelling (Majda et al. 2009) or “super modelling” (van den Berge et al. 2010).

It is very difficult to find a methodology that separates uncertainty due to different error sources because it is nearly impossible to distinguish practically between the two error layers. Differences between output of a computational model and reality are always due to a combination of error sources in both layers at the same time. One step towards a differentiated analysis of the two error layers is the development of an error estimation technique that practically and conceptually separates both layers. To do this, we suggest to start with the model approximation error. The approximation error can be treated independently by setting the theoretical solution of a specified model as truth. The detailed analysis of approximation error is not yet standard in geophysical fluid dynamics for mainly two reasons: First, the approximation error has been deemed to be less important (= smaller) than the model formulation errors for long-term simulations. This is not a priori true for all models and all types of approximation errors as has been shown for example by (Rasch et al. 2006). Second, there are simply no techniques available that can be used for the technical variety of GFD models to estimate the approximation error. It is for these reasons that GFD models usually do not attach numerical error bars to numerical outputs. This has to change with the ever increasing importance of GFD models.

1.3 Thesis Objective: A New Kind of Uncertainty Quantification in GFD

The guiding research question:

> How can we formulate an algorithm for GFD models that estimates the numerical approximation error for important time-dependent physical quantities (regional or global)?

As a first step towards comprehensive error bars for GFD models, we develop a new error estimation algorithm for approximation errors that is applicable to existing GFD models, easy to use, and easy to implement. To achieve this, we employ the idea of algorithmic learning: the error estimation algorithm does as much of the work as possible without explicit user input. The algorithm is model-independent in the sense that it automatically learns everything that is specific about a given model from the model itself. This is the first time that the idea of learning is applied to approximation errors. To develop such an algorithm, we start with the analysis of the fluid dynamical kernel of GFD models. We focus on a CFD method that is called dual weight error estimation (Giles et al. 2004; Becker and Rannacher 2002). It estimates the approximation error for important physical quantities as a weighted aggregation of local model errors on each computational grid cell. Classically, local model errors are estimated using the model solution and information about the underlying model and discretization. To do this for complex, time-dependent problems is a difficult undertaking. To become applicable to GFD problems, this approach has to be substantially modified and extended. The original method depends strongly on expert knowledge of the underlying discretization to estimate local errors. At the same time, there is no general mathematical background for the types of discretizations and time-dependent problems that are typically encountered in GFD models. In outlining ways in which to translate the method to GFD models, we put specific focus on keeping the algorithm simple and general. We require from a potential algorithm to learn the properties of local model errors from the model itself and to calculate the aggregated goal approximation error automatically. The algorithm should not change the forward evaluation of a given model but calculate an error estimate for relevant goals a posteriori.

1.4 Thesis Outline

The Chapters 3, 4, and 5 of this thesis are written in the style of journal publications. As a consequence, they contain their own abstract, introduction and conclusions, and can be read largely independently of one another. Chapter 3 has been submitted to the Journal of Computational Physics 2010 and is currently under revisions (Rauser et al. 2011). Chapter 4 has been published in the ICCS conference proceedings 2010 (Rauser et al. 2010). Both Chapters deal with a deterministic approach to goal error estimation and focus on different aspects of the goal error estimation algorithm. Chapter 5 is currently being prepared for submission. It deals with a stochastic approach to goal error estimation. Chapter 2 gives a mathematical motivation of our algorithmic idea and Chapter 6 concludes the thesis with some final remarks. For editorial consistency, references to the publications underlying Chapter 3 and 4 have been changed to link to the respective Chapter.

- In **Chapter 2**, we introduce the general idea behind everything we do. We define a mathematical framework for a model and its different types of errors. We propose an error estimation method based on the concept of local model error learning and suggest two possible descriptions for local model errors: stochastic and deterministic.

- In **Chapter 3**, we use a deterministic description of local model errors and introduce our adaptation of dual weight error estimation with deterministic, empirical local error estimators. We explain our idea of modelling error production as functional of the flow state and describe how to learn the properties of this functional from comparison of model solutions. We show results for a numerical model of the spherical shallow water equations. We discuss the robustness of our method and show results for two different test cases.

- In **Chapter 4**, we discuss the second component of our error estimation technique in detail, the adjoint sensitivities. We show a new way to efficiently calculate adjoint solutions with AD tools by using discrete adjoints for large matrix multiplications. This is of general interest to GFD applications because most discretization schemes include the solution of large linear systems.

- In **Chapter 5**, we introduce a stochastic extension of dual weight error estimation, using a description of local model errors as a random process. We present a new learning algorithm that determines the model-specific properties of these local model error random processes from comparison of model solutions. This approach leads to an a posterior goal ensemble derived from a single run. We show results for a model of the spherical shallow water equations and two different test cases. We analyze the connection between our a posteriori ensembles and classical forward ensemble techniques.

The thesis closes with a summary of our main findings in **Chapter 6**, in which we also propose directions for future research.

Chapter 2

Problem Statement and Algorithm Proposal

In this chapter we introduce the mathematical nomenclature and motivate a general algorithm that will be described, extended and evaluated throughout this thesis. We start with the definition of a model. The process of defining a model is equivalent to a sequence of discriminating choices. We select a subsystem of physical quantities that we want to describe and call these variables "state vector" $\mathbf{q}$, defined on a space-time domain $\Omega \times T$. We then formulate mathematical rules that govern the evolution of the state vector. These rules can either be deduced from microscopic principles or heuristically from macroscopic observations. At this time, we also decide which external processes to parameterize and which to describe as external forcings. We decide for each variable if we want to use a deterministic or stochastic description. To finish the model specification, we determine the boundary conditions $\mathbf{q}_b$ and the initial conditions $\mathbf{q}^0$. These boundary conditions determine the behavior of the state vector $\mathbf{q}$ on the boundary $\partial\Omega \times \partial T$ of the domain $\Omega \times T$. We formulate the rules as a nonlinear (potentially stochastic) partial differential equation N

$$\begin{aligned} N(\mathbf{q}(\mathbf{x},t)) &= 0 \quad \text{on} \quad \Omega \times T, & (2.1)\\ \mathbf{q}(\mathbf{x},t) &= \mathbf{q}_b \quad \text{on} \quad \partial\Omega, & (2.2)\\ \mathbf{q}(\mathbf{x},t) &= \mathbf{q}^0 \quad \text{on} \quad \partial T. & (2.3) \end{aligned}$$

We introduce physical quantities of interest $J(\mathbf{q})$ that depend on the state vector $\mathbf{q}$. These derived quantities are called goals. Goals are affected by the whole variety of model formulation errors, leading to the following definition of goal model error

$$\varepsilon_1 := J_{true} - J. \tag{2.4}$$

The goal model error is the difference between the real value J_{true} of a physical quantity of interest and the solution of a model J. This error is rarely relevant in GFD modelling because it is only applicable to simple models with an analytical solution for $\mathbf{q}$. For more complex models, we need computational tools to help us get an approximative solution of our model. The next step is therefore the formulation of a discretized model N_Δ. We introduce a discrete representation $\mathbf{q}_\Delta$ of the state vector $\mathbf{q}$. We also choose a discrete representation $\Omega_\Delta \times T_\Delta$ of the domain $\Omega \times T$, a discrete boundary condition $P\mathbf{q}_b$ that is a projection of the continuous boundary condition and a discrete initial condition $\mathbf{q}^0_\Delta$. The details of the discretization process are problem-dependent and involve the choice of grid, differential operators, and interpolation operators. We formulate this discretization scheme as a general discrete operator N_Δ

$$\begin{aligned} N_\Delta(\mathbf{q}_\Delta) &= 0 \quad \text{on} \quad \Omega_\Delta, && (2.5)\\ \mathbf{q}_\Delta &= P\mathbf{q}_b \quad \text{on} \quad \partial\Omega_\Delta, && (2.6)\\ \mathbf{q}^0_\Delta &= P\mathbf{q}^0 \quad \text{on} \quad \partial T_\Delta. && (2.7) \end{aligned}$$

Given the discrete state vector $\mathbf{q}_\Delta$ we introduce the goal approximation $J_\Delta(\mathbf{q}_\Delta)$. We define the approximation error ε_2 and the total model error ε

$$\begin{aligned} \varepsilon_2 &:= J - J_\Delta, && (2.8)\\ \varepsilon &:= J_{true} - J_\Delta. && (2.9) \end{aligned}$$

This total model error is the standard quantity for error quantification in GFD modelling.
There are two possible strategies for error estimation: a priori and a posteriori. A priori error estimates are based on properties of the discrete model N_Δ and give general upper and lower error bounds for all possible solutions $\mathbf{q}_\Delta$. A posteriori methods use a specific solution $\mathbf{q}_\Delta$ and estimate the solution error or goal error after the model is solved. All error estimates throughout this thesis are a posteriori error estimates.

The problem statement

Given a model N, its discrete version N_Δ and physical quantities of interest J, our error estimation algorithm should produce an error estimate ε_{est} that quantifies the approximation error ε_2 in any goal J a posteriori. The algorithm should be applicable to existing GFD models without extensive code rewriting.

2.1 The Connection between Goal Errors and Local Errors

Given this problem statement, we connect goal approximation errors to local errors because this enables us to construct an algorithm based on local model errors for all possible goals. Local model errors are the errors at all computational grid cells. Our idea is a new interpretation of a classical error estimation technique called dual weight error estimation (Giles et al. 2004; Becker and Rannacher 2002; Oden and Prudhomme 2002) that estimates any goal error ε_2 as a weighted sum of local model errors

$$\varepsilon_2 \approx \left\langle \mathbf{q}^*_\triangle, \hat{N}_\triangle(\mathbf{q}_\triangle) \right\rangle_{\Omega \times T} \tag{2.10}$$

with an arbitrary scalar product $\langle ., . \rangle_{\Omega \times T}$, the adjoint solution $\mathbf{q}^*_\triangle$ as weights and the local model errors $\hat{N}_\triangle(\mathbf{q}_\triangle)$ (details to (2.10) can be found in Chapter 3). Equation (2.10) shows that an error estimate ε_{est} requires two components: First, the solution of the adjoint problem $\mathbf{q}^*_\triangle$ which is defined by the choice of model N, goal J, and scalar product $\langle ., . \rangle_{\Omega \times T}$. The adjoint solution $\mathbf{q}^*_\triangle$ represents the sensitivity of our goal with respect to local changes of the discrete state vector $\mathbf{q}_\triangle$. Second, a local error estimator $\hat{N}_\triangle$ that estimates local model errors and is dependent on the underlying discretization $N_\triangle$ and the discrete state vector $\mathbf{q}_\triangle$.
The first component $\mathbf{q}^*_\triangle$ is conceptually easy: we "only" need a method to calculate derivatives of any goal with respect to all local state vector changes. To do this for existing GFD models we suggest to use Algorithmic Differentiation (AD) to obtain the necessary goal sensitivities (details can be found in Chapter 4 and Appendix A). The second component $\hat{N}_\triangle$ is very hard to construct for time-dependent problems and depends strongly on the used discretization scheme. There is no mathematical basis for general local error estimates for all types of discretization. We deviate strongly from classical implementations in CFD to make the method useful for GFD applications. We replace $\hat{N}_\triangle$ by proposing empirical local error estimators $F_\triangle(\mathbf{q}_\triangle, \mathbf{p})$. This results in a new error estimate

$$\varepsilon_{est} := \langle \mathbf{q}^*_\triangle, F_\triangle(\mathbf{q}_\triangle, \mathbf{p}) \rangle_{\Omega \times T}\,, \tag{2.11}$$

with $\mathbf{p}$ a set of parameters that defines and specifies a problem-specific empirical local error estimator $F_\triangle$. The information about the flow regime, discretization and model is encapsulated in the parameter set $\mathbf{p}$. We call these local error estimators "empirical local error estimators" because the parameter set $\mathbf{p}$ is to be determined empirically and not from prior knowledge. Before we present our idea to determine the parameter set $\mathbf{p}$, we motivate two different types of empirical local error estimators.

2.2 Local Model Errors and Unresolved Processes

Following original work from (Mori 1965; Mori et al. 1974; Zwanzig 1973) and a review article from (Givon et al. 2004) we demonstrate that any model description implies local errors that can be described both stochastically and deterministically. The operator N (2.1) as introduced in the previous section is a general time-dependent stochastic differential equation for $\mathbf{q}$

$$N(\mathbf{q}) := \frac{d\mathbf{q}}{dt} + g(\mathbf{q}) + \gamma(\mathbf{q})\frac{dW}{dt} = 0, \tag{2.12}$$

with $W(t)$ a Wiener process, $g(\mathbf{q})$ and $\gamma(\mathbf{q})$ deterministic functionals of the solution $\mathbf{q}$. The discrete approximation N_Δ (2.5) solves only a part of the full dynamics of N. The full state vector $\mathbf{q} = (\mathbf{q}_\Delta, \hat{\mathbf{q}})$ can be written as a combination of a resolved part $\mathbf{q}_\Delta$ and an unresolved part $\hat{\mathbf{q}} \in \mathcal{Y}$ (with $\mathcal{Y}$ representing the space of unresolved scales). It is possible to exactly rewrite (2.12) into two different equations for $\mathbf{q}_\Delta$ and $\hat{\mathbf{q}}$

$$\begin{aligned} \frac{d\mathbf{q}_\Delta}{dt} + h(\mathbf{q}_\Delta, \hat{\mathbf{q}}) + \alpha(\mathbf{q}_\Delta, \hat{\mathbf{q}})\frac{dU}{dt} &= 0 \qquad (2.13) \\ \frac{d\hat{\mathbf{q}}}{dt} + i(\mathbf{q}_\Delta, \hat{\mathbf{q}}) + \beta(\mathbf{q}_\Delta, \hat{\mathbf{q}})\frac{dV}{dt} &= 0, \qquad (2.14) \end{aligned}$$

with U, V Wiener processes and h, i, α, β deterministic functionals of $\mathbf{q}_\Delta$ and $\hat{\mathbf{q}}$ that depend on the original functionals g, γ. Equations (2.13 – 2.14) both depend on the resolved and unresolved state vectors. Mori and Zwanzig have shown that it is possible to rewrite (2.13) to obtain a equation for the resolved state vector $\mathbf{q}_\Delta$ in which the direct dependencies on $\hat{\mathbf{q}}$ are eliminated

$$\frac{d\mathbf{q}_\Delta}{dt} + f(\mathbf{q}_\Delta) + M(\mathbf{q}_\Delta(t)) + O(\mathbf{q}_\Delta(0), \hat{\mathbf{q}}(0)) = 0. \tag{2.15}$$

The term $M(\mathbf{q}_\Delta(t)) = \int_0^t K(\mathbf{q}_\Delta(t-s), s)ds$ is called memory kernel and includes the memory of all interactions between $\mathbf{q}_\Delta$ and $\hat{\mathbf{q}}$. This means that to calculate the exact tendency of $\mathbf{q}_\Delta$ at a time t we need to know the exact evolution of $\mathbf{q}_\Delta$ up to this point. The term $O(\mathbf{q}_\Delta(0), \hat{\mathbf{q}}(0))$ is subject to an orthogonal dynamics equation that acts on the unknown initial state of the unresolved scales $\hat{\mathbf{q}}(0)$ at initial time. The solution of the orthogonal dynamics can be interpreted as noise because the initial data for the full problem is not known. The memory kernel is a noise with memory of the interactions between resolved and unresolved scales.
In the case of most GFD discretization methods the evolution equation for the discrete state vector $\mathbf{q}_\Delta$ includes only the explicit effects of the resolved scales $f(\mathbf{q}_\Delta)$

$$N_\Delta(\mathbf{q}_\Delta) = \frac{d\mathbf{q}_\Delta}{dt} + f(\mathbf{q}_\Delta) = 0. \tag{2.16}$$

The representation of $f(\mathbf{q}_\Delta)$ is not perfect for all resolved scales. Together with the neglect of the effect of unresolved scales, this is the reason that numerical errors must occur.
The interpretation of discrete model equations as a low order approximation of the underlying model shows that local errors are the consequence of a combination of deterministic and random processes.

2.3 The Concept of Error Learning

Following the Mori Zwanzig formalism, we propose two strategies to estimate approximation errors: First, to use the deterministic interpretation of local errors to estimate approximation error. Second, to use the stochastic interpretation of local errors to quantify approximation error in a probabilistic setting. These methods can also be combined or mixed. This means for the error estimate (2.11) that the empirical local error estimators F_Δ should be either a deterministic, empirical function of the flow state $\mathbf{q}_\Delta$ or that the empirical local error estimators F_Δ should represent local random processes. In both cases, the mathematical form of the general class of local error descriptions depends on a parameter set $\mathbf{p}$. The concept of error learning means that we use model information to determine a problem-specific parameter set $\tilde{\mathbf{p}}$ to choose the correct local error description for the problem under consideration. With deterministic local error estimators, Equation (2.11) delivers a single error estimate that can also be used for error correction purposes. With stochastic local error estimators, Equation (2.11) yields an ensemble of error estimates.
The error estimation algorithm must show how to learn the parameter set $\tilde{\mathbf{p}}$ for a given model, model discretization, flow regime, flow state, and resolution.

The Algorithm Proposal

- **Phase 0:** Choose a reference truth.
- **Phase 1 - Specification:** Choose a functional form (deterministic) or a specific form of a random process (stochastic) for the empirical local error estimators $F_\Delta(\mathbf{q}_\Delta, \mathbf{p})$ for a given model N_Δ and goal J_Δ.
- **Phase 2 - Learning:** The model learns the characteristics of local model errors represented by a problem-specific parameter set $\tilde{\mathbf{p}}$.
- **Phase 3 - Application:** Estimate goal errors a posteriori with a variant of dual weight error estimation. To do this obtain sensitivities $\mathbf{q}^*_\Delta$ for any goal and model with respect to local model errors and calculate the scalar product with the local error estimators $F_\Delta(\mathbf{q}_\Delta, \tilde{\mathbf{p}})$.

The choice of reference truth is identical to the choice of the error type. The algorithm estimates model approximation errors if we choose reference model solutions as local reference truth. The algorithm estimates total model errors if we choose measurements as local reference truth. Throughout this thesis we use high-resolution solutions as local reference truth to estimate model approximation errors.

2.4 The Research Questions

The thesis is structured by the two possible strategies of Section 2.3. We use a discrete shallow water model as a prototype model N_Δ to approximate regional potential energy as physical quantity of interest J_Δ to evaluate both strategies.

1. Deterministic Error Correction of Goal Approximation Errors for GFD Models (Chapter 3 and Chapter 4)

We derive a deterministic version of the proposed algorithm with deterministic empirical local error estimators. This brings us to the following research questions:

- Can empirical functionals of the flow state be used to estimate goal approximation errors?
- How can the algorithm learn the properties of these functionals?
- Is the parameter set of these functionals dependent on flow-regime / goal / resolution?
- How do we obtain the sensitivities automatically and efficiently?
- How long are the error estimates of our algorithm useful?

2. Stochastic Uncertainty Quantification of Goal Approximation Errors (Chapter 5)

We derive a stochastic version of the proposed algorithm with stochastic empirical local error estimators. The stochastic interpretation of local errors yields goal error PDFs, which can be used to construct error bounds that constrain the goal approximation. The stochastic approach leads to the following research questions:

- Can a local error random process $\mathcal{P}$ be used to quantify goal approximation errors?
- How can the algorithm learn the properties of this random process?
- Is the parameter set of these random process dependent on flow-regime / goal / resolution?
- How long is the goal error ensemble of our algorithm useful?
- Can we use the local error learning algorithm to use classical ensembles to estimate goal approximation error?
- How does the computational cost of a posteriori goal ensembles compare to that of a stochastic physics forward ensemble?

Chapter 3

Predicting Goal Error Evolution from Near-Initial-Information: a Learning Algorithm

We estimate the discretization error of time-dependent goals that are calculated from a numerical model of the spherical shallow-water equations. The goal errors are described as a weighted sum of local model errors. Our algorithm divides goal error estimation into three phases. In phase one, we select deterministic functionals of the flow as a mathematical description of local model error estimators. In phase two, a learning algorithm adapts the selected functionals to the numerical experiment under consideration by determining the free parameters of the functionals. To do this, the learning algorithm analyzes a short numerical simulation at two different resolutions. In phase three, goal errors are estimated using the local error estimators with the parameters learned in phase two. The required weights are the sensitivities of the goal with respect to local model errors; these sensitivities are calculated automatically with an Algorithmic Differentiation tool applied to the model's source code.

We apply this new error estimation algorithm to two different shallow water test cases: solid-body rotation and zonal flow against a mountain. For the solid-body rotation we successfully estimate the error of simulated regional potential energy and can track its evolution for up to 24 hours. For the zonal flow against a mountain we also successfully estimate the error of simulated regional potential energy. From the comparison of the two test cases we see that the learning period must incorporate a similar flow state as the prediction period to enable useful goal error estimators.

Our algorithm produces goal error estimates without detailed knowledge of the employed discretization. We believe that this learning approach can be useful in adapting error estimation techniques to complex models.

3.1 Introduction

Numerical models of atmospheric and oceanic circulations are affected by a variety of error sources such as missing system components, closure problems, or heuristic physical parameterizations. The resulting total error of numerical models can be categorized into two components (Oden and Prudhomme 2002): the modelling error caused by the difference between model description and physical process, and the approximation error caused by the difference between the true model solution and the computational approximation. Both types of solution errors lead to errors in physical quantities of interest such as energy, vorticity, or transport quantities, that are derived from the model solution. These quantities are called “goals” and they characterize the state of the physical system. The approximation goal error is the difference between an approximated goal and its “true” value; they quantify how much we trust our model to approximate the true solution of the model formulation. In this paper, we show how to estimate goal errors for time-dependent solutions of a model of the rotating shallow water equations.

We present a new algorithm that estimates the approximation goal error for a given model solution *a posteriori*, and we evaluate the algorithm for the rotating shallow water equations. The major novel feature of our algorithm is that it “learns” model-specific properties of local error production by using information from a very limited time interval at the beginning of the simulation to estimate the goal error at the end of this simulation. The goal error at the end of the simulation is estimated as the weighted sum of the local error estimates of each grid cell in space and time. The local error estimators are described by a class of generic smoothness measures of the solution. They are weighted with the sensitivity of the goal to changes in the grid cells. The sensitivities are calculated with an Algorithmic Differentiation Griewank (2000) tool. The local error estimators are adapted toward the behavior of a given numerical model in a learning period. The learning period requires a short high-resolution integration of the model, where “short simulation” means an integration time significantly smaller than the full integration time and where “high-resolution” means a resolution that we cannot afford for the full integration time. By comparing the high-resolution solution with a standard-resolution solution we determine the free parameters of the local error estimators. The learning approach circumvents the error analysis of specific model discretizations, a difficult task for nonlinear model equations. The contribution of this paper is to introduce this idea of “learning” in the context of error estimation for time-dependent goals.

The general idea to estimate goal errors as weighted sum of local errors is known as “goal-oriented error estimation” or “dual-weighted-residual method” (Becker and

Rannacher 2002) and has been researched in the computational fluid dynamics (CFD) community for many years (Stewart and Hughes 1998; Giles and Pierce 2000; Giles et al. 2004; Venditti and Darmofal 2000). The method originates in the theory of finite element discretizations (Ainsworth and Oden 1997; Babuska and Rheinboldt 1978; Johnson et al. 1995), but attempts have been made to generalize the method to finite volume discretizations (Sonar and Sueli 1998). The method connects local error estimates in each computational grid cell with the output error in physical goals via the solution of a goal-dependent adjoint problem. Parallel to the extension of goal-oriented error estimation to various discretization schemes and different applications, the class of treated problems has also been extended from elliptic equations to steady and unsteady Euler and Navier-Stokes equations (Prudhomme and Oden 2002; Becker and Rannacher 2002; Mani and Mavriplis 2009). There are two common applications of goal-oriented *a posteriori* error estimation. In the first application, the local error estimates can be used to dynamically adapt the spatial grid in order to improve the solution and consequently the quality of the goal estimate. For geophysical problems, adaptive grid adaptation for a primitive equation ocean model was investigated in (Power et al. 2006). Recently, progress has been reported towards the dynamic adaptation of temporal grids (Mani and Mavriplis 2009). In the second application, an error estimate is constructed and then used as a correction/improvement for certain model outputs only. In (Giles et al. 2004) an error estimate for a time-evolving goal for a non-linear equation, namely the 1D Burgers equation, is investigated. To our best knowledge, we are the first to quantify numerical goal error evolution in a GFD environment for the spherical shallow water equations. Our work employs the general philosophy of goal-oriented error estimation and dual-weighted residual methods but it differs from previous work in the crucial construction of local error estimators. The construction of our "learning" goal error estimator does not directly rely on the structure of the underlying nonlinear Partial Differential Equation (PDE). This might appear as a drawback as we lose important structural information about the problem. On the other hand we believe that our understanding of these equations has not progressed towards the points where we are able to construct goal error estimators from analytical considerations. The potential drawback of the learning approach is furthermore compensated by the possibility to apply our algorithm to future problems that do not have a pure PDE-structure such as complex atmosphere/ocean circulation models that include physical parameterizations without underlying PDE structure. By a thorough analysis of numerical experiments we try to provide evidence that goal error prediction via learning algorithms is a potential alternative to classical discretization-based approaches.

The paper is organized as follows: in Section 3.2 we introduce the shallow water equations on a sphere and time-dependent solutions thereof as prototype GFD problems. We repeat in Section 3.3 the basics of general adjoint-based goal error estimation.

ICON grid properties			
Resolution	Number of cells	Average cell distance	Time step length
Δ1	320	1115.3 km	900 s
Δ2	1280	556.4 km	600 s
Δ3	5120	278.0 km	450 s
Δ4	20480	139.0 km	200 s
Δ5	81920	69.5 km	100 s
Δ6	327680	34.7 km	50 s

Table 3.1: Basic properties of the ICON discretization. The number of cells is identical to the height field degrees of freedom. Average cell distance is the average of all the distances between triangle cell centers.

We then define our new concept of empirical local error estimators and discuss their specific characteristics. We introduce our concept of goal error estimation with local error learning. In Section 3.4 we show that it is possible to estimate the goal error of low-resolution runs with our empirical local error estimators. We show results for different integration times and various regions. In Section 3.4.3 we conclude with a review of the strengths and weaknesses of our approach.

3.2 Problem Statement

The shallow water equations (SWE) on a rotating sphere serve as testbed for our effort to extend CFD error analysis techniques to GFD problems. The SWE share significant properties of the global atmospheric and oceanic fluid system with more complex descriptions and are able to simulate large-scale flows (Pedlosky 1982). The SWE are typical for geophysical fluid dynamics but differ significantly from classical CFD applications because they include Coriolis effects on the sphere.

The inviscid SWE on the sphere Ω written in vector invariant form are

$$\begin{aligned}\frac{\partial \mathbf{v}}{\partial t} &= (\xi + f)\mathbf{k} \times \mathbf{v} - \nabla(gh + \frac{1}{2}|\mathbf{v}|^2) \\ \frac{\partial h}{\partial t} &+ \nabla \cdot (h\mathbf{v}) = 0.\end{aligned} \tag{3.1}$$

Here $\mathbf{v}$ is the horizontal velocity, ξ the vorticity, f the Coriolis parameter, $g = 9.81 m/s^2$ the gravitational acceleration and h the height of the fluid surface. The initial conditions are $\mathbf{v}(t_0) = \mathbf{v}_0$ and $h(t_0) = h_0$. We consider (3.1) on a time interval $T := [t_0, t_{end}]$ and with periodic spatial boundary conditions. The state vector $\mathbf{q} = (h, \mathbf{v})$ consists of the prognostic fields height and velocity. The hyperbolic partial differential equations (3.1) describe the flow of a single layer of fluid.

Our numerical framework is ICOSWM (Bonaventura and Ringler 2005), a shallow water model on a triangular spherical grid with C-type staggering of the variables. ICOSWM shares the operators and the grid with ICON, a next-generation General Circulation Model. The grid is derived from an icosahedron (20 triangular cells) and then refined (Bonaventura and Ringler 2005). One refinement level is equivalent to a quadrupling of the number of cells by halving the triangle edge lengths. The lowest resolution $\Delta 1$ is a two-times refined icosahedron and has 320 cells. More details can be found in Table 3.1. ICOSWM uses a hybrid finite volume / finite difference method to approximate the SWE (3.1). ICOSWM calculates a solution vector $\mathbf{q}_\Delta = (h_\Delta, \mathbf{v}_\Delta)$ with the discrete height field h_Δ in the cell centres of our triangular grid and the normal velocities $\mathbf{v}_\Delta$ at the mid points of the triangular edges. The solution process is sequential in nature; the discrete model yields discrete time slices $\mathbf{q}^k_\Delta$ for each time step. In our notation, the solution vector $\mathbf{q}_\Delta = (\mathbf{q}^k_\Delta)_k$, $k = 1, ..., n$, incorporates all time slices $\mathbf{q}^k_\Delta$ and represents the discrete approximation of the full solution. For further details see (Giorgetta et al. 2009; Ripodas et al. 2009).

To evaluate our algorithm for a physically relevant goal, we introduce regionally averaged potential energy density $e_{pot} = gh^2$ at the end of the integration time t_{end} as a goal:

$$J(\mathbf{q}) := J(h(t_{end})) = \frac{g}{A(\Omega_0)} \int_{\Omega_0} h^2(\mathbf{x}, t_{end}) d\mathbf{x}, \tag{3.2}$$

where Ω_0 denotes an arbitrary subdomain of the sphere Ω and $A(\Omega_0)$ denotes the area of Ω_0. The goal depends directly only on the height field h as part of state vector $\mathbf{q}$. We omit the factor 1/2 in the definition of potential energy because a constant factor does not change the structural form of the goal functional and its error characteristics. The computational equivalent of Equation (3.2) is the numerical integration of an approximated discrete height field after n time steps h^n_Δ on the discrete subdomain $\Omega_{\Delta 0}$

$$J_\Delta(\mathbf{q}_\Delta) := J_\Delta(h^n_\Delta) = \frac{g}{A_\Delta(\Omega_{\Delta 0})} \sum_{i \in \Omega_{\Delta 0}} a_i \left(h^n_{\Delta,i}\right)^2, \tag{3.3}$$

where the a_i denote the grid cell areas, $h^n_{\Delta,i}$ is the value of the discrete height field after n time steps on the ith triangle. The discrete area $A_\Delta(\Omega_{\Delta 0}) = \sum_{i \in \Omega_{\Delta 0}} a_i$ is the sum of all triangle areas that are part of the subdomain $\Omega_{\Delta 0}$ and approximates the true area $A(\Omega_0)$. This midpoint integration is consistent with the assumptions that are made in the ICON model discretization. Throughout this paper, we calculate the regional potential energy goals for different areas on the sphere. All regions used in this chapter always have the size of a grid resolution $\Delta 1$ triangle. This allows us easy comparisons of model approximations for the same region at different resolutions without the need for a sophisticated interpolation algorithm. We define the goal error as the difference between (3.2) and (3.3)

$$\varepsilon := J_\triangle(\mathbf{q}_\triangle) - J(\mathbf{q}). \tag{3.4}$$

This is the difference between an exact evaluation of the analytical solution of our continuous problem and the approximated evaluation of the approximated solution of the discrete problem. The exact evaluation of $J(\mathbf{q})$ is usually impossible. The numerical approximation $J_\triangle(P\mathbf{q})$ of $J(\mathbf{q})$ produces errors even if the correct solution $\mathbf{q}$ is known, with P a projection operator that maps $\mathbf{q}$ on the same discrete grid as $\mathbf{q}_\triangle$. We can neglect this approximation error of the goal because it is small compared to the error that is caused by the solution error. Hence, the error in Equation (3.4) is approximated by

$$\varepsilon \approx J_\triangle(\mathbf{q}_\triangle) - J_\triangle(P\mathbf{q}). \tag{3.5}$$

For time-dependent flows, the goal is changing in time. We want to be able to estimate the error at the end of an arbitrary integration time. This leads to the following questions that define our problem

1. How can the error for time-dependent goals as defined in Equation (3.5) be estimated?
2. Is it possible to use these error estimates to correct goal approximations obtained from low-resolution solutions, i.e., to improve their quality to the quality of goals obtained from high-resolution solutions, without solving the underlying problem at this high resolution?
3. Over how long integration times can the error be consistently reduced?

We attempt a general answer to question one in Section 3.3. The answers to questions two and three are inherently test-case specific and are addressed in the results Section 3.4.

3.3 Deterministic Estimation of Goal Approximation Errors

In this section, we review the fundamentals of of goal-oriented a posteriori error estimation, following closely the finite volume derivation proposed in (Giles 1998), before we describe our learning goal error estimation algorithm.

3.3.1 Goal Errors and Local Error Estimators

The shallow water equations (3.1) can be described as a general nonlinear differential operator N acting on the state vector $\mathbf{q} = (h, \mathbf{v})$

$$N(\mathbf{q}(\mathbf{x}, t)) = 0, \qquad \mathbf{q}(\mathbf{x}, t_0) = \mathbf{q}^0, \tag{3.6}$$

on the domain $\Omega \times T$ with periodic spatial boundary conditions in $\mathbf{x}$, and $\mathbf{q}^0$ as the initial condition. The state vector $\mathbf{q}$ represents the solution on the complete space time domain. The corresponding discretized equations can be formalized as

$$N_\Delta(\mathbf{q}_\Delta) = 0, \qquad \mathbf{q}_\Delta^0 = \mathbf{q}^0, \tag{3.7}$$

with $\mathbf{q}_\Delta := (\mathbf{q}_\Delta^k)_k$ the full discrete solution vector, $\mathbf{q}_\Delta^k := (h_\Delta^k, \mathbf{v}_\Delta^k)$ the state vector time slice for time step k, N_Δ the discretized version of operator N and P a projection operator that maps the initial condition $\mathbf{q}^0$ on the discrete space. The discrete solution vector $\mathbf{q}_\Delta = (h_\Delta, \mathbf{v}_\Delta)$ represents the discrete solution for all time steps and spatial degrees of freedom. The dimensionality of $\mathbf{q}_\Delta$ is the number of time steps times spatial degrees of freedom. Equation (3.7) is valid for all elements of $\mathbf{q}_\Delta$, the dimensionality of $N_\Delta(\mathbf{q}_\Delta)$ is identical to the dimensionality of $\mathbf{q}_\Delta$. Equation (3.7) holds only up to machine precision or the precision of the iterative solver in case of an implicit discretization. We neglect both iteration and round-off error.

We introduce the pointwise solution error $\mathbf{e}_\Delta$ as

$$\mathbf{e}_\Delta := \mathbf{q}_\Delta - P\mathbf{q}, \tag{3.8}$$

with P again the projection operator that evaluates $\mathbf{q}$ on the same discrete grid as $\mathbf{q}_\Delta$ and $\mathbf{e}_\Delta$. This vector of pointwise errors incorporates the solution error in all points of space and time. The dependency of the goal error ε on the pointwise error $\mathbf{e}_\Delta$ can be calculated by linearizing J_Δ around the discrete solution $\mathbf{q}_\Delta$

$$\begin{aligned} \varepsilon = J_\Delta(\mathbf{q}_\Delta) - J_\Delta(P\mathbf{q}) &= J_\Delta(\mathbf{q}_\Delta) - J_\Delta(\mathbf{q}_\Delta - \mathbf{e}_\Delta) \\ &\approx \left\langle \left.\frac{\partial J_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta}, \mathbf{e}_\Delta \right\rangle_{\Omega\times T} . \end{aligned} \tag{3.9}$$

We introduce on the right hand side of (3.9) an arbitrary discrete scalar product $\langle ., . \rangle_{\Omega\times T}$ on the space-time domain. For our purposes, we use the Euclidean scalar product where all discrete vector components are weighted with the associated volume in the space-time domain (the product of cell area and time step length). We will from now on omit the explicit notation of the space-time domain $\Omega \times T$ unless needed for clarification. From Equation (3.9) we observe that we need both the sensitivities of our goal with respect to the solution errors and the pointwise solution errors itself to obtain an error estimate. Unfortunately, the solution error $\mathbf{e}_\Delta$ is hard to estimate, especially for time-dependent problems. It incorporates local error production, error advection, and local error accumulation at each grid point. It is advisable to replace the solution error by something that is easier to estimate. Therefore, we perform a linearization of the discrete operator N_Δ around $\mathbf{q}_\Delta$, using the definition of solution error (Equation (3.8))

$$\begin{aligned} N_\Delta(P\mathbf{q}) &= N_\Delta(\mathbf{q}_\Delta - \mathbf{e}_\Delta) \\ &\approx N_\Delta(\mathbf{q}_\Delta) - \left.\frac{\partial N_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta} \mathbf{e}_\Delta. \end{aligned} \tag{3.10}$$

The second term of the right hand side is a standard matrix vector product between a square matrix $\frac{\partial N_\Delta}{\partial \mathbf{q}_\Delta}$ and the discrete vector of pointwise errors. The square matrix can be assumed to be invertible; it is an upper triangular matrix because solutions at a time step n can only depend on time slices $\mathbf{q}^i_\Delta$ if $i <= n$. We use (3.7) and (3.10) to get

$$0 = N_\Delta(\mathbf{q}_\Delta) \approx N_\Delta(P\mathbf{q}) + \left.\frac{\partial N_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta} \mathbf{e}_\Delta. \tag{3.11}$$

We can now solve Equation (3.11) for the solution error $\mathbf{e}_\Delta$ and insert $\mathbf{e}_\Delta$ into Equation (3.9) to obtain an error estimate for ε without explicitly using the solution error

$$\begin{aligned} \varepsilon = J_\Delta(\mathbf{q}_\Delta) - J_\Delta(P\mathbf{q}) &\approx \left\langle \left.\frac{\partial J_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta}, - \left(\left.\frac{\partial N_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta}\right)^{-1} N_\Delta(P\mathbf{q}) \right\rangle \\ &= \left\langle {\mathbf{q}^*_\Delta}^T, N_\Delta(P\mathbf{q}) \right\rangle, \end{aligned} \tag{3.12}$$

with ${\mathbf{q}^*_\Delta}^T$ the transposed of the solution $\mathbf{q}^*_\Delta$ of the adjoint problem

$$\left(\left.\frac{\partial N_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta}\right)^T \mathbf{q}^*_\Delta + \left(\left.\frac{\partial J_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta}\right)^T = 0. \tag{3.13}$$

The adjoint problem can be derived from Equation (3.12) as

$$\begin{aligned} {\mathbf{q}^*_\Delta}^T &= - \left.\frac{\partial J_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta} \left(\left.\frac{\partial N_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta}\right)^{-1} \\ \Leftrightarrow {\mathbf{q}^*_\Delta}^T \left.\frac{\partial N_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta} &= - \left.\frac{\partial J_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta} \\ \Leftrightarrow \left(\left.\frac{\partial N_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta}\right)^T \mathbf{q}^*_\Delta + \left(\left.\frac{\partial J_\Delta}{\partial \mathbf{q}_\Delta}\right|_{\mathbf{q}_\Delta}\right)^T &= 0. \end{aligned} \tag{3.14}$$

The operator N_Δ in Equation (3.12) is applied to the analytical solution $\mathbf{q}$, which is not known. The resulting vector $N_\Delta(P\mathbf{q})$ is called the vector of truncation errors. Equation (3.12) shows that the goal error is approximately the scalar product of the adjoint sensitivities $\mathbf{q}^*_\Delta$ and the vector of truncation errors $N_\Delta(P\mathbf{q})$. The adjoint sensitivities serve as weights for the vector of truncation errors and connect the goal error with local truncation errors (Giles 1998). This connection is easier to use than Equation (3.9) because the vector of truncation errors is usually easier to estimate than the pointwise error used in Equation (3.9).

The adjoint problem (3.13) is also called dual problem to the primal problem that consists of the model (3.7) and the goal (3.3). Formally, the adjoint problem is the transposed linearized original problem. For a time-dependent problem, the adjoint system propagates backwards in time and is initialized and forced via the choice of the goal. For our specific problem of the global SWE, the adjoint problem has the same (periodic) spatial boundary conditions as the forward problem (3.6). For our type of forecast goal, the adjoint problem has one temporal initial condition, and its discrete version is defined as the derivative of the discrete goal at the last time step

$$\mathbf{q}_{\Delta}^{*\,n} = \frac{\partial J_{\Delta}^{n}(\mathbf{q}_{\Delta}^{n})}{\partial \mathbf{q}_{\Delta}^{n}}. \tag{3.15}$$

This temporal initial condition is defined at the end of the forward integration time of (3.7) and is sometimes called adjoint end condition. If one uses structurally different goals that incorporate information from more than the last time step, the goal also influences the adjoint solution as a forcing.

The derivation of the goal error estimate Equation (3.12) via a Taylor-series expansion is only a linear estimate, holding if the linear approximations of the operator and the goal functional J in (3.10) and (3.9) are justified. Giles et al. (2004) argue that higher order terms become negligible compared to the linear error estimate if the solution errors for the nonlinear primal problem and the linear adjoint problem are of the same order.

Every *a posteriori* error estimation technique needs to approximate the truncation-error vector $N_{\Delta}(P\mathbf{q})$ using the numerical solution $\mathbf{q}_{\Delta}$

$$N_{\Delta}(\mathbf{q}) \approx \hat{N}_{\Delta}(\mathbf{q}_{\Delta}). \tag{3.16}$$

The new operator $\hat{N}_{\Delta}$ has to be introduced because the naive evaluation of $N_{\Delta}(\mathbf{q}_{\Delta})$ is zero up to machine precision by definition. The approximation $\hat{N}_{\Delta}$ is called local error estimator or local residual estimator and estimates the errors at each computational grid point in space and time. The exact construction and derivation of this local error estimator traditionally depends on the discretization that is used. The description of local error estimators is a key feature of the whole methodology because it translates the problem of estimating goal errors into that of estimating local errors for one time step. It is at this point where our method deviates from previous work. To estimate the local error one can start from an analysis of the spatial and temporal discretization scheme to develop a measure for the error that takes into account different sources of numerical errors as well as their mutual interplay. For a nonlinear time-dependent problem such as the shallow water equations this is a rather complex task, and it would be even more difficult for a 3D GFD model. Additionally, some proposed methods that involve interpolation on higher resolution grids (Venditti and Darmofal 2000) are too expensive to be used for time-dependent problems. We therefore decide to take a different route and propose the construction of local error estimators that are based on smoothness measures of the flow solution and not on the model discretization and underlying PDE. These local error estimators feature degrees of freedom $\mathbf{p}$ that have to be learned from model behavior. This means that we approximate $\hat{N}_\Delta$ with cheap and simple functionals F_Δ of the discrete solution $\mathbf{q}_\Delta$, characterized by a set of parameters $\mathbf{p}$,

$$\hat{N}_\Delta(\mathbf{q}_\Delta) := F_\Delta(\mathbf{q}_\Delta, \mathbf{p}). \tag{3.17}$$

The structure of these local error estimators and the learning algorithm for the parameters $\mathbf{p}$ are explained in section 3.3.3. Inserting(3.17) into (3.12) leads to

$$\varepsilon_{est} := \left\langle {\mathbf{q}_\Delta^*}^T, F_\Delta(\mathbf{q}_\Delta, \mathbf{p}) \right\rangle \approx \varepsilon, \tag{3.18}$$

with ε_{est} the estimate for the goal error ε. We rewrite (3.18) to use the error estimate to improve the original goal approximation J_Δ

$$J(\mathbf{q}) \approx J_\Delta(\mathbf{q}_\Delta) - \varepsilon_{est}. \tag{3.19}$$

Our error estimate ε_{est} must have the correct sign and magnitude for error correction. This prevents the use of relative local error estimators that are commonly used for grid adaptation purposes. Equation (3.18) shows that our algorithm needs two ingredients to estimate the goal error:

1. The sensitivities $\mathbf{q}_\Delta^*$ that are the solution of the adjoint problem (3.13) and
2. A local error estimator $F_\Delta(\mathbf{q}_\Delta, \mathbf{p})$.

3.3.2 The Algorithm Proposal

Our version of goal-oriented error estimation uses a general class of functionals $F_\Delta(\mathbf{q}_\Delta, \mathbf{p})$ as local error estimators. We now propose an algorithm that selects a specific functional $F_\Delta(\mathbf{q}_\Delta, \tilde{\mathbf{p}})$ from this class by learning a parameter set $\tilde{\mathbf{p}}$ that includes information on the model under consideration - from the model under consideration.

Goal Error Estimation Algorithm

1. Define a general class of deterministic functionals $F_\Delta(\mathbf{q}_\Delta, \mathbf{p})$ of the flow that can be used as error estimators.
2. Learn a specific parameter set $\tilde{\mathbf{p}}$ from the model in short runs at varying resolution.
3. Use Algorithmic Differentiation to obtain automatic goal sensitivities $\mathbf{q}^*_\Delta$. Calculate scalar product $\left\langle {\mathbf{q}^*_\Delta}^T, F_\Delta(\mathbf{q}_\Delta, \tilde{\mathbf{p}})\right\rangle$ between the local error estimators and these sensitivities. Use this scalar product as a goal error estimate or as error correction to improve the approximated goal.

3.3.3 Step 1: Functional Form of Local Error Estimators

As a first step, we need a definition of our general class of local error estimators $F_\Delta(\mathbf{q}_\Delta, \mathbf{p})$. Our approach grants complete freedom at this point to construct functionals that relate flow states to error production. For the goal "potential energy" (3.3) we do not use the complete state vector $\mathbf{q}_\Delta$ but construct a local error estimator as a functional of the h_Δ field only, i.e. we estimate the local errors in velocities to be zero

$$F_\Delta(\mathbf{q}_\Delta, \mathbf{p}) := F_\Delta(h_\Delta, \mathbf{p}). \tag{3.20}$$

The dimensionality of F_Δ is equal to the number of time steps times the spatial degrees of freedom. The spatial component of the scalar product for the error estimate (3.12) therefore reduces to the dimensionality of the height field solution h_Δ. While we need to compute the full adjoint solution including adjoint velocities for a correct solution of the adjoint height field, we do not need to save the adjoint velocities for the scalar product. We motivate this reduction of scalar product dimensionality because GFD models classically feature a large number of variables in the state vector. For an efficient usage of our method, it is necessary to reduce the learning aspect to the dominating parts of the vector; here the variables that are used to calculate the goal. This reduction is not fundamentally necessary for ICOSWM but the general applicability of our method depends crucially on the computational costs that have to be lower than the full high-resolution simulation. We choose a parameter set consisting of only one scalar scaling factor $\mathbf{p} = \omega$. We use local error estimators $F_\Delta(h_\Delta, \omega)$ that are of the form

$$F_\Delta(h_\Delta, \omega) = \omega \tilde{F}_\Delta(h_\Delta), \tag{3.21}$$

where $\tilde{F}_\Delta$ is a smoothness measure of the height field (with the same dimensionality as h_Δ). The smoothness measure $\tilde{F}_\Delta$ takes the spatial structure of the error into account while the term ω scales this error indication to a given discretization and grid resolution. It is here, in the scaling factor ω, that the information about discretization and the grid resolution enters our error estimation algorithm.

The term ω is conceptually dependent on the discretization and grid resolution. If the user of our algorithm is interested in applying the error estimates to many different resolutions it is possible to model the resolution dependency of ω as a function of a typical grid length (see for example the power laws of typical grid length for error estimates in (Sonar and Sueli 1998)). We refrain from this approach because we are usually only interested in estimating the error of goals derived from a standard resolution. If we want to estimate errors for different resolutions we use separate scaling factors for different resolutions.

We suggest three different smoothness measures $\tilde{F}^i_\Delta$:

1. Regions of large spatial gradients are a potential candidate for large errors. We construct a smoothed field $\bar{h}_{\Delta,i} = \frac{1}{3}\sum_{j=1}^{3} h_{\Delta,j}$ in each cell that is the average over the respective three neighbor cell values $h_{\Delta,j}$. The first smoothness measure is the difference between this averaged field and the solution in the cell itself
$$\tilde{F}^1_\Delta(h_{\Delta,i}) := h_{\Delta,i} - \bar{h}_{\Delta,i}. \tag{3.22}$$

2. The second smoothness measure is a simplification of the finite element gradient estimator method. We approximate the size of the height gradient in a cell i with the finite differences of the height field at the three cell edges δh_j
$$\tilde{F}^2_\Delta(h_{\Delta,i}) := \max_{j=1,2,3} \delta h_j. \tag{3.23}$$

3. Regions of large temporal gradients are another potential candidate for large errors. The third smoothness measure is therefore based on temporal rates of change and is given by
$$\tilde{F}^3_\Delta(h^k_\Delta) := \frac{h^{k+1}_\Delta - h^k_\Delta}{\Delta t}, \tag{3.24}$$
with k the time step and Δt the time step length. The last time step value $\tilde{F}^3_\Delta(h^n_\Delta)$ is set to be $\tilde{F}^3_\Delta(h^{n-1}_\Delta)$.

The three proposed local error estimators are

$$\begin{aligned} F^1_\Delta(h_{\Delta,i}) &= \omega\left(h_{\Delta,i} - \bar{h}_{\Delta,i}\right), & (3.25)\\ F^2_\Delta(h_{\Delta,i}) &= \omega \max_{j=1,2,3} \delta h_j, & (3.26)\\ F^3_\Delta(h^k_\Delta) &= \omega \frac{h^{k+1}_\Delta - h^k_\Delta}{\Delta t}. & (3.27) \end{aligned}$$

The smoothness measures above are similar to error indicator functions used for grid refinement purposes (e.g., Power et al. 2006). The new aspect here is the concept to "tune" a general local error indicator quantitatively with a parameter ω for a specific model.

3.3.4 Step 2: Learning the Properties of Local Error Estimators

As a second step, we need to learn the correct parameter $\tilde{\mathbf{p}}$ that completely determine the local error estimators $F_\Delta(\mathbf{q}_\Delta, \tilde{\mathbf{p}}) = F_\Delta(h_\Delta, \tilde{\omega})$ for a specific model. The learning algorithm can be adjusted accordingly if the parameter set consists of more components. We suggest to train the local error estimators with short high and low-resolution runs on an arbitrarily chosen region:

1. Perform low and high-resolution runs for a short time interval and obtain the height field solutions $h_{\Delta,low}$ and $h_{\Delta,high}$.
2. Calculate the goal approximations $J_{\Delta,high}(h_{\Delta,high})$ and $J_{\Delta,low}(h_{\Delta,low})$ using the two solutions $h_{\Delta,low}$ and $h_{\Delta,high}$. The difference $\tilde{\varepsilon} = J_{\Delta,low} - J_{\Delta,high}$ is an approximation of the true error ε.
3. Perform the low-resolution adjoint run to obtain a low-resolution adjoint height solution $h^*_{\Delta,low}$.
4. Calculate a smoothness measure with the low-resolution solution $\tilde{F}_{\Delta,low}(h_{\Delta,low})$. Calculate the approximate scaling weight $\tilde{\omega}$ by dividing the estimated error $\tilde{\varepsilon}$ by the low-resolution error estimate

$$\tilde{\omega} = \frac{\tilde{\varepsilon}}{\left\langle \mathbf{q}^*_{\Delta,low}, \tilde{F}_{\Delta,low} \right\rangle}. \qquad (3.28)$$

This procedure can be repeated for different regions to get an averaged and more robust estimate of $\tilde{\omega}$. The computational cost of this learning algorithm is cheaper than a full solution at the higher resolution. During the learning period, the adjoint problem needs to be solved only for a few time steps on the low-resolution grid. The result of the learning algorithm is the determination of one degree of freedom that connects smoothness properties with quantitative model errors. After the learning is done once for a given model discretization and flow regime, the error estimator $F_\Delta(h_\Delta, \tilde{\omega})$ can be used to estimate goals in this flow regime, i.e., different goals, different regions, and longer and varying integration times.

3.3.5 Step 3: Automatic Goal Sensitivities

The last step is to calculate the scalar product (3.18), which requires the solution of the adjoint solution at the low resolution for the full period of the simulation. We need the goal sensitivities for a given numerical model with respect to local changes in the discrete state vector. We suggest to use Algorithmic Differentiation (AD) software to directly get an approximation of the adjoint solution (e.g., Griewank 2000). AD software interprets the execution of a discretized model as a series of simple elemental operations. The output of an AD tool is the derivative of any model variable with respect to any number of different model variables or variable instances. These derivatives or sensitivities are calculated by the chain rule as a simple concatenation of derivatives of the basic operations of the employed programming language. The process yields an approximation of $\mathbf{q}^*_\Delta$. The advantage of using an AD adjoint version of our model is that we are as close as possible to the discretized solution of our model. Additionally, this solution method of the adjoint problem does not involve new coding and is expected to be easier and less error-prone.

For our specific mode, we have implemented an adjoint version of the shallow water model ICOSWM. ICOSWM-AD is a parallel checkpoint runtime adjoint version of ICOSWM obtained with the AD-enabled NAGware fortran95 compiler (Rauser et al. 2010). The adjoint sensitivities have been successfully compared to sensitivities obtained from a tangent-linear solution of the model and finite-difference gradient approximations.

3.4 Results and Discussion

We apply our new error estimation technique to two test cases that are commonly used in the GFD community.

- Test case 1 (TC1): an unsteady solid body rotation as introduced in example 3 of (Laeuter et al. 2005).
- Test case 2 (TC2): zonal wind against a mountain as described in test case 5 in (Williamson and Drake 1992).

The topography, height field initial condition, and meridional velocity after 12 hours of our test cases are plotted in Figure 3.1. Within these two test cases, we want

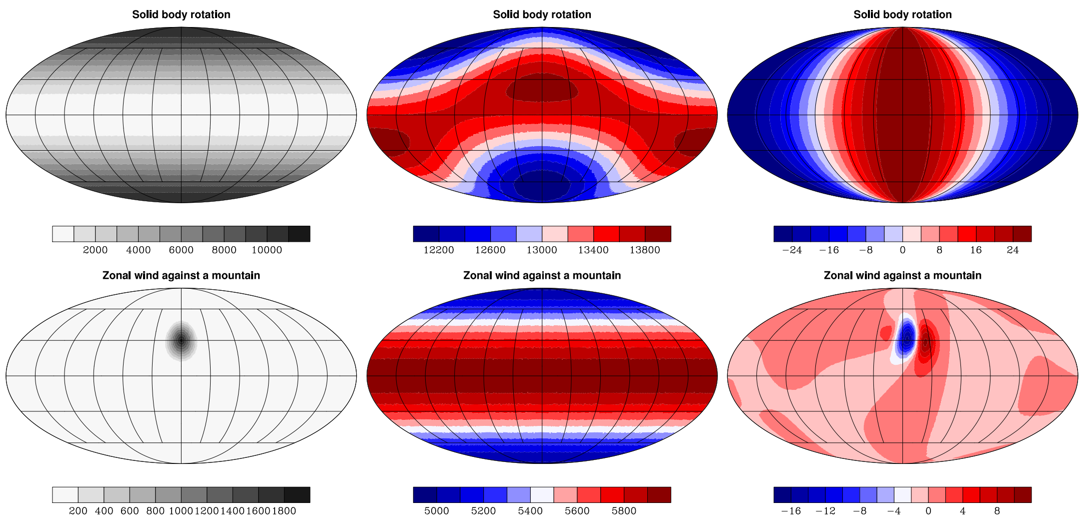


Figure 3.1: Topography (left), height field initial condition (middle), meridional velocity after 24hours (right). Top row for unsteady solid body rotation (TC1), bottom row for zonal wind against a mountain (TC2).

to showcase that our method can be used for error estimation of goals derived from periodic, global flow patterns as in TC1, but also for local phenomena as the evolution around the mountain in TC2.

3.4.1 Unsteady Solid Body Rotation (TC1)

The unsteady solid body rotation is a periodic test case that propagates a wave-like structure in the height field westwards with a periodicity of 24 hours. It is called "unsteady solid body rotation" because the unsteady solution is derived from a solid body rotation of the atmosphere around a rotation axis that is inclined (45°) with respect to the Earth's rotation axis. The westwards propagation is due to this inclination: the height field appears to be moving westwards because the eastward velocities of the inclined coordinate system are smaller than the actual Earth's rotation. The exact derivation can be found in (Laeuter et al. 2005). All goals that are derived from this height field at a fixed latitude show the same 24 hour period, similar amplitudes but differing phases, as can be seen in the left panel of Figure 3.2.

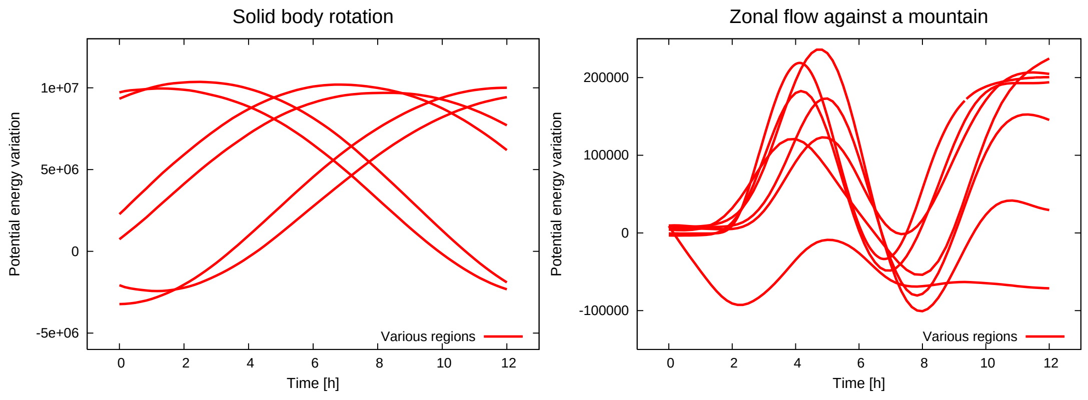


Figure 3.2: Variation in potential energy around the reference height for TC1 (left) and TC2 (right) for a 12 hour evolution.

Local error evolution as a function of the flow state

First we show that local error evolution can be modelled as a functional of the flow $F_\Delta(\mathbf{q}_\Delta, \mathbf{p})$. We define the discrete time derivative of the pointwise error (3.8)

$$\dot{\mathbf{e}}^n_\Delta = \frac{1}{\Delta t}\left(\mathbf{e}^{n+1}_\Delta - \mathbf{e}^n_\Delta\right). \tag{3.29}$$

Equation (3.29) eliminates accumulated errors in time, allowing us a comparison between pointwise error evolution and local error estimators. We see that the initial error evolution appears to be random (Figure 3.3), probably a consequence of the initialization of the test case. Later times show the emergence of an error pattern that is related to the flow pattern. The error development after 6 hours shows a coherent pattern that is structurally related to the flow of our test case, showing the same wave number but different phase. The behavior of the three smoothness measures F^1_Δ, F^2_Δ and F^3_Δ is also shown in Figure 3.3. The smoothness measure F^3_Δ, which is based on temporal gradients, looks most promising because it exhibits similar large-scale structures as the error evolution and exhibits the smallest amount of grid-scale noise (=differing signs or strongly changing values between neighboring cells). The smoothness measure F^2_Δ, which is based on spatial gradients, shows a large amount of grid scale noise with strongly differing contributions from neighboring cells. The superiority of the F^3_Δ smoothness measure might be due to the smooth and wavetype character of our test case and the low-resolution. We conclude that our smoothness measures show some structure that is related to the flow but also show significant differences in noise characteristics.

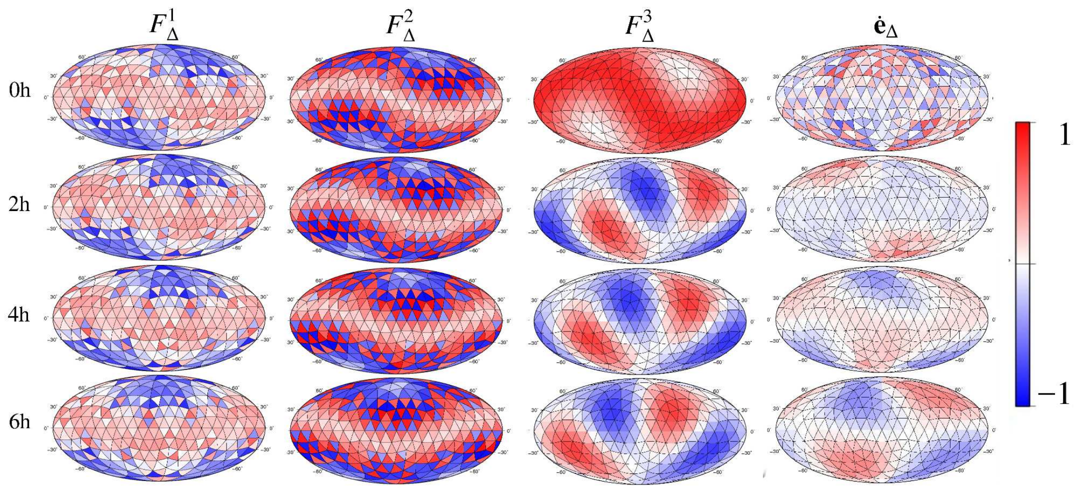


Figure 3.3: TC1: the different local error estimators and the true error rate of change for a 6h interval. The plotted fields are normalized.

ICON triangle center coordinates for cell sets on resolution Δ1											
	Cell ID ⟶										
Cell Set ↓	1	2	3	4	5	6	7	8	9	10	11
S_1 (Lon)	108°	91°	-163°	180°	163°	-91°	-108°	-125°	-19°	-36°	53°
(Lat)	11°	10°	10°	11°	10°	10°	11°	10°	10°	11°	10°
S_2 (Lon)	-135°	-137°	-131°	-118°	-108°	-91°	-108°	-125°	-80°		
(Lat)	50°	34°	39°	37	42°	57°	52°	57°	50°		
S_3 (Lon)	-91°	-108	-125								
(Lat)	57°	52	57								

Table 3.2: Cell center coordinates for all ICON Δ1 triangles used for goal calculations

TC1: Improvement Percentage for cells in set S_1														
	Gauging Set ↓	LEE	Cell ID ⟶											
			1	2	3	4	5	6	7	8	9	10	11	$\oslash$
ω_1	S_1	F^3_Δ	50	53	90	70	70	81	81	50	85	67	88	71
ω_*	S_1	F^3_Δ	62	64	90	86	86	54	99	16	96	82	92	75

Table 3.3: Improvement percentage for different cells of cell set S_1 and TC1. The ω_1 row shows the application of a trained weighting factor ω_1 for a 6h integration. The ω_* shows the application of the optimal weighting factor ω_* after 6h. Values between 0 and 100 mean an improvement (100 = we completely correct the error). $\oslash$ shows the average improvement. Gauging set: the set of cells that is used to gauge the local error estimator. LEE: the used local error estimator.

Learning the properties of local error estimators

Now we demonstrate that a specific parameter set $\tilde{\mathbf{p}}$ of the functional local error description of Section 3.3.3 can be determined by the learning algorithm proposed in Section 3.3.2. We determine a learned scaling factor ω_1 for the estimator F^3_Δ (3.27) for resolution $\Delta 1$ with a one-hour run. For this learning period, we choose a set of grid cells S_1 that are part of a zonal band at a latitude of about 10N for all learning and robustness experiments, see Table 3.2. This specific test case with an analytical solution allows us also to calculate the optimal weight ω_* as the ratio of true goal error and the smoothness measure. Table 3.3 shows the results after 6 hours for the trained ω_1 and with the optimal ω_*. We can see that the average improvement for our method has an upper limit of 75% for optimal weights. This level can nearly be reached by the gauged error estimates: the improvement averages to 71%. The single region results for ω_1 do not deviate more than 20% from the optimal weights. We conclude that for this test case it is possible to find a useful scaling factor and to improve the goal estimates by applying our goal errors.

Goal error estimates with a single scaling factor

We now provide evidence that the proposed simple smoothness measures lead to useful goal error estimates in a variety of applications, given a specific value of $\tilde{\omega}$. All following experiments are conducted using the single value ω_1 obtained in the previous section.

We have to be careful to get answers that are not simply tuned to fit the data due to the degrees of freedom we introduced with our parameter set $\tilde{\mathbf{p}}$. Our method must be better than other simple hypotheses with similar degrees of freedom. With $\tilde{\omega}$ we have one degree of freedom, which means we have to beat one tuned number that estimates

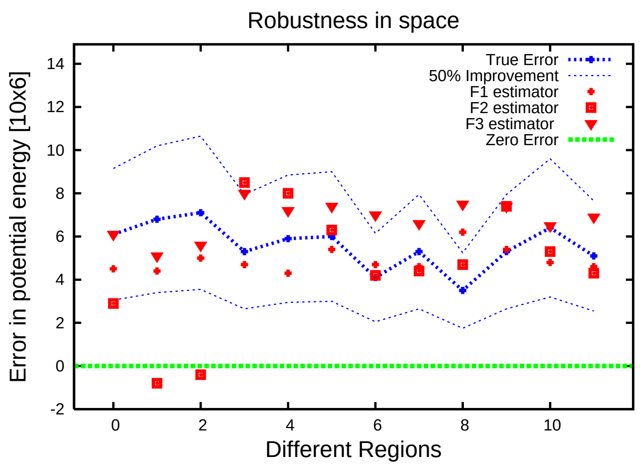


Figure 3.4: TC1: True error and error estimates of potential energy for different regions. The thick line depicts the true error. The dots are estimated errors for the three different classes of error estimators. The two thin lines indicate a goal improvement of at least 50% if the error estimate would be used for error correction.

the error for all regions, times, and goals. We thus have to look for robustness of various error estimates for a single, learned value $\tilde{\omega}$. The local error estimators F^1_Δ, F^2_Δ or F^3_Δ should lead to robust error estimates for different regions on the sphere, different goals, different resolutions, and different integration times.

The robustness requirements suggest four experiments: variation in location for fixed resolution and integration time, variation in the formulation of the goal for fixed resolution and integration time, variation in time for fixed resolution and location, and finally variation in resolution for fixed integration time. All experiments are shown at the lowest resolution $\Delta 1$ ($\approx$ 1100km grid spacing). For all experiments that follow we evaluate the "true" value of a goal with the analytical solution at the reference resolution $\Delta 6$ of ($\approx$ 35km).

Spatial robustness

To test the robustness with respect to region, we define a fixed integration time of six hours and compare different estimators. Our local error estimators work sufficiently well to estimate the errors of low-resolution runs of our model (Figure 3.4). We can observe from Figure 3.4 that most error estimates improve the quality of the goal approximation by at least 50%. Two error estimates from estimator F^2_Δ are close to zero and therefore do not improve the quality of the goal approximation. The wide spread throughout the different estimators (3.25) leaves no local error estimator the clear winner. The F^1_Δ

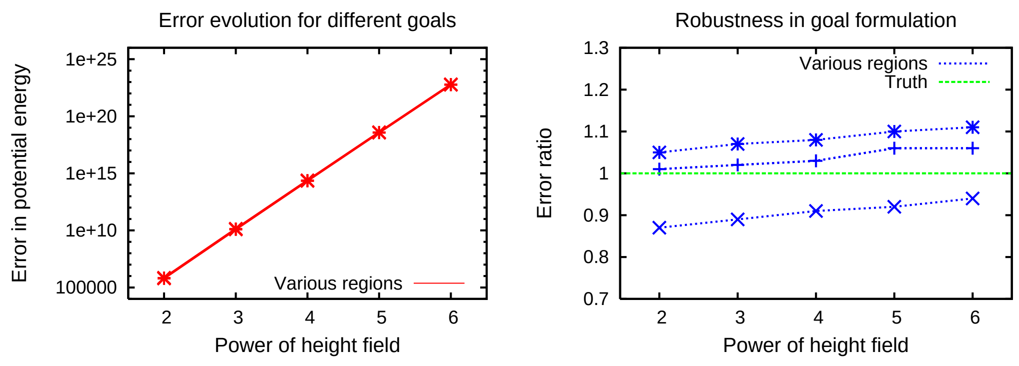


Figure 3.5: TC1: absolute error (left) and ratio of error estimates and true errors (right) for different regions (cells 9, 10, 11 from set S_1). The goal error estimates use the local error estimator F^1_Δ. The absolute error increases drastically for higher powers of the height field, the quality of the estimate stays nearly constant.

local error estimator appears to be the least volatile. The cells that fall out of the 50% improvement region are identical to the bad performers identified from Table 3.3.

Goal formulation robustness

To test the robustness with respect to choice of functional, we define the same fixed integration time of six hours as in the section before. We calculate different goals J_β as regional integrals over different powers of the height field $J_\beta \sim \int gh^\beta$ for $\beta = 2, ..., 6$. Increasing β leads to fast increasing absolute errors in the output goals, with varying numerical values of from order 10^6 to 10^{24}. Our error method is robust against this kind of changes in goal formulation (plotted for cells 9, 10 and 11 from Set S_1 in Figure 3.5). All regions show only small changes in the ratio between estimated and true errors, within a range of 85% to 110%. Results are shown for three arbitrary regions and the F^1_Δ local error estimator. The results are similar for all three types of estimators and all regions we looked at.

Corrected estimates and higher resolution approximations

We test if our error estimates can be used as error correction when compared with higher resolution goal approximations. We define a fixed integration time of six hours and compare the solution at resolution $\Delta 1$ with two solutions at higher resolutions ($\Delta 2$ and $\Delta 3$, see Table 3.1). We use the F^2_Δ local error estimator to correct the goal approximation (3.19) for cell 9 of set S_1. We show in Figure 3.6 the best approximation of the truth as constant green line. The corrected goal is of similar quality as the uncorrected goal approximation from a two times refined resolution.

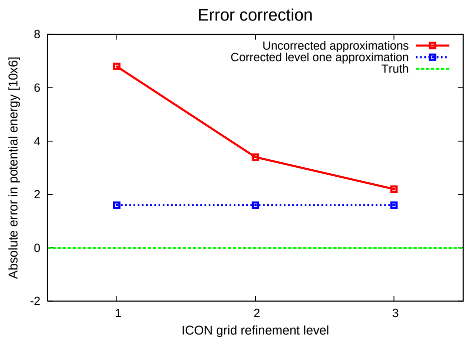


Figure 3.6: TC1: absolute error in potential energy against resolution for cell 9 of set S_1. Red: uncorrected regional potential energy from approximated solutions. Blue: corrected $\Delta 1$ approximation, using F^2_Δ local error estimator. Reference truth is the $\Delta 6$ reference solution.

Time evolution of goal error

To test the robustness with respect to integration time, we fix the region and use again the lowest resolution $\Delta 1$ for different integration times. Looking at the timeseries of estimated and true error for a single region (cell 9 of set S_1) is instructive because it allows us to see if the model evolution is captured correctly. We compare the estimated errors with a F^3_Δ estimator to the true evolution of the pointwise error for 24 hours (Figure 3.7). For each data point in time, the respective adjoint backward problem is solved. The error estimation works well until the model error reaches its maximum value after around 21 hours, but has difficulties following the decline of model error to zero. The estimated error decreases and increases with roughly the same periodicity as the true error but the decrease is not strong enough. This behavior is typical for all cells that we have tested. The general characteristics of TC1 are apparently more important than initial regional flow states to determine how long our error estimates are useful. The F^1_Δ and F^2_Δ estimators are not plotted because they are not able to correctly estimate a decrease in error.

3.4.2 Zonal Flow against a Mountain (TC2)

We now apply our error estimation algorithm to a flow that has a distinctive local feature. Our second test case TC2 was proposed in (Williamson and Drake 1992); a zonal flow hits a Gaussian mountain, and the evolution of the perturbed flow is investigated. For our learning algorithm, the immediate change of the flow after initialization is a challenging property of this test case. We determine a reference solution by performing an integration of ICOSWM at resolution $\Delta 6$ ($\approx 35\,\mathrm{km}$) with a time step of $50s$ because we do not have an analytical solution. The evolving pattern of TC2 in the meridional

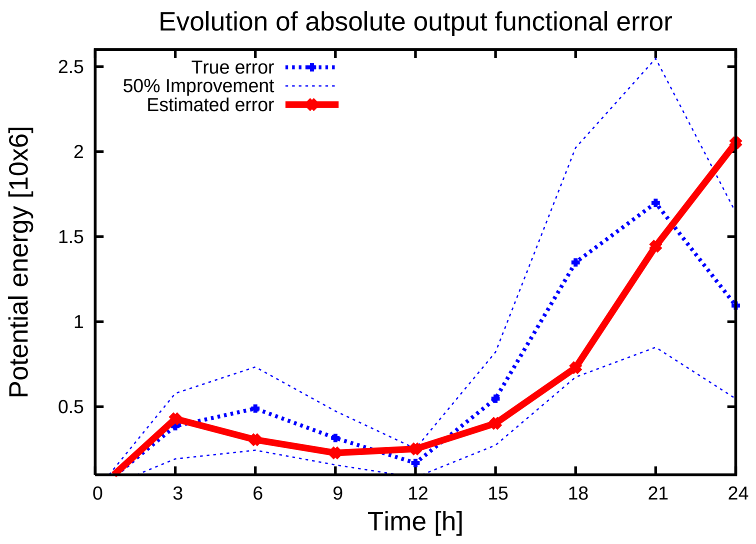


Figure 3.7: TC1: evolution of true error and error estimate for 24 hours in cell 9 of set S_1. Blue: true goal error, calculated with the analytical solution. Red: error estimate, using local error estimator F^3_Δ. Thin blue: 50 % improvement regime.

velocity can be seen in the right panel of Figure 3.1. We do not prove nor claim convergence at resolution Δ6 but instead try to show that our method can estimate the error between solutions derived from two strongly different resolutions.

We are interested in the behavior of potential energy at the downstream side of the mountain perturbation, in contrast to TC1, where we looked at randomly distributed areas across the zonally symmetric Earth. We therefore choose a set S_2 of nine regions that lie downstream from the mountain for our experiments, see Table 3.2. The evolution of potential energy in several of these regions is shown in the right panel of Figure 3.2. The initial zonal flow is perturbed, and this perturbation is transported throughout the flow. The effect of the disturbance sets in after a time that depends on the distance between the region where potential energy is calculated and the mountain, see the right part of Figure 3.2. This makes the learning process demanding: we do not expect a steady zonal flow to show identical behavior to a newly excited gravity wave. To test the dependence of the algorithm on the flow state throughout the learning period, we introduce a second set of regions S_3 that incorporates four regions that are directly connected to the mountain and exhibit the perturbation immediately in the first time steps, i.e., during the learning period. The fact that we gauge only during the first very few time steps requires that the flow during these first time steps should be "typical" for the forecast period.

TC2: Improvement Percentage for cells in set S_2												
	Gauging	LEE	Cell ID $\longrightarrow$									
	Set $\downarrow$		1	2	3	4	5	6	7	8	9	$\oslash$
ω_2	S_2	F^2_Δ	4	22	-200	47	78	-551	-393	-351	-108	-161
ω_{3a}	S_3	F^2_Δ	3	-100	0	68	74	95	95	94	5	38
ω_{3b}	S_3	F^3_Δ	83	64	-112	62	32	61	74	72	92	57

Table 3.4: Improvement percentage for different cells of set S_2 and TC2. Different rows represent different learned parameters $\tilde{\omega}$. Values between 0 and 100 mean an improvement (100 = we completely correct the error), negative values mean a deterioration of the results (-100 = the corrected absolute goal error is twice as big as the original absolute error). $\oslash$ shows the average improvement. Gauging set: the set of cells that is used to gauge the local error estimator. LEE: the used local error estimator.

The topographic representation of the mountain is very crude at the lowest resolution $\Delta 1$. This means that the starting height fields are already differing quite significantly between different resolutions. We choose as a standard low resolution $\Delta 2$ for these experiments.

Learning and spatial robustness

The first experiment aims at learning the parameter set of the local error estimators and use them for a 12 hour forecast of potential energy. We use the spatial estimator F^2_Δ (3.26) to obtain the scaling factor ω_2 by applying the learning algorithm to the set of grid cells S_2. If we use this averaged weight to estimate the errors after 12 hours for all cells in S_2 the quality of error estimates differ strongly (Table 3.4). Slight improvements in four cells are contrasted with a strong deterioration of the results in five different cells, resulting in an averaged deterioration of the results of 161%. We use set S_3, where the perturbation starts directly and is therefore present during the complete learning period and obtain a different scaling factor ω_{3a}. This scaling factor leads to improved results, as can also be seen in Table 3.4. Here we can see significant improvements in 5 out of 9 cells, small changes in 3 and deterioration in one, resulting in an overall average improvement of 38%. We use the temporal estimator F^3_Δ (3.27) to obtain the scaling factor ω_{3b}, again for the set S_3. The results are promising, with 8 out of 9 improved goal approximations and an average improvement of 57%, but the results are not as consistent as for TC1. The strong dependency on the learning region is a result of the original steady flow being replaced by an instantaneous perturbation.

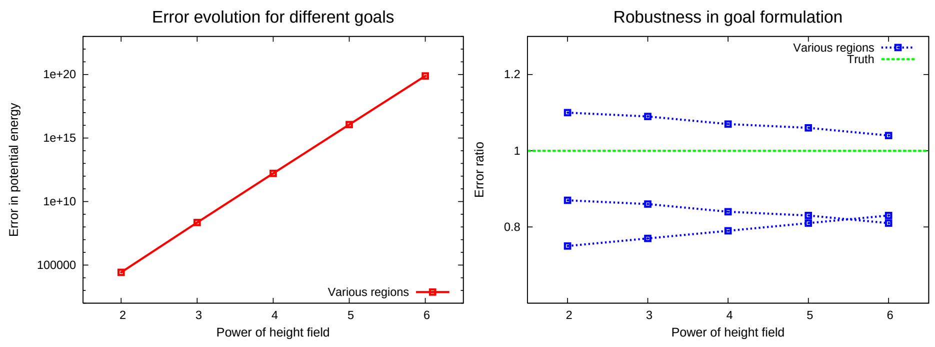


Figure 3.8: TC2: absolute error (left) and ratio of error estimates and true errors (right) for different regions (cells 1, 2 and 3 of set S_3). The absolute error increases drastically for higher β, the quality of the estimate stays nearly constant. The goal error estimates use the local error estimator F^3_Δ.

Goal formulation robustness

To test the robustness with respect to different goals, we repeat the goal formulation experiment of Section 3.4.1. The results in Figure 3.8 look similar to Figure 3.5, albeit slightly worse because the general quality of the error estimates is lower. The three regions shown in Figure 3.8 are the three cells of set S_3 of Table 3.4 and we use ω_{3b} to obtain these results. The goal formulation dependency is again taken care of automatically by the adjoint sensitivities. The weights do not depend on the choice of goal, at least not for slight changes of goal formulation.

Corrected estimates and higher resolution approximations

We test the quality of error correction with regard to resolution by repeating the same experiment as for TC1 in Section 3.4.1. The results in Figure 3.9 show that we can improve our results approximately by one level of grid refinement. This is a promising result but inferior compared to the equivalent improvement of two levels of grid refinement that could be achieved for TC1.

The scaling factor and its flow-type dependency

We use the two learned parameters ω_{3a} and ω_{3b} and see how well our method estimates the errors for single time steps between 3 hours and 24 hours for the three cells of set S_3. Each error estimate needs the solution of one separate adjoint problem. We can see from Table 3.5 that the improvement of the goal approximations is irregular. While the flow is still steady, the error estimates are degrading the original goal approximations. Still, in absolute numbers this is not too surprising because the absolute errors are

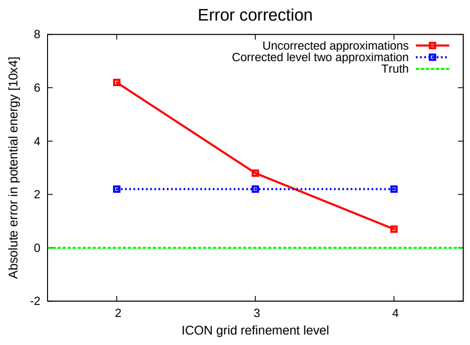


Figure 3.9: TC2: absolute error in potential energy against resolution for cell 1 of set S_3. Red: uncorrected regional potential energy from approximated solutions. Blue: corrected $\Delta 1$ approximation, using the F^2_Δ local error estimator. Reference truth is the $\Delta 6$ reference solution.

very low while the original zonal flow is intact. As soon as the cells experience the perturbation the error increases, and the quality of the error estimates improves as well. This is in line with expectations because we have learned ω_{3a} and ω_{3b} in regions where the perturbed initial wave dominated the flow for the first time steps. The "learned“ flow state, the gravity wave, hits cell 1 approximately during the first hour, cell 2 and cell 3 after approximately 10h. This explains the constant improvement with both scaling factors for cell 1and the different behavior for cells 2 and 3.

3.4.3 Discussion

We have evaluated our algorithm proposal for two test cases. If we combine the results from both test cases we see that the method works for certain flow regimes, given a successful learning of $\tilde{\omega}$. It appears, however, that our algorithm struggles with changing flow regimes. For TC1, we estimate a time window of 24 hours within which we can improve goal approximations with a learned $\tilde{\omega}$. For TC2, we conclude that there is no clear time window because of the very different timings of the initial perturbation in different cells. We can only estimate the error for the time frame during which the flow state is similar to the flow state during the learning period. We estimate this time frame to be around 12 hours for TC2.

The results from TC2 highlight an important point of our algorithm. It is paramount for the learning algorithm to learn within a representative flow regime, i.e., the flow type during the first time steps should be similar to the flow regime throughout the forecast period. This is especially important if topography plays a role. If goals are derived from parts of the solution that are heavily influenced by topography we should

TC2: Improvement Percentage for cells in set S_3							
		Integration time $\longrightarrow$					
Cell ID $\downarrow$	ω	6h	9h	12h	15h	18h	24h
1	ω_{3a}	90	42	95	64	42	76
2	ω_{3a}	-12	-30	95	54	30	64
3	ω_{3a}	-442	-123	94	38	95	-5
1	ω_{3b}	61	56	61	44	66	66
2	ω_{3b}	-75	-45	74	94	75	94
3	ω_{3b}	-87	-90	72	97	10	56

Table 3.5: Improvement percentage for the three cells of set S_3 and different integration times for TC2. Values between 0 and 100 mean an improvement (100 = we completely correct the error), negative values mean a deterioration of the results (-100 = the absolute corrected goal error is twice as big as the absolute original error).

implement the learning algorithm in the same region. From the comparison between both experiments and the importance of the learning region we can deduce a flow-regime dependency of the learned scaling factor $\tilde{\omega}$. This is exemplified in our test case because an originally steady zonal flow is perturbed by a gravity wave and then establishes a third, stable flow regime. For realistic test cases this means that our method needs to be run in a "time window mode", comparable to data assimilation: the scaling factor ω should be re-determined to a new flow-specific value $\tilde{\omega}$ every time we start a new time window. The comparison of specific scaling factors $\tilde{\omega}$ of different time windows can also provide additional information about the flow regime at hand; the usefulness of this information is subject to further research.

3.5 Conclusion & Outlook

We have introduced a novel goal error estimation algorithm and have evaluated its application to a discrete model of the shallow water equations and two test cases. For a global unsteady flow (TC1), our evaluation has shown robustness of the goal error estimate with respect to resolution, integration time and goal specification (functional form and region). For TC1, it is possible to improve the quality of a goal approximation from a low-resolution solution to the quality of a goal derived from a higher-resolution solution (with 16 times more spatial degrees of freedom and two times more temporal degrees of freedom). We estimate the time span over which the error can be reduced consistently to be around a day for our current set of local error estimators. We have also shown for TC1 that our learning algorithm allows us to reach 95% of the theoretically possible improvement of our method, i.e., the gauged error estimators lead to an improvement of 71% compared to a maximum of 75%. For TC2, our evaluation has also shown robustness of the goal error estimate with respect to goal specification and resolution, although the results are not as good as for TC1. The average improvement rate is 38% to 57%, depending on the type of estimator chosen. We have also shown for TC2 that it is possible to improve the quality of a goal approximation from a low-resolution solution to the quality of a goal derived from a higher-resolution solution (with 4 times more spatial degrees of freedom and 1.5 times more temporal degrees of freedom).

The key idea of our error estimation algorithm, namely to train the error estimation algorithm on short time scales, can be interpreted in a more general context: how long can "initial information" (here: the flow state in the learning period) be used for general forecasting purposes (here: error correction)? What happens if the near-initial flow state is not representative of the prediction flow state? Answering these questions needs a more elaborate testbed and possible lines of future research are clear: First, the simple smoothness measures are general and can be refined, extended and combined in the future to obtain improved goal error estimates. Second, the learning algorithm itself is an integral point of our method that should be refined in the future. We believe that it is necessary to implement a truly automatic flow-regime-dependent learning mechanism for our local error estimators that determines at the beginning of a simulation the correct weights. We also think that all local information at grid cell level should be used to make the learned parameters more robust. This means that the local solution differences should be used to learn the correct properties of local error estimators and not only the resulting differences in goals. Third, a logical extension of our work is to also analyze velocity and vorticity-derived goals and to investigate if this forces us to include velocity-based functionals or not. The two test cases and robustness tests in this work do not conclusively answer the question of general applicability. Instead, they are meant as a proof-of-concept for our new concept of goal-oriented error estimation through learning.

Our method has the advantage of avoiding the difficult process of analyzing the specific numerical model and of manually learning where the model locally produces errors. It is reasonable to assume that the use of information on the underlying PDE and the model discretization within local error estimators can lead to more reliable goal error estimates when compared to our empirical approach. This potential for better error estimation has to be balanced, however, with the inherently increased complexity of these approaches which may prevent actual application in some fields. The structural complexity of General Circulation Models with parameterizations prevents the direct use of classical adjoint error estimation techniques because these parameterizations often do not have an underlying PDE. This is why we believe that our learning approach may lead to a compromise between complexity and accuracy that is suitable for GFD applications. Our goal error estimation method is a first step towards enabling geophysical models to deliver estimates of the discretization error together with each numerical output. [1]

Summary of Chapter 3

- We describe the mathematical framework of classical dual weight error estimation.
- We show how to adapt this framework to enable its application for GFD applications.
- We introduce empirical local error estimators that represent a general functional dependency between local model errors and the solution.
- We present a learning algorithm that determines the specific parameters of these empirical local error estimators for a given model.
- We apply the empirical local error estimators successfully to a shallow water model and two test cases.

[1]This Chapter has been published as Rauser et al. (2011).

Chapter 4

On the Use of Discrete Adjoints for Goal Error Estimation

Goal oriented dual weight error estimation has been used in the context of computational fluid dynamics for several years. The technical adaptation of this method to geophysical models is the subject of this chapter. We use a differentiation-enabled prototype of the NAG Fortran compiler to generate a discrete adjoint version of such a geophysical model that computes the required goal sensitivities. We present numerical results for a shallow water configuration of the Icosahedral Non-Hydrostatic General Circulation Model (ICON) and discuss a special treatment of the underlying linear solver, yielding improved scalability of this approach and a significant reduction in runtime. [1]

4.1 Introduction

During the past decades the needs of society, policy makers and industry have led to the increasing usage of Earth system models (ESM) for forecasting tasks (Meehl et al. 2007). This change from predominantly analytic usage to predictive usage has substantially increased the demands on the modeling community to supply not only physically meaningful answers but also uncertainty estimates for these answers. ESMs incorporate a huge number of different possible error sources. The identification and reduction of these error sources is one of the major challenges on the way to reliable climate predictions / projections. One classical example for the efforts to reduce model error is data assimilation for atmospheric and oceanic models (Wunsch et al. 2009; Kalnay 2003). Data assimilation minimizes the distance between model trajectories and any given set of measurements but it remains unclear how to identify and quantify different

[1]The work for the publication (Rauser et al. 2010) was collaborative by nature. Uwe Naumann and Jan Riehme develop the AD extension of the NAG compiler in Aachen. I have constructed the ICOSWM-AD version. Klaus Leppkes has implemented the direct solver for speed-up of the AD version. The writeup was mostly done by Jan Riehme (4.5) and myself (4.1, 4.2, 4.3, 4.6, 4.7). This chapter has been changed and extended for editorial purposes.

sources of uncertainty.

The total error of numerical models can be separated into two components (Oden and Prudhomme 2002): the "modelling error" as the difference between model description and physical process, and the "approximation error" as the difference between the true model solution and the computational approximation. The problem can be simplified because compared to the underlying large number of discrete prognostic variables usually only a limited number of output variables is useful. These outputs are called goals. We therefore need only to estimate the error of these goals and not the error of all the prognostic fields. This a posteriori error estimation of model goals is a method well known from computational fluid dynamics (CFD) and is called goal oriented dual weight error estimation (Giles et al. 2004; Becker and Rannacher 2002; Babuska and Rheinboldt 1978; Johnson et al. 1995). In this chapter we estimate goal errors for a geophysical fluid dynamics numerical model.

4.2 Goal Oriented Dual Weight Error Analysis

We look at a system defined by a nonlinear evolution equation for a state vector $\mathbf{q}$, an initial condition $\mathbf{q}^0$ and a goal J that is evaluated always at the final time t_{end} on a periodic domain Ω

$$N(\mathbf{q}(\mathbf{x},t)) = 0, \qquad \mathbf{q}(\mathbf{x},t_0) = \mathbf{q}^0, \qquad J = J(\mathbf{q}(\mathbf{x},t_{end})). \tag{4.1}$$

We define the error between the true model goal value J and a numerical approximated goal $J_\triangle$ as

$$\varepsilon := J(\mathbf{q}) - J_\triangle(\mathbf{q}_\triangle), \tag{4.2}$$

with $\mathbf{q}_\triangle$ the numerical approximation of $\mathbf{q}$. The fundamental principle of this approach is to solve for a given output goal of interest J an adjoint system that yields the sensitivities $\mathbf{q}^*$ of the goal $J(\mathbf{q})$ towards changes of the prognostic variables $\mathbf{q}$. These sensitivities are then integrated over the space time domain as weights for a function that indicates the error produced by our model. Literature derivations (e.g. (Giles et al. 2004)) show that Equation (4.2) can be approximated as

$$\varepsilon \approx \left\langle {\mathbf{q}_\triangle^*}^T, \hat{N}_\triangle(\mathbf{q}_\triangle) \right\rangle, \tag{4.3}$$

with $\mathbf{q}^*_\triangle$ the discrete solution of the continuous adjoint problem to Equation (4.1) and $\hat{N}_\triangle(\mathbf{q}_\triangle)$ a residual estimator that is a function of the approximated flow state $\mathbf{q}_\triangle$. This estimator is strongly problem dependent. Classical approaches to construct $\hat{N}_\triangle(\mathbf{q}_\triangle)$ are discretization dependent. We have shown in Chapter 3 how to construct local error estimators $\hat{N}_\triangle(\mathbf{q}_\triangle)$ that are discretization independent.

Level	Cells	Edges	Average length [km]
Δ1	320	480	1115
Δ2	1280	1920	556
Δ3	5120	7680	278
Δ4	20480	30720	139
Δ5	81920	122880	70

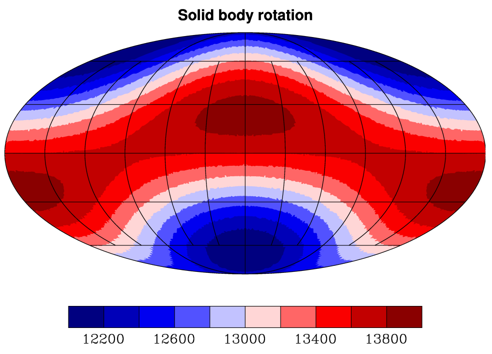


Figure 4.1: Left: Table of ICON grid properties. "Level" equals the number of refinement steps. Right: The initial surface height field is plotted [m].

Equation (4.3) is the scalar product of two components. Therefore, two steps are necessary to adapt this method to geophysical problems: First, we want to obtain an approximation of $\mathbf{q}^*_\Delta$ automatically. Second, we want to construct an estimator that is cheap to compute and easy to implement for arbitrary discretizations. We focus in this chapter on an efficient way to obtain the approximation of $\mathbf{q}^*_\Delta$ with automatic differentiation tools, especially the use of discrete adjoints. Details on the properties of the second component can be found in Chapter 3.

4.3 The Primal Problem

As a prototype application for our error estimation method we choose the shallow water equations (SWE) on a sphere. The SWE share significant properties of the global atmospheric and oceanic fluid system with more complex descriptions and are able to simulate large scale flows. The following equations are the vector invariant form of the shallow water equations on the sphere

$$\begin{aligned}\frac{\partial \mathbf{v}}{\partial t} &= (\xi + f)\mathbf{k} \times \mathbf{v} - \nabla(gh + \frac{1}{2}|\mathbf{v}|^2) \qquad (4.4)\\ \frac{\partial h}{\partial t} &+ \nabla \cdot (h\mathbf{v}) = 0.\end{aligned}$$

Here $\mathbf{v}$ is the horizontal velocity, ξ the vorticity, f the rotational parameter, g the gravitational acceleration and h the height surface elevation. The state vector $\mathbf{q} = (h, \mathbf{v})$ consists of the prognostic fields height and velocity.
An important physical quantity in a flow is the potential energy density gh^2 with h the solution of Equation (4.4). We are interested in the behavior of the potential energy in generic subdomains Ω_0 of the domain Ω, the sphere. The goal is defined as regional potential energy averaged over this subdomain Ω_0 at the end of an integration time

t_{end}

$$J(\mathbf{q}) := J(h(t_{end})) = \frac{g}{A(\Omega_0)} \int_{\Omega_0} h^2(\mathbf{x}, t_{end}) d\mathbf{x}, \tag{4.5}$$

where Ω_0 denotes an arbitrary subdomain of the sphere Ω and $A(\Omega_0)$ denotes the area of Ω_0.

The shallow water equations can simulate a variety of flow regimes. For testing purpose we start with a simple wave like setting. We use the time-dependent solid body rotation test case proposed in example 3 in (Laeuter et al. 2005). Atmospheric values for velocities are used that are comparable with Williamson's test cases (Williamson and Drake 1992). The initial condition can be seen in Figure 4.1. The analytical solution consists of a propagation of a global wave structure westwards, with a periodicity of 24 hours. This periodic flow field implies also a periodic behavior of our goal $J(\mathbf{q})$ from Equation (4.5).
The numerical framework is ICOSWM, a recently developed shallow water model on a triangular grid with C-type staggering on the sphere (Bonaventura and Ringler 2005). ICOSWM uses a hybrid finite volume / finite difference method with a two-level timestepping to approximate the SWE (4.4). For further details see (Giorgetta et al. 2009; Ripodas et al. 2009). ICOSWM calculates the discrete state vector $\mathbf{q}_\Delta = (h_\Delta, \mathbf{v}_\Delta)$ with discrete height field h_Δ in the cell centers of the triangular grid and normal velocities $\mathbf{v}_\Delta$ at the middle points of the triangular edges.
The horizontal grid is derived from the regular icosahedron. The projection of the regular icosahedron on the unit sphere provides a regular grid on the sphere with 20 equilateral spherical triangles, 30 great circle edges, and 12 vertices. Its dual grid is the projection of the regular dodecahedron on the sphere. The Delaunay triangulation then allows to refine each triangle into n^2 smaller triangles by dividing each edge into n sections. For our purposes we use $n = 2$. This procedure may be then repeated ν times, resulting in $20 \times 4^\nu$ triangular cells. The lowest ICON resolution is equivalent to 320 grid cells or two refinements. For this chapter we mainly use this lowest resolution and for comparison the next two refinement levels, see the table in Figure 4.1. It is important to note that the regularity of the spherical triangles of the base grid is lost in the refinement process, though the differences in areas between triangles or lengths between edges remain small. This break in symmetry is obvious in the dual grid that consists of pentagons and hexagons.

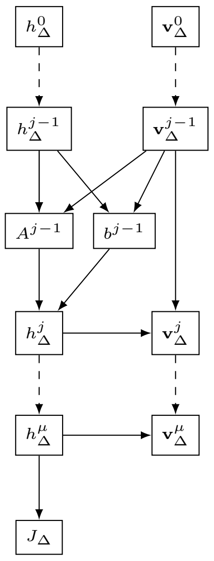


Figure 4.2: Computational Graph

4.4 The Computational Graph

The semi-implicit two level discretization of the shallow water equations on the spherical icosahedral grid is described in detail in (Bonaventura and Ringler 2005). The details of goal error estimation with empirical estimators is described in (Rauser et al. 2011). The complete computational problem combines solving the shallow water equations and estimating the error for output goals. A summary of the whole algorithm is the following:

(1) Do for μ time steps ($j = 1, \ldots, \mu$):
  (a) Calculate new surface height h^j_Δ by solving linear free surface equation
  $A^{j-1}(h^{j-1}_\Delta, \mathbf{v}^{j-1}_\Delta) \cdot h^j_\Delta = b^{j-1}(h^{j-1}_\Delta, \mathbf{v}^{j-1}_\Delta)$ (implicit step)
  (b) Update velocity $\mathbf{v}^j_\Delta = \mathbf{v}^j_\Delta(h^j_\Delta, \mathbf{v}^{j-1}_\Delta)$ (explicit step)
(2) Evaluate discrete goal $J_\Delta(\mathbf{q}_\Delta) = J_\Delta(h^\mu_\Delta)$, with $h^\mu_\Delta = h^\mu_\Delta(h^0_\Delta, \mathbf{v}^0_\Delta)$.
(3) Perform error estimation using discrete adjoint of J_Δ.

The matrix A^{j-1} depends on the height field and the velocities of the last time step. The right hand side b^{j-1} is a vector that also depends on the old height field and the old velocities.
The corresponding computational graph is shown in Figure 4.2.

4.5 The Dual Problem

As an alternative to the derivation of the dual / adjoint system followed by its approximate numerical solution to get $\mathbf{q}^*_\Delta$ we apply the adjoint (or reverse) mode of algorithmic differentiation (AD) (Griewank 2000) to the given numerical solution scheme for the primal problem defined in Equation (4.4). Adjoint mode AD yields the gradient ∇J_Δ of the discrete goal J_Δ at a typically small constant factor of the computational cost $\mathrm{Cost}(J_\Delta)$ of a single evaluation of J_Δ. Forward sensitivities computed by the tangent-linear (or forward) mode of AD or approximations thereof based on finite difference quotients yield an often infeasible computational cost of $O(n) \cdot \mathrm{Cost}(J_\Delta)$.

Let $N(\mathbf{q}(\mathbf{x}, t))$ be solved by an iterative algorithm F as the semi-implicit scheme that is sketched in Section 4.4 for a given start vector $\mathbf{q}_\Delta^0$ and let $\mathbf{q}_\Delta^\mu = F(\mathbf{q}_\Delta^0)$ denote the state vector after μ time steps. Conceptionally, adjoint mode AD runs the primal code

$$\begin{aligned} \mathbf{q}_\Delta^\mu &= F(\mathbf{q}_\Delta^0) \\ J_\Delta(\mathbf{q}_\Delta) &= J_\Delta(\mathbf{q}_\Delta^\mu) \end{aligned}$$

in order to memorize intermediate quantities required for the evaluation of products of the transposed Jacobian $\nabla F^T = \nabla F(\mathbf{q}_\Delta^0)^T$ with a vector in $I\!R^n$ followed by the adjoint code

$$\begin{aligned} \bar{\mathbf{q}}_\Delta^\mu &= \nabla J_\Delta(\mathbf{q}_\Delta^\mu)^T \cdot \bar{J}_\Delta \\ \bar{\mathbf{q}}_\Delta^0 &= \nabla F(\mathbf{q}_\Delta^0)^T \cdot \bar{\mathbf{q}}_\Delta^\mu . \end{aligned}$$

Initializing $\bar{J}_\Delta = 1$ yields the required gradient $\bar{\mathbf{q}}_\Delta^0 = \nabla J_\Delta(\mathbf{q}_\Delta^0)$. The general relationship between the discrete adjoints $\bar{\mathbf{q}}_\Delta^0$ and the discrete solution $\mathbf{q}_\Delta^*$ of the continuous adjoint problem for Equation (4.1) is the subject of ongoing investigations. A new prototype of the NAG Fortran compiler is currently being developed to enable the mostly automatic semantical transformation of numerical input code into adjoint code (Naumann and Riehme 2005). The derivative code compiler has been applied successfully to ICOSWM in order to provide the required discrete adjoints. For a given implementation (in Fortran) of

$$F : I\!R^n \to I\!R^n, \quad \mathbf{q}_\Delta^\mu = F(\mathbf{q}_\Delta^0),$$

the compiler produces code for the evaluation of $\bar{\mathbf{q}}_\Delta^0 = \nabla F(\mathbf{q}_\Delta^0)^T \cdot \bar{\mathbf{q}}_\Delta^\mu$.

While a detailed discussion of adjoint code generation is beyond the scope of this chapter we still need to take a closer look at some of the underlying principles. The given implementation of F is assumed to decompose into a *single assignment code* (SAC) at every point of interest as follows:

$$\begin{aligned} &\text{for } j = n+1, \ldots, n+p+m \\ &\quad v_j = \varphi_j(v_i)_{i \prec j} \quad , \end{aligned} \tag{4.6}$$

Forward code	Adjoint code
$v_3 = v_1 \cdot v_2$	$\bar{v}_3 = -\sin(v_3) \cdot \bar{v}_4$
$v_4 = \cos(v_3)$	$\bar{v}_2 = v_1 \cdot \bar{v}_3$
	$\bar{v}_1 = v_2 \cdot \bar{v}_3$

Figure 4.3: A simple example for automatic differentiation of single assignment codes

where $i \prec j$ denotes a direct dependence of v_j on v_i. The result of each elemental function φ_j is assigned to a unique auxiliary variable v_j. The n *independent inputs* $x_i = v_i$, for $i = 1, \ldots, n$, are mapped onto m *dependent outputs* $y_j = v_{n+p+j}$, for $j = 1, \ldots, m$, and involve the computation of the values of p *intermediate variables* v_k, for $k = n+1, \ldots, n+p$.

For given adjoints of the dependent and independent variables, reverse mode AD propagates adjoints backward through the SAC as follows:

$$\begin{aligned} \text{for } j =& n+p+m, \ldots, n+1 \ \text{ and } \ i \prec j \\ \bar{v}_i =& \bar{v}_i + \bar{v}_j \cdot \frac{\partial \varphi_j}{\partial v_i}(v_i)_{i \prec j} \quad . \end{aligned} \tag{4.7}$$

The variables $\bar{v}_j$ are assumed to be initialized to $\bar{y}_j$ for $j = n+p+1, \ldots, n+p+m$ and to zero for $j = 1, \ldots, n+p$. A forward evaluation of the SAC is performed to compute all intermediate variables whose values are required for the adjoint propagation in reverse order. The elemental functions in the SAC are processed in reverse order in the second part of Equation (4.7). See Figure 4.3 for a simple example. The two entries of the gradient are computed by setting $\bar{v}_4 = 1$. The correctness of this approach follows immediately from the associativity of the chain rule of differential calculus.

4.5.1 The Differentiation-Enabled NAG Fortran Compiler

The differentiation-enabled NAG Fortran compiler (from now on referred to as "the compiler") combines a two stage semantical transformation with a set of runtime support libraries in a hybrid approach to AD that blends source transformation capabilities and overloading techniques. The robustness of the runtime solution based on overloading is supported by potential performance gains to be expected from a source code transformation algorithm. Without loss of generality, we present the discrete adjoint in the light of overloading rather than pure source transformation. Our current research prototype compiler cannot handle the full ICOSWM code in source transformation mode. Nevertheless we are able to achieve very good runtime results that are shown in Section 4.6.

i	Tape (forward eval.)				Variables (forward)		Tape (reverse)
	opc	a1	a2	val	x	y	adj
1	IDP	0	0	$v_1 = x\%val$	$\{v_1, 1\}$	$\{v_2, 0\}$	$-\sin(v_3) * v_2$
2	IDP	0	0	$v_2 = y\%val$	$\{v_1, 1\}$	$\{v_2, 2\}$	$-\sin(v_3) * v_1$
3	MUL	1	2	$v_3 = v_1 * v_2$	$\{v_1, 1\}$	$\{v_2, 2\}$	$-\sin(v_3)$
4	COS	3	0	$v_4 = \cos(v_3)$	$\{v_1, 1\}$	$\{v_4, 4\}$	1

Table 4.1: Tape generated for code in Figures 4.4 and 4.5

```
SUBROUTINE F( x, y )           ! 1
   DOUBLE PRECISION  :: x, y   ! 2
   y = cos( x * y )            ! 3
END SUBROUTINE
```

Figure 4.4: Source for SAC in Figure 4.3

Every support library (`compad_module`) defines an active datatype (`compad_type`) with corresponding overloaded arithmetic operators and intrinsic functions (for example, forward and reverse mode AD, second order derivatives by forward over reverse). After selecting a specific `compad_module`, the first stage of AD-related semantical transformation changes the datatype of all floating-point variables into `compad_type`. Any operation with arguments of `compad_type` are resolved by the compiler to operators from the selected `compad_module`. In the optional second stage of semantical transformation the compiler modifies the internal representation by inserting code that works directly on the components of `compad_type`. Thereby the overhead of calling overloaded operators and intrinsics from the `compad_module` can be avoided.

Discrete adjoints for ICOSWM are obtained using a support library that records every arithmetic operation on a *tape* during the *augmented forward evaluation* of F. Adjoints are propagated during a subsequent interpretative *reverse evaluation* of the tape. Each tape entry represent one unique auxiliary variable v_j in Equation (4.6). A tape entry (see columns 2–5 in Table 4.1) contains an operation code (opc), tape index(es) of the argument(s), the value of the corresponding auxiliary variable v_j, and its adjoints (see last column in Table 4.1, initially set to `0`). The data type `compad_type` consists of the value and the index of the corresponding auxiliary variable / tape entry (see columns 6–7 in Table 4.1). We apply the compiler to the simple Fortran source code in Figure 4.4, that corresponds to the SAC in Figure 4.3, for illustration. Figure 4.5 shows the hand-written driver program required to compute sensitivities

```
PROGRAM TEST_TAPE
  USE compad_module              ! 1
  TYPE(COMPAD_TYPE)  :: x, y     ! 2
  DOUBLE PRECISION   :: grad(2)  ! 3
  INTEGER(TAPE_IKND) :: idy      ! 4
  x = 1.3D0; y = 0.4D0           ! 5
  CALL TAPE_INIT( 100 )          ! 6
  CALL TAPE_TURN_ON              ! 7
  CALL INDEPENDENT( x )          ! 8
  CALL INDEPENDENT( y, idy )     ! 9
  CALL F( x, y )                 !10
  CALL TAPE_TURN_OFF             !11
  CALL SEED( y, 1.D0 )           !12
  CALL TAPE_INTERPRETER          !13
  grad(1) = DERIV(x)             !14
  grad(2) = DERIV_INDEX(idy)     !15
END PROGRAM TEST_TAPE
```

Figure 4.5: Driver program for code in Figure 4.4

by the tape based discrete adjoint code generated by the compiler. The driver includes `compad_module` (line 1), and declares independent and dependent variables as `compad_type` (line 2). Memory for storing the sensitivities is allocated in line 3. After initializing the tape environment (line 6) the beginning of the computation to be recorded on the tape is marked (line 7). Both independent variables are recorded (lines 8–9). The tape indexes are stored in the corresponding `compad_type` data structures (see rows 1 − 2 in Table 4.1). All adjoint values are initialized to `0`. The tape index of `y` as an independent input needs to be stored explicitly (see declaration and use of `idy` in lines 4 and 9, respectively) as `y` is overwritten by calling the adjoined version of `F` (line 3, Figure 4.4). The value of `idy` is used to access the correct tape entry when retrieving the corresponding gradient entry in line 15.

The augmented forward evaluation of the code in Figure 4.4 is performed in line 10 of the driver. Two new tape entries ($v_3 = x * y$, $v_4 = \cos(v_3)$) (see rows 3 and 4 in Table 4.1) are created. The overloaded assignment of the result (v_4) to `y` stores the tape index `4` in the `compad_type` data structure associated with `y` (row 4, column 7 in Table 4.1). The end of the augmented forward evaluation is marked in line 11 of the driver.
Following the initialization of the adjoint of the dependent variable `y` in line 12 (commonly referred to as *seeding*; see also row 4, last column in Table 4.1), the reverse evaluation (interpretation of the tape) is started in line 13. Three steps are performed

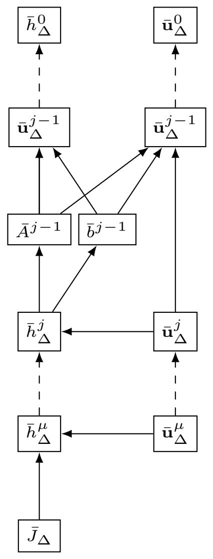

$\bar{h}^0_\Delta$
$\bar{\mathbf{u}}^0_\Delta$
$\bar{\mathbf{u}}^{j-1}_\Delta$
$\bar{\mathbf{u}}^{j-1}_\Delta$
$\bar{A}^{j-1}$
$\bar{b}^{j-1}$
$\bar{h}^j_\Delta$
$\bar{\mathbf{u}}^j_\Delta$
$\bar{h}^\mu_\Delta$
$\bar{\mathbf{u}}^\mu_\Delta$
$\bar{J}_\Delta$

Figure 4.6: Adjoint Computational Graph

yielding the third, second, and first entries in the last row of Table 4.1. Finally, the gradient is *harvested* from the accumulated adjoints of the independent variables (lines 14 and 15) utilizing the tape index `idy` stored for the overwritten instance of `y`.

4.5.2 The Adjoint Linear Solver

Figure 4.6 shows the adjoint computational graph computing $\bar{h}^0_\Delta$ and $\bar{\mathbf{v}}^0_\Delta$ for given $\mathbf{v}^0_\Delta$, h^0_Δ, $\bar{J}_\Delta$ and $\bar{\mathbf{v}}^\mu_\Delta$ based on information stored during the augmented forward evaluation of $J_\Delta(h^\mu_\Delta(h^0_\Delta, \mathbf{v}^0_\Delta))$ (in our case, the tape).

The semi-implicit method involves the solution of the linear system

$$A^{j-1} \cdot h^j_\Delta = b^{j-1} \tag{4.8}$$

for $j = 1, \ldots, \mu$. Both the adjoints $\bar{A}^{j-1}$ of A^{j-1} and $\bar{b}^{j-1}$ of b^{j-1} are functions of the adjoint $\bar{h}^j_\Delta$ of h^j_Δ. A black-box differentiation by the compiler would record all operations performed by the linear solver on the tape followed by an interpretative reverse propagation as outlined in Section 4.5.1.

Alternatively, the following algebraic manipulations of the the linear system (4.8) yield the adjoints $\bar{A}^{j-1}(\bar{h}^j_\Delta)$ and $\bar{b}^{j-1}(\bar{h}^j_\Delta)$ at a significantly lower computational cost. Partial differentiation of Equation (4.8) with respect to A^{j-1} yields

$$A^{j-1} \cdot \frac{\partial h^j_\Delta}{\partial A^{j-1}} + \frac{\partial A^{j-1}}{\partial A^{j-1}} \cdot h^j_\Delta = A^{j-1} \cdot \frac{\partial h^j_\Delta}{\partial A^{j-1}} + h^j_\Delta = \frac{\partial b^{j-1}}{\partial A^{j-1}} = 0$$

and hence

$$A^{j-1} \cdot \frac{\partial h^j_\Delta}{\partial A^{j-1}} = -h^j_\Delta .$$

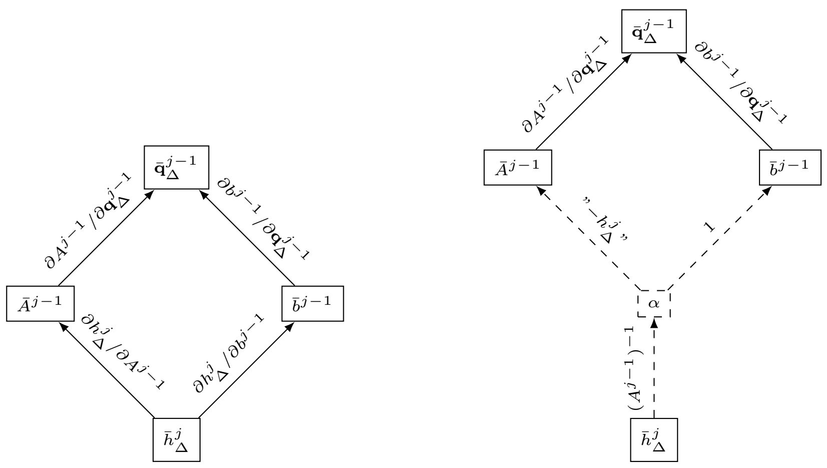
The corresponding discrete adjoint becomes

$$(\bar{h}^j_\Delta)^T \cdot \frac{\partial h^j_\Delta}{\partial A^{j-1}} = -(\bar{h}^j_\Delta)^T \cdot (A^{j-1})^{-1} \cdot h^j_\Delta \quad .$$

Similarly, partial differentiation of Equation (4.8) with respect to b^{j-1} leads to

$$A^{j-1} \cdot \frac{\partial h^j_\Delta}{\partial b^{j-1}} = \frac{\partial b^{j-1}}{\partial b^{j-1}} \qquad \text{and hence} \qquad (\bar{h}^j_\Delta)^T \cdot \frac{\partial h^j_\Delta}{\partial b^{j-1}} = (\bar{h}^j_\Delta)^T \cdot (A^{j-1})^{-1} \quad .$$

The solution h^j_Δ of the linear system (4.8) is computed passively during the augmented forward execution by a direct solver. The resulting LU or QR decomposition of A^{j-1} is reused during the reverse execution yielding a computational cost of $O(n^2)$ for the adjoint as opposed to $O(n^3)$ if taking the black-box approach. With $\alpha := (\bar{h}^j_\Delta)^T \cdot (A^{j-1})^{-1}$ we get $\bar{A}^{j-1}(k_1, k_2) = -\alpha(k_1) \cdot h^j_\Delta(k_2)$ for each (nonzero) entry $A(k_1, k_2)$ of A. Similarly, $\bar{b}^{j-1}(k_1) = \alpha(k_1)$. A graphical illustration is shown in Figure 4.7.


$\bar{\mathbf{q}}^{j-1}_\Delta$
$\partial A^{j-1}/\partial \mathbf{q}^{j-1}_\Delta$
$\partial b^{j-1}/\partial \mathbf{q}^{j-1}_\Delta$
$\bar{A}^{j-1}$
$\bar{b}^{j-1}$
$\partial h^j_\Delta/\partial A^{j-1}$
$\partial h^j_\Delta/\partial b^{j-1}$
$\bar{h}^j_\Delta$
"$-h^j_\Delta$"
1
α
$(A^{j-1})^{-1}$

(a) (b)

Figure 4.7: Adjoint Linear Solver: All arithmetic operations performed by the linear solver are recorded on the tape during the augmented forward evaluation to be used by the subsequent reverse propagation of the adjoints in (a). Dashed lines in (b) mark the tapeless computation of the adjoints $\bar{A}^{j-1}$ and $\bar{b}^{j-1}$ based on a passively derived decomposition of A^{j-1}.

4.6 Results

We have shown in the previous section how to efficiently obtain an approximation to the goal sensitivities needed for Equation (4.3). To estimate the error of the goal defined in

Equation (4.5) we introduce now simple empirical residual error estimators into Equation (4.3). They depend only on flow field information and not on explicit information about the used discretization. The information about the model discretization comes into play via the solution of the adjoint model. We perform different robustness tests with respect to region and functional with a fixed integration time of 6h. To avoid a possible influence of the topography we choose a set of grid cells that are part of a zonal band parallel to the equator.

In the left panel of Figure 4.8 it can be seen that our residual estimator works sufficiently well to estimate the goal errors for the lowest resolution of our model. The estimate is higher than 50% of the true error for most regions. This quality of the error estimates allows us to use them as an error correction term to the original goal approximation, following again (Giles et al. 2004). The right panel of Figure 4.8 highlights that corrected low-resolution goals can be of similar quality as goals derived from higher resolution runs. For the lowest resolution we correct at least half of the error, for higher resolution we approach 100%. The method is also extremely robust versus modifications of the goal functional as can be seen in Figure 4.9. The left plot shows that the numerical values of the output goal vary from order 10^6 to 10^{24} for different powers of the potential energy density. At the same time, the error estimates scale well: the ratio between estimated error and true error stays close to one as can be seen

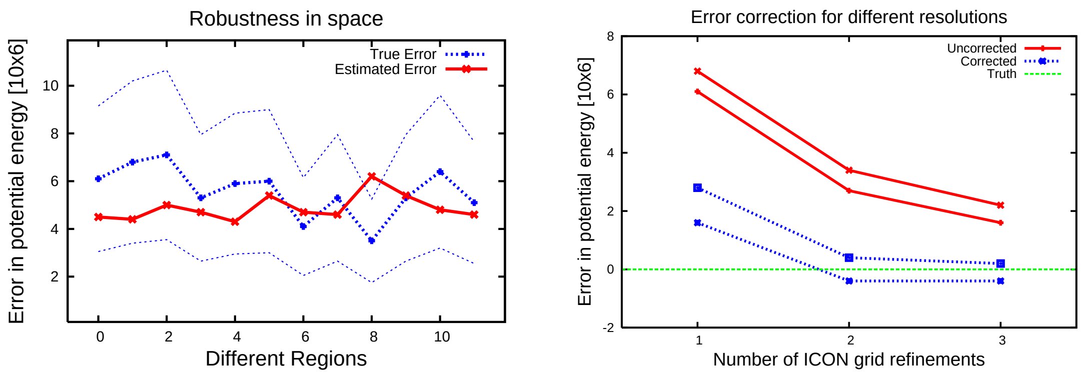


Figure 4.8: Left: On the x-axis different regions are plotted, sorted by the longitude of their cell center. On the y-axis the error in regional potential energy is plotted. The thick solid line represents the true error. The red line represents the estimated error. All estimates that lie between the two thin blue lines represent an improvement of the goal estimate by at least 50 percent. Right: On the x-axis the resolution is plotted. On the y-axis the errors in regional potential energy are plotted. The best approximation of the truth is the green line (no error). The red lines show calculated goals. The blue lines show corrected goals.

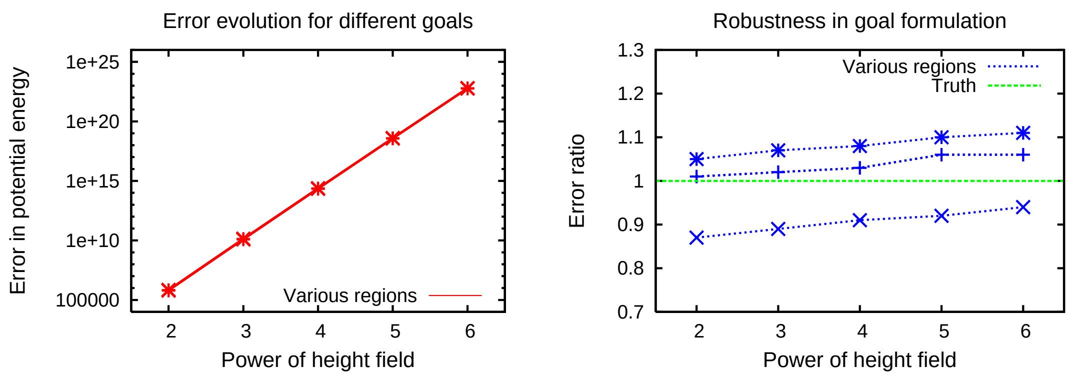


Figure 4.9: Left: On the x-axis different powers of the goal field are plotted. On the y-axis the error in regional potential energy is plotted for three random regions. Right: On the x-axis different powers of the goal field are plotted. On the y-axis the ratio between the estimated error and the true error is plotted for the same three regions. The green line indicates a perfect match between estimate and true error.

in the right plot of Figure 4.9. These results show that our empirical error estimator can work. A detailed discussion of the empirical error estimators is beyond the scope of this chapter and can be found in (Rauser et al. 2010).

For the special treatment of the linear solver outlined in Section 4.5.2 we replaced the originally used custom iterative sparse solver by the direct sparse linear solver UMFPACK Version 5.4 (Davis 2004) running without taping as the factorization of A^{j-1} is reused in the adjoint propagation. For a relevant problem size we get impressive improvements of the adjoint computation for the linear solver both in terms of memory requirement and runtime.

	Black-Box	**Adjoint Direct Solver**
Time for solving linear system (in sec.)	2.4	1.1
Memory for tape and factorization (in GB)	1.2	0.077
Time for computing $\bar{A}^{j-1}(\bar{h}^j_\Delta)$ and $\bar{b}^{j-1}(\bar{h}^j_\Delta)$ (in sec.)	2.08	0.01

We observe a local speedup of 4 in runtime and savings in memory of 94% for adjoining the linear problem, and for the overall process a 50% memory reduction and 27% reduced runtime. The ratio of the runtime of the discrete adjoint to compute $\bar{h}^0_\Delta$ and $\bar{\mathbf{v}}^0_\Delta$ with respect to the runtime of a single evaluation of J_Δ is roughly 10 – a very good value for a solution that is based purely on overloading and tape interpretation rather than source transformation. We are working on a further reduction of this factor by generating parts of the discrete adjoint through source transformation and by exploiting parallelism during the augmented forward evaluation.

4.7 Conclusion

We have shown in this chapter how to construct and improve an differentiation-enabled version of ICOSWM. Differentiating linear operations of the type $Ax = b$ manually improves the performance of the AD version tremendously. This is of general significance because many GFD models solve at least parts of the state vector implicitly. These implicit solvers involve the application of high dimensional matrices, which requires high amounts of memory. In case of iterative solvers, these matrix multiplications are repeated many times. We believe therefore that it is a useful idea to optimize this part of the differentiation process by hand. A next step is the inclusion of manual differentiations into a module to facilitate the implementation. Our idea of manually linearizing these parts of the code by hand will lead to significant performance improvements in many application scenarios.

Summary of Chapter 4

- We show how to obtain the sensitivities that are needed for goal error estimation with Algorithmic Differentiation tools.
- We discuss the properties of the NAG Fortran compiler that are relevant to the problem.
- We suggest a new way to calculate the derivatives of code that involves large matrix multiplication. This is of general interest because many GFD models have to solve large linear systems, especially all implicit methods.

Chapter 5

Goal Error Ensembles with Local Error Random Processes

We introduce a new a posteriori ensemble method to obtain approximation error estimates for relevant physical quantities from a single evaluation of a numerical model. The approximation errors in physical quantities – so-called goals – are estimated as a weighted sum of local model errors on all computational grid cells. The weights are the sensitivities of the goals with respect to local changes of the state of the system. We use an Algorithmic Differentiation tool to approximate these sensitivities. We describe local model errors as a local random process. The full algorithm consists of three steps. First, we choose a general class of local error random processes. Second, we determine a model-specific random process through a local error learning algorithm. The algorithm learns local error properties from local differences between two solutions calculated on varying resolutions. These properties represent the underlying model, the discretization, resolution, and the flow regime. Third, we use different realizations of the local error random process to obtain an ensemble of goal error estimates. The algorithm can be applied to any model of geophysical fluid dynamics because it learns the model-specific properties from model solutions. We use the learned local error random process to produce ensembles of goal approximations with forward ensemble techniques: one ensemble started from perturbed initial conditions and one ensemble produced with stochastic perturbations in the model formulation. We evaluate the algorithm for a shallow water model and examine the evolution of regional potential energy. We show error bounds for a solid body rotation test case and zonal flow against a mountain. A posteriori ensembles compare favorably to stochastic physics ensembles.

5.1 Introduction

Uncertainty quantification is an essential step to improve existing, imperfect models of geophysical fluid dynamics (GFD). It is possible to conceptually separate the sources of this uncertainty into two error layers, the model formulation / specification error and the approximation error (Oden and Prudhomme 2002). The approximation error is the difference between the hypothetically true model solution and a numerical approximation. We estimate the approximation error for physical quantities of interest (goals) that are derived from the solution of a GFD model. These derived physical quantities are called goals. The quantification of goal approximation errors is usually done deterministically (Giles et al. 2004). Deterministic means that local model errors are a deterministic consequence of the modelled flow and the goal error is a deterministic consequence of these local errors. With this work we want to show that stochastic methods can also be used to quantify goal approximation error by generating goal error ensembles. Stochastic means that local model errors are a realization of a random process and the goal error is estimated as a goal error probability density function (PDF). The two interpretations "deterministic" and "stochastic" are closely connected to the Mori Zwanzig formalism (Mori et al. 1974; Zwanzig 1973). This formalism motivates that local model errors are the result of deterministic errors in approximating resolved processes and the effectively stochastic influence of unresolved processes. We show how to use this interpretation of local model error as a random process to construct an error ensemble for relevant goals from a single model solution.

Our concept relies on an original idea how to estimate goal approximation error that was brought forward originally as dual weight error estimation for computational fluid dynamics models (Oden and Prudhomme 2002; Giles et al. 2004; Becker and Rannacher 2002). The main idea of this method is to divide goal errors into local model errors and estimate goal errors as weighted sums thereof. The weights represent the influence of the local model errors, that is the sensitivity of the goal with respect to local changes. The weights are the adjoint solution of a goal-dependent dual problem. The local model errors are modelled a posteriori with local error estimators that depend on the discretization scheme and use the solution. In Chapter 3 we have suggested an extension of this algorithm for GFD applications that incorporates the basic idea and additionally introduces the concept of local learning. The local model errors are described with empirical functionals that structurally do not depend on the underlying Partial Differential Equation (PDE) or its discretization. For a given model and flow the required dependency is expressed in form of degrees of freedom; it is possible to specify a single empirical functional by determining its degrees of freedom. The degrees of freedom are learned in short training runs on different resolutions. The adjoint solution is obtained with Algorithmic Differentiation (Griewank 2000). The empirical functionals of Section 3.3.2 are a deterministic functional of the solution.
In this chapter we describe local errors as a general class of local error random processes. To determine a single random process for a given model and flow, we specify a local error random process with information learned from the model, similarly to the local

error learning algorithm suggested in Section 3.3.2. The resulting “learned” random process incorporates information on the model, the discretization and the flow state. This stochastic approach leads to an ensemble of error estimates from one single model run. In the context of numerical goal error estimation, we are the first to interpret local error production as a local error random process.

The idea of an a posteriori ensemble from a single solution appears counterintuitive at first because classical ensembles consist of multiple solutions of a given problem (forward ensembles). To show how our a posteriori ensemble is connected to this classical concept of an ensemble we combine also forward ensembles with our concept of learning local model errors. We want to investigate if forward ensemble techniques can be used to estimate approximation error, given a “correct” perturbation. There are two commonly used ensemble techniques that rely on multiple forward model runs, the initial condition ensemble and the stochastic physics ensemble. The former has been used for a long time to obtain forecast ensembles to combat initial condition error and model bias (Molteni et al. 1996). The latter is a newer approach that uses stochastic perturbations of the model physics to construct forecast ensembles on top of already perturbed initial conditions (Buizza et al. 1999). We suggest to use both forward ensemble methods to quantify approximation error by using the learned local model error random processes as perturbations. We implement simplified versions of both forward ensembles and compare the result to our a posteriori goal error ensemble.

This chapter is organized as follows: in Section 5.2 we formulate the general problem. Section 5.3 deals with our algorithm proposal to solve this problem. We propose our concepts of stochastic local error estimation and local error learning. In Section 5.4 we introduce a model to evaluate our algorithm and in Section 5.5 results are shown for this model and two test cases. In Section 5.6 we compare the results from a posteriori goal error ensembles with the results of forward ensembles. We conclude in Section 5.8 with some discussion and an outlook.

5.2 Problem Statement

We keep the problem statement general to permit a general formulation of our error estimation algorithm in Section 5.3. We introduce the general model N

$$N(\mathbf{q}(\mathbf{x},t)) = 0, \qquad \mathbf{q}(\mathbf{x},t_0) = \mathbf{q}^0, \;\; \mathbf{q}(\mathbf{x},t) = \mathbf{q}_b \qquad on\ \partial\Omega, \tag{5.1}$$

with $\mathbf{q}(\mathbf{x},t)$ the solution state vector on a space-time domain $\Omega \times T$, $\mathbf{q}^0$ the initial condition and $\mathbf{q}_b(t)$ the boundary conditions on the boundary $\partial\Omega$. The corresponding discrete equations can be summarized as

$$N_\Delta(\mathbf{q}_\Delta) = 0, \qquad \mathbf{q}^0_\Delta = P\mathbf{q}^0, \;\; \mathbf{q}_\Delta = P\mathbf{q}_b \qquad on\ \partial\Omega_\Delta \tag{5.2}$$

with $\mathbf{q}_\Delta = (\mathbf{q}^n_\Delta)_n$ the discrete state vector that incorporates all timeslices of the state $\mathbf{q}^n_\Delta$ in the discrete space-time domain, and the projection operator P that maps the continuous initial and boundary conditions on the discrete space. We are interested in selected physical quantities (goals) $J(\mathbf{q})$ and their approximations $J_\Delta(\mathbf{q}_\Delta)$. The dependency of a goal on the state may include only parts of the full state vector and it may focus on specific regions or times. The error we are interested in is the goal error

$$\varepsilon := J_\Delta(\mathbf{q}_\Delta) - J(\mathbf{q}). \tag{5.3}$$

The classical solution error $\mathbf{e}_\Delta = \mathbf{q} - \mathbf{q}_\Delta$ is a special case of (5.3) with identity as goal, $J = ID$. We try to estimate the error bounds in goals a posteriori, i.e., for a given solution $\mathbf{q}_\Delta$. We introduce the error bounds $\varepsilon_{max} > 0$ and $\varepsilon_{min} < 0$ that constrain the original functional value

$$J_\Delta(\mathbf{q}_\Delta) + \varepsilon_{min} < J(\mathbf{q}) < J_\Delta(\mathbf{q}_\Delta) + \varepsilon_{max}. \tag{5.4}$$

We summarize the general problem statement: Given a model N and its discretization N_Δ, how can we estimate error bounds ε_{max} and ε_{min} that quantify the uncertainty for arbitrary physical quantities J so that $\varepsilon_{min} < \varepsilon < \varepsilon_{max}$?

5.3 Stochastic Quantification of Goal Approximation Errors

Following original work from (Mori 1965; Mori et al. 1974; Zwanzig 1973) and a review article from (Givon et al. 2004) we see that any model description implies local errors that can be described both stochastically and deterministically. Any discrete model description is equivalent to the extraction of resolved dynamics from a process of higher complexity. The numerical model N_Δ (5.2) is a low order approximation of the full problem N (5.1). The state vector $\mathbf{q} = (\mathbf{q}_\Delta, \hat{\mathbf{q}})$ consists of a resolved part $\mathbf{q}_\Delta$ and an unresolved part $\hat{\mathbf{q}} \in \mathcal{Y}$ (with $\mathcal{Y}$ representing the space of unresolved scales). Classical GFD models model the time evolution of resolved scales as a function f of the resolved scales

$$N_\Delta(\mathbf{q}_\Delta) := \frac{d\mathbf{q}_\Delta}{dt} - f(\mathbf{q}_\Delta) = 0. \tag{5.5}$$

We use the Mori Zwanzig approach to rewrite Equation (5.1) as

$$\frac{d\mathbf{q}_\Delta}{dt} - f(\mathbf{q}_\Delta) + M(\mathbf{q}_\Delta(t)) + O(\mathbf{q}_\Delta(0), \hat{\mathbf{q}}(0)) = 0, \tag{5.6}$$

with M, the so-called memory kernel of all interactions between $\mathbf{q}_\Delta$ and $\hat{\mathbf{q}}$, and O the orthogonal dynamics equation. We see in (5.6) that the unresolved scales $\hat{\mathbf{q}}$ also have an influence on the exact time evolution of the resolved scales $\mathbf{q}_\Delta$. If we compare Equation (5.5) and Equation (5.6) we see that this influence is usually neglected. Deterministic local model errors occur if we handle the influence f of the resolved scales $\mathbf{q}_\Delta$ wrongly. Stochastic local model errors occur because we neglect the influence of the unresolved scales. Therefore, as long as there are unresolved scales, numerical errors always occur.
Local error production is a complicated function of the resolved and unresolved variables and can be described either stochastically or deterministically. Previous works have tried to describe local model errors with a deterministic function of the state $\mathbf{q}_\Delta$. In this chapter we describe local model errors as a stochastic random process.

5.3.1 The Algorithm Proposal

If local model errors are realizations of a random process, the outcome of a specific model run is an aggregated random process, with respective probability distributions for the approximations of relevant goals. The model is uncertain of the solution it calculates and the algorithm we propose needs to quantify this degree of uncertainty. We assume that the uncertainty in goal approximation is connected to the properties of local model errors. The advantage of this reasoning is the fact that the properties of local model errors can be learned from model solutions on different resolutions. The local grid point differences of model solutions on different resolutions are an indicator of the local model error. This concept shares the general idea of error learning with the algorithm in Section 3.3. We propose a three-step algorithm:

Algorithm 1

1. Define a general class of local error random processes $\mathcal{P}(\mathbf{p})$ that describe local model errors and that are determined by a parameter set $\mathbf{p}$.
2. Learn a model-specific parameter set $\tilde{\mathbf{p}}$ in short training runs on varying resolution, using local differences between solutions on different resolutions as realizations of the local error random process.
3. Use the local error random process $\mathcal{P}(\tilde{\mathbf{p}})$ with learned parameter set $\tilde{\mathbf{p}}$ as local perturbations to create goal approximation ensembles for a given solution.

This approach is general and does not yet indicate a particular choice for steps 1 to 3. The key idea is that the class of random processes $\mathcal{P}$ can be chosen a priori model-independent. The model-dependency comes into play through a specific parameter set $\tilde{\mathbf{p}}$ that is different for different models, discretizations or resolutions and has to be tuned accordingly. This tuning is a learning step: the model uses solutions on various resolutions to learn a specific parameter set $\tilde{\mathbf{p}}$. We show proposals for steps 1 to 3 in the next sections.

5.3.2 Step 1: Local Error Random Processes

As a first step, Algorithm 1 requires to specify a set of local error random processes $\mathcal{P}$ that describe the distribution of local model errors. We suggest a memory-less Gaussian Normal distribution $\mathcal{N}$ as null hypothesis

$$\mathcal{P}(\mathbf{p}) := \mathcal{N}(\mu, \sigma). \tag{5.7}$$

The parameter set $\mathbf{p} = (\mu, \sigma)$ consists of the mean μ and the standard deviation σ. The exact structure of the local error random process depends on model, discretization and resolution. The Normal distribution is a priori equivalent to the assumption that all unresolved processes combine to an approximately Normal distribution by the Central Limit Theorem. This holds strictly only in case of a scale separation when a large number of unresolved processes act on the resolved scales. We use the Gaussian process for its simplicity but it is clear that other distributions can also be chosen at this point.

5.3.3 Step 2: Learning the Properties of Local Error Random Processes

As a second step, Algorithm 1 requires a learning algorithm that determines a unique parameter set $\tilde{\mathbf{p}}$ that selects a single model-specific random process out of the set of the random processes $\mathcal{P}(\mathbf{p})$. We have introduced the Gaussian $\mathcal{N}(\mu, \sigma)$ as the class of local error random processes and need therefore a learning technique that determines the specific mean $\tilde{\mu}$ and the specific standard deviation $\tilde{\sigma}$ for a given model and model solution. We suggest a learning algorithm that uses all local grid point errors between model solutions on varying resolutions (at least one higher resolution is necessary).

1. Integrate the model for one time step on the low standard resolution and $j \geq 1$ available higher resolutions and obtain a set of high-resolution solutions $\mathbf{q}_{\Delta j}$ and the standard solution $\mathbf{q}_{\Delta,low}$.

2. Use a projection operator I_j to project all higher resolution solutions onto the grid of the low-resolution solution

$$\mathbf{q}_{\Delta j,low} := I_j \mathbf{q}_{\Delta j}. \tag{5.8}$$

3. For each higher resolution solution $\mathbf{q}_{\Delta j}$ calculate the vector of pointwise local errors as grid point differences between the projected high-resolution solution and the original low-resolution solution
$$\mathbf{e}_{\Delta j} := \mathbf{q}_{\Delta j,low} - \mathbf{q}_{\Delta,low}. \tag{5.9}$$
4. For each higher resolution j calculate one parameter set of mean $\mu_{\Delta j}$ and standard deviation $\sigma_{\Delta j}$ of local errors
$$\mu_{\Delta j} := \frac{1}{K} \sum_{\Delta j} \mathbf{e}_{\Delta j} \tag{5.10}$$
and
$$\sigma_{\Delta j} := \sqrt{\frac{1}{K} \sum_{\Delta j} (\mathbf{e}_{\Delta j} - \mu_{\Delta j})^2}, \tag{5.11}$$
with K the number of computational cells of the low-resolution solution and $\sum_{\Delta}$ the sum over each computational grid cell on resolution Δj.
5. Average the different $\mu_{\Delta j}$ and $\sigma_{\Delta j}$ over all resolutions j to determine the final parameter set $\tilde{\mathbf{p}} = (\tilde{\mu},\ \tilde{\sigma})$.

This training algorithm is cheap because it only needs one time step of the forward higher resolution solutions. It is also robust because the total number of realizations scales with the spatial degrees of freedom. More available different resolutions mean a higher amount of information that the algorithm can analyze to determine $\tilde{\mathbf{p}}$, or in turn $\mathcal{N}(\tilde{\mu}, \tilde{\sigma})$. There are two points in this algorithm that necessitate further explanation: the projection operator I_j and the length of the time step. We suggest reconstruction-type projection operators for I_j, i.e., operators that reconstruct pointwise values from the high-resolution solution. A nearest neighbor approach is a very simple reconstruction-type approach, where the value of nearest neighbor cells of a high-resolution solution are used to reconstruct the value of the low-resolution solution. We have chosen this method to keep the implementation simple but other projection operators are also possible. We suggest to use the length of the time step of the standard model solution as length of the learning period. For reference solutions $\mathbf{q}_{\Delta j}$ that are much higher resolved than the standard resolution $\mathbf{q}_{\Delta,low}$ we need to adapt the time step so that the model is still stable.

5.3.4 Step 3: A Posteriori Goal Error Ensembles

As a third step, Algorithm 1 requires to construct a goal error ensemble based on a given specified local error random process $\mathcal{P}(\tilde{\mathbf{p}})$. To do this we propose a stochastic

variant of the deterministic method suggested in Section 3.3.
Our method is based on the assumption that goal errors ε (5.3) can be approximated as the scalar product of estimated local model errors $\hat{N}_\Delta(\mathbf{q}_\Delta)$ and the sensitivity $\mathbf{q}^*_\Delta$ of the goal with respect to local model changes (e.g., Giles and Pierce 2000; Becker and Rannacher 2002; Johnson et al. 1995)

$$\varepsilon_{est} := \left\langle \mathbf{q}^*_\Delta, \hat{N}_\Delta(\mathbf{q}_\Delta) \right\rangle_{\Omega \times T} \approx \varepsilon. \tag{5.12}$$

The choice of the scalar product $\langle ., . \rangle_{\Omega \times T}$ in time and space is a priori arbitrary. The adjoint solution $\mathbf{q}^*_\Delta$ depends on this choice, though, and on the choice of model and goal. Throughout this Chapter we use a standard Euclidean scalar product (each discrete grid point is weighted with areas and time step length). We omit the explicit notation of $\Omega \times T$ unless needed for clarification. The sensitivities $\mathbf{q}^*_\Delta$ are the solution of an adjoint problem defined by the model N and the goal J. They are approximated with the help of Algorithmic Differentiation (see Chapter 4, (Naumann and Riehme 2006)). The local model error estimator $\hat{N}_\Delta(\mathbf{q}_\Delta)$ is classically discretization-dependent and reflects the structure of the underlying PDE. We suggest a new variant that interprets local errors as a stochastic local random process. Instead of one error estimate we aim to obtain a goal approximation error PDF. We replace the local error estimators $\hat{N}_\Delta(\mathbf{q}_\Delta)$ of Equation (5.12) by the specified random process $\mathcal{P}(\tilde{\mathbf{p}})$

$$\hat{N}_\Delta(\mathbf{q}_\Delta) := \frac{1}{\Delta t}\mathcal{P}(\tilde{\mathbf{p}}) \left(= \frac{1}{\Delta t}\mathcal{N}(\tilde{\mu}, \tilde{\sigma}) \right), \tag{5.13}$$

with Δt the timestep and the brackets reflecting our specific choice of random process in step 1. The procedure to determine a posteriori ensembles works for any chosen and specified random process $\mathcal{P}(\tilde{\mathbf{p}})$. The error estimates are fully determined by the parameter set $\tilde{\mathbf{p}}$

$$\varepsilon_{est} = \left\langle \mathbf{q}^*_\Delta, \frac{1}{\Delta t}\mathcal{P}(\tilde{\mathbf{p}}) \right\rangle. \tag{5.14}$$

This means that for each computational cell in space and time we draw a random number from the identical random process $\mathcal{P}(\tilde{\mathbf{p}})$ and multiply the resulting random number with the adjoint sensitivity at that cell. Equation (5.14) is not a useful error estimate in it self. Instead of a single error estimate we calculate R realizations of the scalar product (5.14) to obtain error bounds

$$\begin{aligned} \varepsilon_{max} &= \max_R \left\langle \mathbf{q}^*_\Delta, \frac{1}{\Delta t}\mathcal{P}(\tilde{\mathbf{p}}) \right\rangle, & (5.15) \\ \varepsilon_{min} &= \min_R \left\langle \mathbf{q}^*_\Delta, \frac{1}{\Delta t}\mathcal{P}(\tilde{\mathbf{p}}) \right\rangle. & (5.16) \end{aligned}$$

The goal error ensemble can be constructed as an ensemble of perturbed goal approximations $\tilde{J}_\Delta$

$$\tilde{J}_\Delta := J_\Delta(\mathbf{q}_\Delta) + \left\langle \mathbf{q}^*_\Delta, \frac{1}{\Delta t}\mathcal{P}(\tilde{\mathbf{p}}) \right\rangle . \tag{5.17}$$

The resulting ensemble of goal error estimates from a single solution $\mathbf{q}_\Delta$ is called "A Posteriori Goal Error Ensemble".
We summarize step 3 of our algorithm:

1. Calculate a solution $\mathbf{q}_\Delta$ of the model N_Δ to obtain an approximation of a relevant goal $J_\Delta(\mathbf{q}_\Delta)$.

2. Calculate an adjoint solution $\mathbf{q}^*_\Delta$ of the problem to obtain the weights $\mathbf{q}^*_\Delta$.

3. Calculate R scalar products (5.14) between $\mathbf{q}^*_\Delta$ and the specified local error random process $\mathcal{P}(\tilde{\mathbf{p}})$ from steps 1 and 2.

Multi-component state vectors

For complex models and on high resolutions the discrete state vector $\mathbf{q}_\Delta$ is high dimensional and may consist of various fields of different prognostic variables $\mathbf{q}_\Delta = (\mathbf{q}_1, ..., \mathbf{q}_m)$ (for m prognostic variables). It is reasonable to assume that the local error properties of different prognostic variables $\mathbf{q}_i$ are not identical. We therefore suggest to learn local error random processes for each part of the state vector $\mathbf{q}_\Delta$ separately.
At the same time we suggest to only use local error random process for those parts of the state vector that are actually used to calculate the goal J_Δ. This is due to technical reasons. To calculate the scalar product in Equation (5.17) requires local error estimates for all prognostic variables $\mathbf{q}_i$ and the solution of the adjoint sensitivities for all prognostic variables. This becomes computationally expensive to accomplish for a high number of prognostic variables in high spatial and temporal resolutions. Using only local error random processes for variables that are used directly to calculate the goal is identical to the assumption that the errors in the other parts of the state vector have only a minor influence on the goal because the corresponding sensitivities are small.

5.3.5 Forward Ensembles

We propose that classical forward ensemble techniques can be used for quantification of goal approximation error if they use the correct local error random process $\mathcal{P}(\tilde{\mathbf{p}})$. The forward ensembles can be used to understand the concept of our a posteriori ensemble better. We introduce the basic concepts of two classical forward ensemble techniques.

Initial Condition Ensemble (ICE)

The main properties of initial condition ensembles are

- The initial state $\mathbf{q}_\Delta^0$ is perturbed by the random process $\mathcal{P}(\tilde{\mathbf{p}})$

$$\tilde{\mathbf{q}}_\Delta^0 := \mathbf{q}_\Delta^0 + \mathcal{P}(\tilde{\mathbf{p}}). \tag{5.18}$$

- The model N_Δ is solved R times from R different initial condition $\tilde{\mathbf{q}}_\Delta^0$, yielding an ensemble of solutions $\tilde{\mathbf{q}}_\Delta$.

- The goal $\tilde{J}_\Delta$ represents one instance of the ensemble and is derived from each model solution $\tilde{\mathbf{q}}_\Delta$. We define the error bounds as minimum and maximum over the ensemble

$$\varepsilon_{min} := \min_R(\tilde{J}_\Delta - J_\Delta) \tag{5.19}$$

$$\varepsilon_{max} := \max_R(\tilde{J}_\Delta - J_\Delta) \tag{5.20}$$

  with J_Δ the solution without initial condition noise and $\tilde{J}_\Delta$ one perturbed instance of the goal ensemble.

Stochastic Physics Ensemble (SPE)

The main properties of stochastic physics ensembles are

- The model formulation $N_\Delta(\mathbf{q}_\Delta) = 0$ is perturbed. The perturbations can act on parameterizations, tendencies, forcings or boundary conditions. We use the local error random process $\mathcal{P}(\tilde{\mathbf{p}})$ as stochastic forcing

$$N_\Delta(\mathbf{q}_\Delta) = \mathcal{P}(\tilde{\mathbf{p}}) \tag{5.21}$$

- The model N_Δ is solved R times, using different realizations of the stochastic process $\mathcal{P}(\tilde{\mathbf{p}})$, yielding an ensemble of solutions $\tilde{\mathbf{q}}_\Delta$.

- The goal $\tilde{J}_\Delta$ represents one instance of the ensemble and is derived from each model solution $\tilde{\mathbf{q}}_\Delta$. We define the error bounds identical to the initial condition ensemble as minimum and maximum over the ensemble

$$\varepsilon_{min} := \min_R(\tilde{J}_\Delta - J_\Delta) \tag{5.22}$$

$$\varepsilon_{max} := \max_R(\tilde{J}_\Delta - J_\Delta) \tag{5.23}$$

  with J_Δ the solution without initial condition noise and $\tilde{J}_\Delta$ one perturbed instance of the goal ensemble.

ICON grid properties			
Refinement level	Number of cells	Average cell distance	Time step length
$\Delta 1$	320	1115.3 km	900 s
$\Delta 2$	1280	556.4 km	600 s
$\Delta 3$	5120	278.0 km	450 s
$\Delta 4$	20480	139.0 km	200 s
$\Delta 5$	81920	69.5 km	100 s
$\Delta 6$	327680	34.7 km	50 s

Table 5.1: Basic properties of ICON discretization. One refinement level is equivalent to a quadrupling of the number of cells by halving the triangle edge lengths. Refinement level $\Delta 1$ is a two times refined icosahedron (4 * 4 * 20 cells). Average cell distance is the average of all the distances between triangle cell centers.

Both initial condition and stochastic physics ensembles are forward ensembles because they are created by solving a given model N_Δ for different realizations of a perturbation. Both create a goal approximation ensemble; the goal error ensembles are derived from comparisons to the unperturbed solutions.

5.4 The Testbed

We have introduced the general problem in Section 5.2 and a general possible solution strategy in Section 5.3. We now introduce the testbed for the evaluation of our algorithm: the model and two test cases.

The shallow water equations (SWE) on a rotating sphere are a specific example of the general operator N of Section 5.2. We write the inviscid SWE on the sphere Ω in vector invariant form as

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} &= (\xi + f)\mathbf{k} \times \mathbf{v} - \nabla(gh + \frac{1}{2}|\mathbf{v}|^2) \\ \frac{\partial h}{\partial t} &+ \nabla \cdot (h\mathbf{v}) = 0. \end{aligned} \tag{5.24}$$

Here $\mathbf{v}$ is the horizontal velocity, ξ the vorticity, f the Coriolis parameter, $g = 9.81\, m/s^2$ the gravitational acceleration, and h the height of the fluid surface. The initial conditions are $\mathbf{v}(t_0) = \mathbf{v}_0$ and $h(t_0) = h_0$. We consider (5.24) on a time interval $T := [t_0, t_n]$ and with periodic spatial boundary conditions on the sphere Ω. The state vector $\mathbf{q} = (h, \mathbf{v})$ consists of the prognostic fields height and velocity. The hyperbolic partial differential equations (5.24) describe the flow of a single layer of fluid.

Our numerical framework N_Δ is ICOSWM, a shallow water model on a triangular spherical grid with C-type staggering of the variables. ICOSWM uses a hybrid finite volume / finite difference method to approximate the SWE (5.24). ICOSWM calculates a solution vector $\mathbf{q}_\Delta = (h_\Delta, \mathbf{v}_\Delta)$ with h_Δ the discrete height field in the cell centres of our triangular grid and $\mathbf{v}_\Delta$ the normal velocities at the mid points of the triangular edges. The solution process is sequential in nature, the discrete model yields discrete time slices $\mathbf{q}_\Delta^n$ for each time step. In our notation, the solution vector $\mathbf{q}_\Delta = (h_\Delta, \mathbf{v}_\Delta)$ incorporates all time slices and represents the discrete approximation of the full solution. For further details see (Giorgetta et al. 2009; Ripodas et al. 2009).

We choose a reference goal J for our evaluation: regionally averaged potential energy.

$$J(\mathbf{q}) := J(h(t_{end})) = \frac{g}{A(\Omega_0)} \int_{\Omega_0} h^2(\mathbf{x}, t_{end}) d\mathbf{x}, \tag{5.25}$$

where Ω_0 denotes an arbitrary subdomain of the sphere Ω and $A(\Omega_0)$ denotes the area of Ω_0. The goal depends directly only on the height field h at the end time t_{end} as part of the state vector $\mathbf{q}$. We omit the factor 1/2 in the definition of potential energy because a constant factor does not change the structural form of the goal functional and its error characteristics. The computational equivalent is the numerical integration of an approximated discrete height field after n time steps h_Δ^n on the discrete subdomain $\Omega_{\Delta 0}$

$$J_\Delta(\mathbf{q}_\Delta) := J_\Delta(h_\Delta^n) = \frac{g}{A_\Delta(\Omega_{\Delta 0})} \sum_{i \in \Omega_{\Delta 0}} a_i \left(h_{\Delta,i}^n\right)^2, \tag{5.26}$$

where the a_i denote the grid cell areas, $h_{\Delta,i}^n$ is the value of the discrete height field after n time steps on the ith triangle. The discrete area $A_\Delta(\Omega_{\Delta 0}) = \sum_{i \in \Omega_{\Delta 0}} a_i$ is the sum of all triangle areas that are part of the subdomain $\Omega_{\Delta 0}$ and approximates the true area $A(\Omega_0)$.

We apply our new error estimation technique to two test cases that are commonly used in the GFD community: 1) a solid body rotation test case (TC1) as introduced in example 3 of (Laeuter et al. 2005) and 2) zonal wind against a mountain as described in (Williamson and Drake 1992) (TC2). The topography and height field initial condition of our test cases are plotted in Figure 5.1. The solid body rotation test case (TC1) is interesting because it has an analytical solution that allows clear comparisons between actual performance of our error estimation algorithm and the best theoretical possible performance. It covers a smooth wave-type flow on a zonal topography with realistic velocities. This test case is not very realistic but it allows us a first evaluation if our method can be applied to time-dependent GFD models at all. We prefer it to other classical test cases with analytical solution because of its time-dependent nature. The second test case (TC2) is taken from the classical Williamson test suite for shallow water models. It describes an initial steady zonal flow that gets perturbed by a mountain.

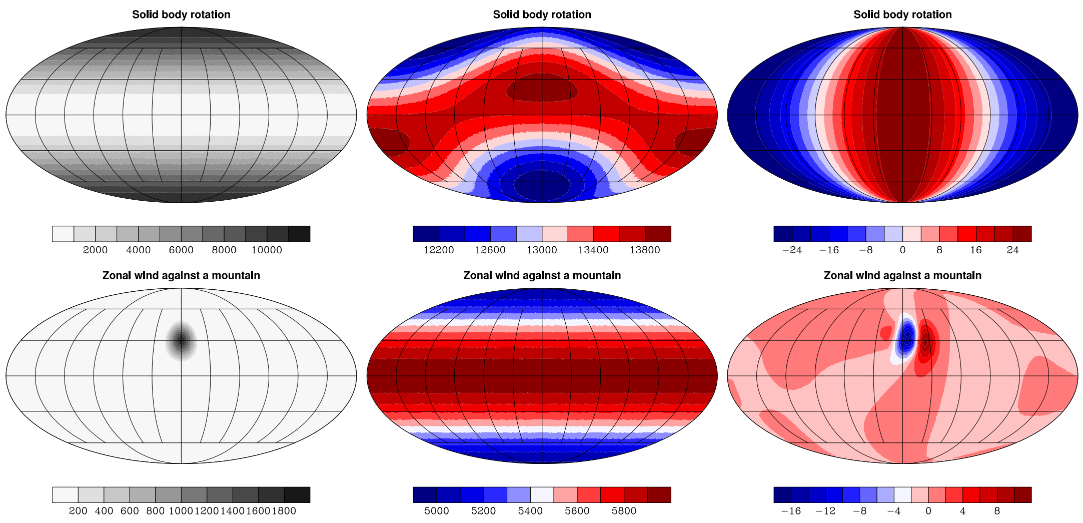


Figure 5.1: Topography (left), height field initial condition (middle), meridional velocity after 24hours (right). Top row for solid body rotation, bottom row for zonal wind against a mountain.

The initial condition is a smooth height field and the mountain appears instantaneously in the flow. This perturbation excites an initial gravity wave. After the initial perturbation the mountain causes Rossby waves that form a standing wave behind the mountain.

We implement the ensemble strategies of Section 5.3.4 and Section 5.3.5 for ICOSWM. The perturbations for all ensembles are instances of a Gaussian random process $\mathcal{N}$ that are derived from the standard Fortran uniform random generator via the Marsaglia polar method (Marsaglia 1991) for a specified set $\tilde{\mathbf{p}}$. The initial condition perturbations are added on the initial height field

$$\tilde{h}^0_\Delta := h^0_\Delta + \mathcal{N}(\tilde{\mu}, \tilde{\sigma}). \tag{5.27}$$

The stochastic physics perturbations are added as a stochastic forcing on the height field at each time step n

$$\tilde{h}^n_\Delta := h^n_\Delta + \mathcal{N}(\tilde{\mu}, \tilde{\sigma}). \tag{5.28}$$

The perturbations for the a posteriori ensembles are included in the scalar product

$$\left\langle \mathbf{q}^*_\Delta, \frac{1}{\Delta t}\mathcal{N}(\tilde{\mu}, \tilde{\sigma}) \right\rangle . \tag{5.29}$$

TC1 height field properties for 1 time step: mean $\tilde{\mu}$
(Analytical solution)

	Evaluation time of analytical solution$\longrightarrow$					
Resolution $\downarrow$	50s	100s	200s	450s	600s	900s
$\Delta 1$	1.0×10^{-6}	1.7×10^{-5}	6.5×10^{-5}	3.1×10^{-4}	5.2×10^{-4}	$\mathbf{1.0} \times 10^{-3}$
$\Delta 2$	5.4×10^{-7}	8.6×10^{-6}	3.3×10^{-5}	1.4×10^{-4}	$\mathbf{2.0} \times 10^{-4}$	3.2×10^{-4}
$\Delta 3$	2.4×10^{-7}	3.6×10^{-6}	1.2×10^{-5}	$\mathbf{3.4} \times 10^{-5}$	4.4×10^{-5}	7.5×10^{-5}

TC1 height field properties for 1 time step: standard deviation $\tilde{\sigma}$
(Analytical solution)

	Evaluation time of analytical solution$\longrightarrow$					
Resolution $\downarrow$	50s	100s	200s	450s	600s	900s
$\Delta 1$	8.94×10^{-2}	0.357	0.711	1.55	2.03	**2.86**
$\Delta 2$	4.87×10^{-2}	0.194	0.379	0.774	**0.955**	1.21
$\Delta 3$	2.76×10^{-2}	0.108	0.201	**0.348**	0.398	0.477

Table 5.2: Mean and standard deviation as obtained from one time step learning runs on different standard resolutions compared to the analytical solutions. The three rows are equivalent to the three different standard model resolutions. The six columns are equivalent to the six standard model time step lengths.

5.5 Results

In this section we evaluate Algorithm 1 in the test bed of Section 5.4. First, we analyze the behavior of our learning algorithm (Step 2 of Algorithm 1). We then show results when the gauged random process is used to produce an a posteriori ensemble of error estimates for important regional physical quantities (Step 3 of Algorithm 1). As a last step we show results for two forward ensembles, perturbed by the same local error random process.

5.5.1 Learning for Different Test Cases

As a first step, we want to analyze the true form of the local error rate of change. We can do this within the solid body rotation test case (TC1) because of its analytical solution. We look at regional potential energy J derived from the solution of TC1. The fact that this functional depends only on a part of our state vector allows us to use the suggested reduction in dimensionality of Section 5.3.4. We only look at the height field h_Δ as part of the state vector $\mathbf{q}_\Delta$ and ignore the local errors in velocities.

TC1 height field properties for 1 time step: $\tilde{\sigma}$ (Reference numerical solution $\Delta 5$)						
	Time step of model and reference solution$\longrightarrow$					
Resolution $\downarrow$	50s	100s	200s	450s	600s	900s
$\Delta 1$	12.1	12.1	12.1	12.21	12.3	**12.4**
$\Delta 2$	7.3	7.3	7.3	7.3	**7.3**	7.4
$\Delta 3$	2.4	2.4	2.4	**2.4**	2.4	2.5

Table 5.3: Mean and standard deviation as obtained from one time step learning runs on different standard resolutions compared to a reference numerical solution on resolution $\Delta 5$. The three rows denote the three different standard model resolutions. The six columns denote the six standard model time step lengths.

TC1 height field properties for 1 time step: $\tilde{\sigma}$ (Reference numerical solution time step 900s)							
	Resolution of reference solution$\longrightarrow$						
Resolution $\downarrow$	1	2	3	4	5	6	$\oslash$
$\Delta 1$	0	19.7	10.1	14.8	12.4	13.6	14.1
$\Delta 2$	-	0	9.8	5.0	7.4	6.2	7.1
$\Delta 3$	-	-	0	4.9	2.5	3.7	3.7

TC2 height field properties for 1 time step: $\tilde{\sigma}$ (Reference numerical solution time step 900s)							
	Resolution of reference solution$\longrightarrow$						
Resolution $\downarrow$	1	2	3	4	5	6	$\oslash$
$\Delta 1$	0	10.6	5.4	8	6.7	7.4	7.6
$\Delta 2$	-	0	5.3	2.7	4.0	3.34	3.8
$\Delta 3$	-	-	0	2.6	1.3	2.0	2.0

Table 5.4: Standard deviation as obtained from one time step learning runs on different reference resolutions. The three rows are equivalent to the three different standard model resolutions. The six columns are equivalent to the six possible reference model resolutions. The $\oslash$ column denotes the average of the standard deviations.

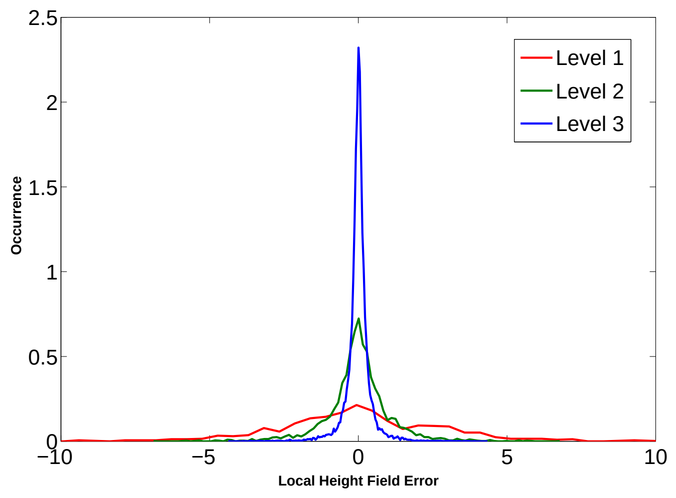


Figure 5.2: TC1 local error PDF calculated from three resolutions (one time step).

We solve ICOSWM for one time step on resolutions $\Delta1$ to $\Delta3$ (see table for their respective resolution and time step length). We compare the respective local height field solution h_Δ pointwise with the discrete projection of the analytical solution after one time step to calculate the local errors of Equation (5.9). Using the learning algorithm of Section 5.3.3 we obtain values for the mean and the standard deviation for all time step lengths that are used in ICOSWM with a focus on the typical time step length for each resolution, see Table 5.2. The first conclusion is that the mean of our process is effectively zero. This is in line with expectations, a non-zero local error distribution would introduce strong biases in local solutions and would probably not be stable. As a second conclusion, we see that the standard deviation critically depends on the choice of time step length. This can be easily explained by the initialization with pointwise values of the underlying analytical initial condition. For very short integration time the errors between the discrete model and the analytical solution therefore converge to zero, the initial state. It is not a priori clear which is the "correct" random process but we think that clearly the most useful data comes from the time step length that is actually used in the model (bold in Table 5.2).
We use the analytical solution of TC1 to plot the approximate PDFs of the local error random process for resolutions $\Delta1$ to $\Delta3$ in Figure 5.2.

5.5.2 A Posteriori Goal Error Ensembles

We use the learned mean and standard deviation from Section 5.5.1 to estimate goal approximation error PDFs for regional potential energy J. We use the averaged learned

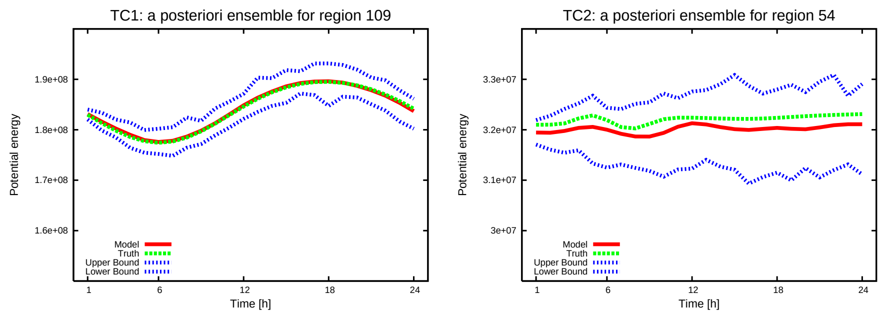


Figure 5.3: Goal approximation and error bounds for regional potential energy on resolution Δ1 for 24 hours. Left solid body rotation, right zonal flow against a mountain.

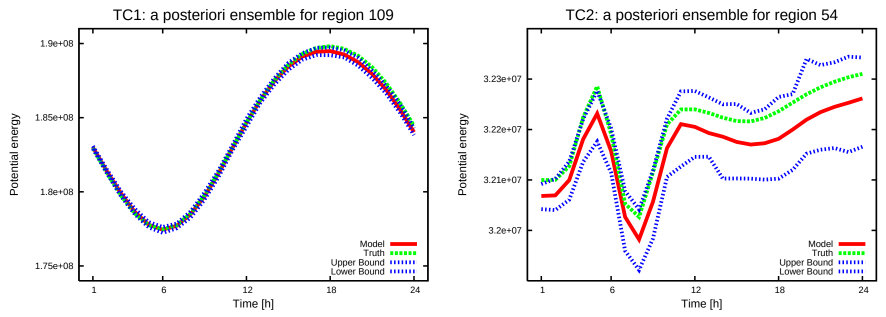


Figure 5.4: Goal approximation and error bounds for regional potential energy on resolution Δ2 for 24 hours. Left solid body rotation, right zonal flow against a mountain.

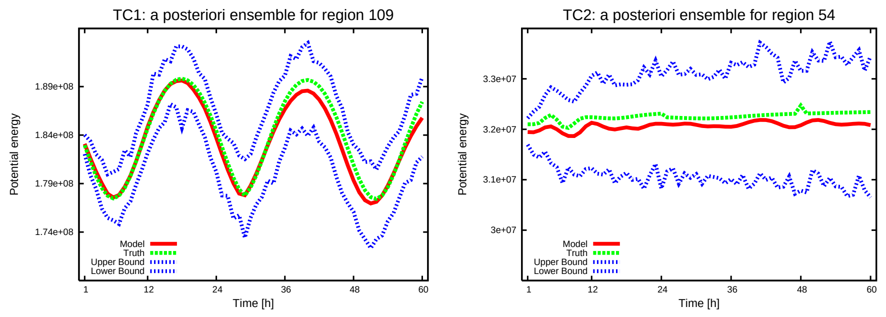


Figure 5.5: Goal approximation and error bounds for regional potential energy on resolution Δ1 for 60 hours. Left solid body rotation, right zonal flow against a mountain.

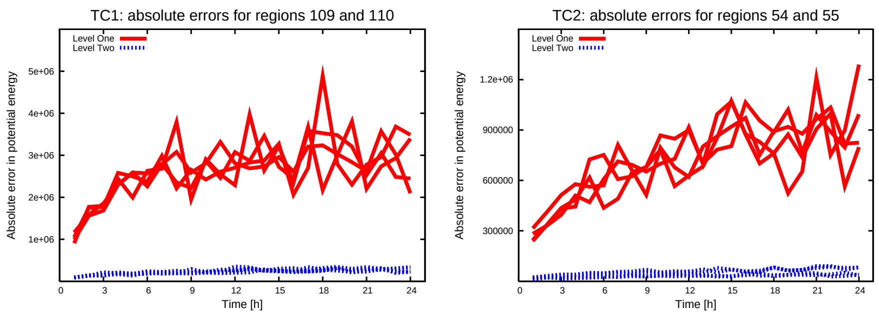


Figure 5.6: The evolution of absolute error bounds in time for TC1 (left) and TC2 (right) on resolution Δ1

$\tilde{\sigma}$ from Table 5.4 to create an a posteriori ensemble of 100 stochastic error estimates and calculate error bounds (Equation (5.15)) to see if the "true" error lies within the error bound. To calculate the "true" error we define the true goal value as the reference truth of the highest possible resolution Δ6 for both test cases. We solve the model on the three low standard resolutions Δ1 and Δ2 for both test cases. For all four experiments we show results for a typical region for regional potential energy for 24 hours. The respective learned $\tilde{\sigma}$ lead to error bounds that confine the true error. The error bounds seem to cover the original model error quite well for 24 hours. We can see that we overestimate the error on the lowest resolution Δ1 but capture it quite well on Δ2 (see Figure 5.3 and Figure 5.4). We plot the mean, minimum and maximum of the ensemble to represent the ensemble spread. The most probable error from this ensemble is much closer to the original approximation than to the lower and upper bounds.
We extend the integration time to 60 hours for the low-resolution Δ1 experiments to check the long-term development of the ensemble spread, again for both test cases TC1 and TC2. The spread behavior is similar; for both TC1 and TC2 the ensemble spread only increases slightly after the initial 24 hours (see Figure 5.5). The initial increase is stronger for TC2. It appears that the ensemble spread depends on the linearity of the test case. We test if the ensemble spread grows linearly by taking the absolute values of ε_{min} and ε_{max}. The errors do not grow strictly linearly in time (see Figure 5.6).

Robustness with Respect to Goal Changes

To test the robustness of our method with respect to goal changes we introduce two new goals ${J_\Delta}^2$ and ${J_\Delta}^3$

$$\begin{aligned} {J_\Delta}^2 &:= J_\Delta(h^n_\Delta) = \frac{g}{A_\Delta(\Omega_0)} \sum_{i\in\Omega_0} a_i \left(h^n_{\Delta,i}\right)^{10}, && (5.30) \\ {J_\Delta}^3 &:= J_\Delta(\mathbf{v}^n_\Delta) = (\nabla \times \mathbf{v}^n_\Delta)_{\Omega i}\,. && (5.31) \end{aligned}$$

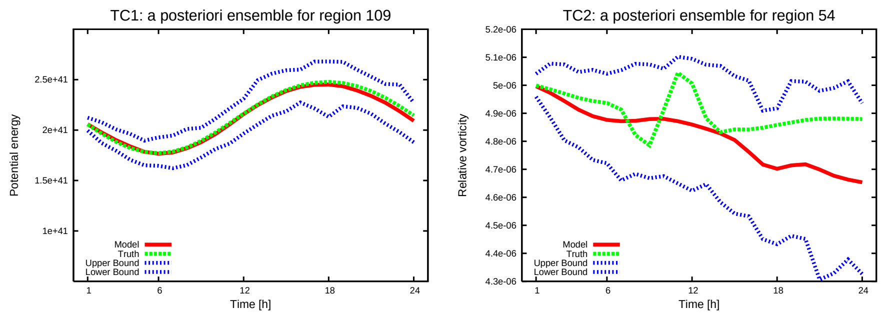


Figure 5.7: Error bounds for two goals for 24 hours. Left ${J_\triangle}^2$ for TC1, right ${J_\triangle}^3$ for TC2.

Goal ${J_\triangle}^2$ is a variation of the potential energy goal $J_\triangle$ with a different functional dependency on the height field. Goal ${J_\triangle}^3$ is the relative vorticity at a given grid vertex with no direct dependency on the height field. We use the identical specified local random process $\mathcal{N}(\tilde{\mu}, \tilde{\sigma})$ to calculate an ensemble of scalar products. For ${J_\triangle}^2$, the change of the goal formulation does not change the behavior of the error estimates although it changes the size of the errors itself quite drastically (compare TC1 results in Figure 5.3 and Figure 5.7). For ${J_\triangle}^3$, the radical change of the goal formulation does not change the structural behavior of the error estimates (compare TC2 results in Figure 5.3 and Figure 5.7). We conclude that – given a specified random process – variations in the goal formulation do not change the error estimation behavior.

The Distribution of A Posteriori Goal Errors

We have argued that the Gaussian is a reasonable starting point for local error random processes because it is the natural distribution if there is a scale separation between the resolved and unresolved processes. However, this necessary scale separation is not given a priori for arbitrary resolutions of our discrete model. Looking at the data from Figure 5.2, one might argue that for our test case the local error PDF fits also to an exponential distributions. Given the inherent ambiguity of the choice of the local error process we want to check how sensitive our algorithm is to the choice of the form of local error random processes.
We introduce a second local error random process (approximately an exponential distribution) with an underlying PDF of the following form:

$$f(x) = \sigma e^{-x}, \tag{5.32}$$

with σ the standard deviation we used before for the Gaussian process. We compare this new random process to the original Gaussian process in Figure 5.9. The two local model error PDFs look distinctively different. To see the form of the approximate PDF

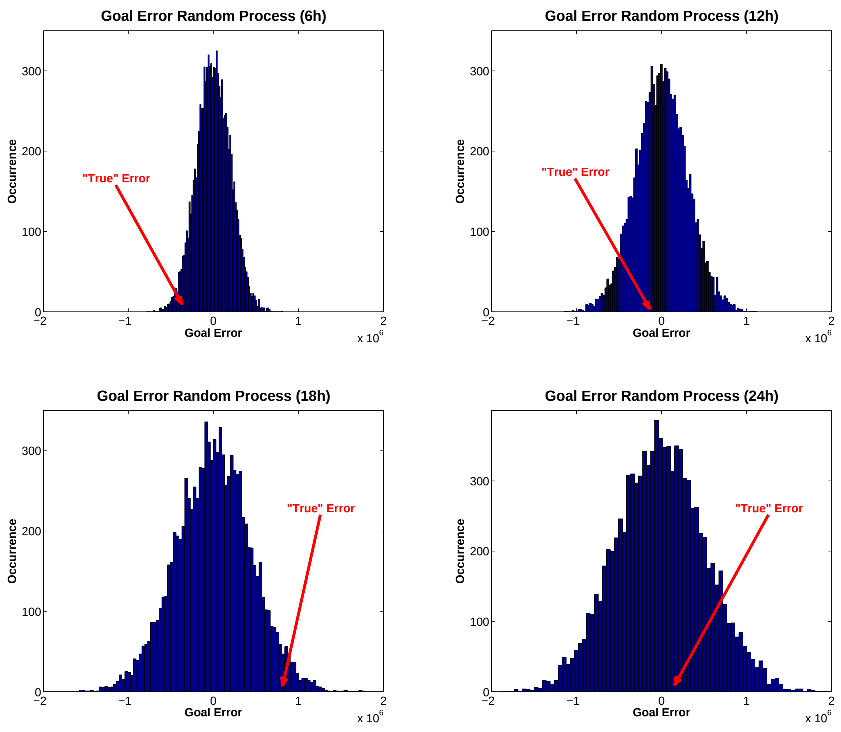


Figure 5.8: The a posteriori goal error ensemble for TC1 on resolution Δ1 after 6, 12, 18 and 24 hours for a Gaussian local error random process. The “true” error denotes the actual model error at that time step.

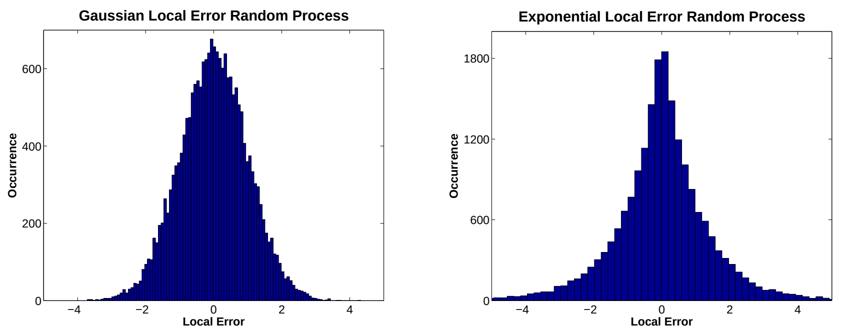


Figure 5.9: The local error distributions for resolution Δ1 for Gaussian and Exponential distributions (with increased ensemble size for better visibility).

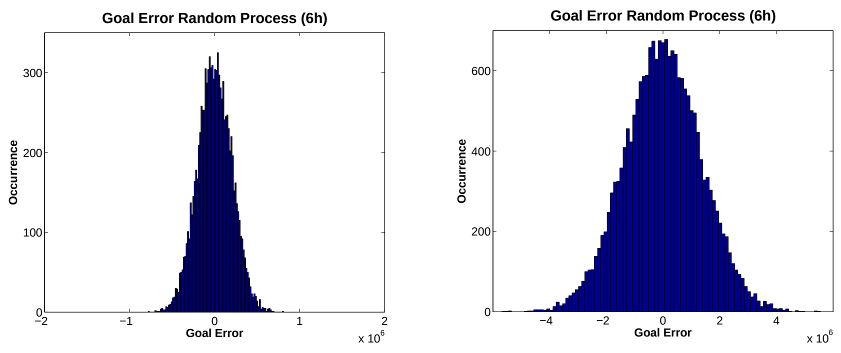


Figure 5.10: The resulting a posteriori goal error ensemble for TC1 for resolution $\Delta 1$ after 6 hours for Gaussian (left) and Exponential (right) local error random processes (with increased ensemble size for better visibility).

Spread between maximum and minimum goal approximation (TC1, J)								
$\times 10^8$	Integration time$\longrightarrow$							
Ensemble $\downarrow$	1h	3h	6h	12h	18h	24h	48	72
ICE	13	12	8	5	6	4	2	2
SPE	24	34	41	59	71	66	72	88
APE	20	35	50	50	84	62	69	90

Table 5.5: The difference between minimum and maximum goal approximation is shown for three different ensemble techniques. ICE = Initial Condition Ensemble. SP = Stochastic Physics Ensemble. APE = A Posteriori Ensemble

of the goal error distribution, we plot an approximate PDF of the goal error estimate for regional potential energy for TC1, see Figure 5.8. To clarify the structure of the underlying PDF we increase the ensemble size to 10′000 and look at approximated histograms. For Gaussian local error estimates the resulting distribution from our stochastic post-processing is also Gaussian, with increasing standard deviation in time, see Figure 5.8. This is not overly surprising, an accumulation of Gaussians random processes remains Gaussian. We compare this accumulated Gaussian after 6 hours with the goal error PDF from an exponential local error distribution in Figure 5.10. Both PDFs approximate a Gaussian distribution with different standard deviations.

Spread between maximum and minimum goal approximation (TC2, J)								
$\times 10^7$	Integration time$\longrightarrow$							
Ensemble $\downarrow$	1h	3h	6h	12h	18h	24h	48	72
ICE	18	29	23	14	12	8	-	-
SPE	47	53	98	105	149	191	-	-
APE	48	87	119	153	164	180	-	-

Table 5.6: The difference between minimum and maximum goal approximation is shown for three different ensemble techniques. ICE = Initial Condition Ensemble. SPE = Stochastic Physics Ensemble. APE = A Posteriori Ensemble

Process time forward and a posteriori ensembles			
Integration Time [h]	Process Time ICE /SPE [s]	Process Time APE [s]	Ratio
2	28	2	14
4	79	6	13.2
6	166	10	16.6
8	328	15	21.8
10	503	29	17.4
$\varnothing$			16.6

Table 5.7: The process time is shown for both forward ensembles (ICE/SPE = Initial Condition Ensemble and Stochastic Physics Ensemble) and A Posteriori Ensemble (APE) for different integration times.

Comparison of three ensemble techniques for goal J and both test cases

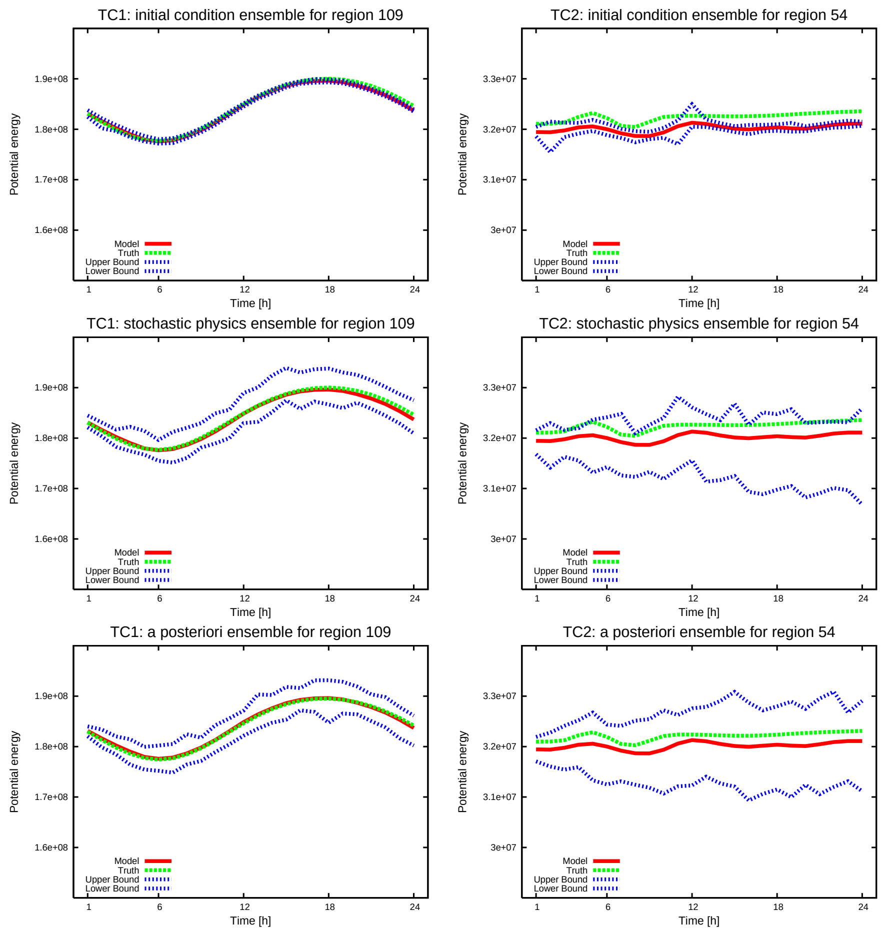


Figure 5.11: Error bounds from all three ensemble techniques for one typical goal for TC1 (left) and TC2 (right). All experiments of one test case share the same underlying local random process. Model resolution Δ1, ensemble size 100.

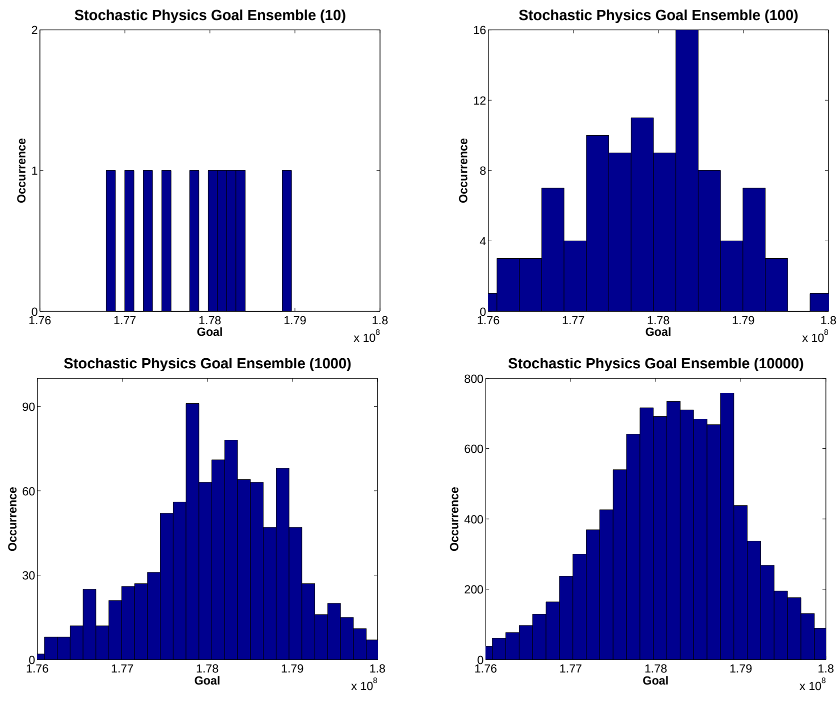


Figure 5.12: Goal error PDF after 6 hours for a stochastic physics ensemble for different ensemble sizes.

5.5.3 Forward Ensembles

As an alternative to our new a posteriori goal error ensemble we calculate two forward ensembles according to Section 5.3.5. The model is solved on the lowest ICON resolution $\Delta 1$, see Table 5.1. As a first test we use the goal J in both test cases TC1 and TC2. We plot the solutions without noise as model solution and the maximum and minimum values of the goal ensembles as error bounds, see Figure 5.11. The zero-noise curves are identical for all three ensembles (by construction). The spread in error is distinctly different for the three plots. We can identify two types of behavior. For the initial condition ensemble (ICE), the ensemble spread introduced by the initial condition perturbation remains the same for a short time and decreases for the rest of the integration period. The second type of behavior can be found for both a posteriori ensemble (APE) and stochastic physics ensembles (SPE): the ensemble spread introduced by local perturbations increases over the full integration period. To quantify the behavior, we look at the spread $\varepsilon_{max} - \varepsilon_{min}$ at different time steps in Table 5.5. The behavior for both test cases is similar: during the first 24 hours the spread of the IC ensemble is reduced to a minimum and fluctuates around this. For the APE and SPE spread, the same holds in opposite direction: during the first 24 hours the spread increases rapidly, with a very slow increase afterwards. For TC2, the similarity of APE and SPE is not as remarkable as for TC1. The size of the ensemble spread is still similar but the stochastic physics experiments show more variation, see Table 5.6. It appears that the ensemble size of 100 is not enough to guarantee a robust estimate of lower and higher bound for all time steps. To test this, we show the dependency of the resulting goal PDF on the number of ensemble members in Figure 5.12 for a 6 hour run of the stochastic physics ensemble. The number 100 for the ensemble size is chosen low, the resulting Gaussian goal error distribution is far from converged. This can explain that the lower and upper bounds are sometimes closer to the mean goal approximation than expected.

We show the computational costs for our experiments in Table 5.7. Both the initial condition ensemble and the stochastic physics ensemble are forward ensembles and need multiple model solutions. Our simple evaluation shows that forward ensembles that rely on multiple forward solutions are more expensive than our ensemble post-processing. The time needed for these forward approaches is about 18 times higher (mean ratios for 6 -10 hour runs): 100 forward runs vs 1 forward and 1 adjoint run.

5.6 Discussion

An increasing error ensemble spread means that the probability density function of the state gets less constrained by the initial condition in time, which is an expected result for approximation errors. Our a posteriori ensembles quantify the decreasing confidence

in the numerical solution in time, based on a local error learning algorithm.
We see that the increase of ensemble spread depends on the test case; the initial increase is larger for the shock-type initialization of the zonal flow against a mountain (TC2). It is hard to determine a clear cut-off criterion that exactly determines how long the error bounds are useful because of their probabilistic nature. For the solid body rotation (TC1) we obtain a reasonable error PDF for at least 60 hours. For TC2 we obtain a reasonable error PDF for at least 60 hours. These are promising results because the validity of atmospheric adjoint solutions becomes also questionable at time frames longer than a few days.
However, the applicability of the method is challenged if the flow faces regime changes because the characteristic properties of the local error random process change in the case of a flow regime change, as has been shown for the initial states of TC1 and TC2.

The ensemble spread behavior of APE and SPE ensemble techniques look remarkably similar, qualitatively and quantitatively. Both are conceptually closely connected. APE can be described as a linearized, simplified version of SPE, derived from only one model solution. Stochastic physics ensemble are usually initialized with stochastic processes that are either based on the study of physical processes (Majda and Stechmann 2009) or on empirical studies of the shortcomings of the underlying model (Seiffert et al. 2006) Both methods do look at total model error spread, not focussing on approximation error. We can conclude that a) stochastic physics ensembles can be used for goal approximation estimation when combined with our local learning algorithm and b) our a posteriori ensembles are - at least for linear test cases - a viable alternative to stochastic physics ensembles.
The ensemble spread behavior of initial condition ensembles shows that ICE are not suited to estimate goal approximation error, even if equipped with our local learning algorithm. The inherent tendency to decrease variance is a fatal property for an approximation error estimation algorithm. This result is known in a different context (total model error) for initial condition ensembles as variance deflation. The concept of covariance inflation has been introduced to combat this problem (Anderson and Anderson 1999). It is not clear how to implement a similar technique in single time frame. However, it is interesting to compare our local error initialization with classical initialization methods for GFD initial condition ensembles. Initial condition ensembles are routinely initiated by time-lag initializations, using instances of previous time steps or different instances of a long control run to initialize the model run (Jungclaus et al. 2010). The concept of optimal perturbations is a more sophisticated way to create an initial condition ensemble. These perturbations maximize error growth from a tangent-linear error evolution point of view (Ehrendorfer 1997). The optimal perturbation concept can be seen as a worst-case error-growth scenario for the first time step. It is as unrealistic as the time-lag initialization to get a useful ensemble spread of approximation error. It is possible though to connect our method to the optimal perturbation point of view: our adjoint solution includes the tangential directions of error growth for the full run, not only the directions of maximum growth for the first time step. This means that ensemble post-processing propagates all random errors, not on random trajectories (time-lag initialization) or worst-case-scenarios (optimum perturbations) but on the “correct” trajectory.

A posteriori ensembles are computationally cheap and produce similar results as stochastic physics ensembles for our test cases. Can they replace stochastic physics ensembles? The answer depends on the experimental setting. The computational expense is only in favor of a posteriori ensembles if the number of goals (here = 1) is small compared to the number of ensemble realizations(here = 100). For higher numbers of goals, the additional cost of the adjoint is larger than the additional cost caused by a forward ensemble: we need one adjoint solution per goal, while the cost of the forward ensemble is independent of the number of goals. At the same time the computational expense of the a posteriori ensemble is practically independent of the number of ensemble members while the cost for both forward ensembles scales linearly with the number of ensemble members. This is an important property because the ensemble size is decisive for the quality of the goal PDF approximation. The ensemble size is also very important if we look at one major difference between the ensemble techniques: how they deal with the nonlinearity of the model. A posteriori ensembles cannot quantify errors induced by nonlinearity because they rely on one model solution and linear perturbations around it. Forward ensembles can - but only if the ensemble size is large enough. A posteriori ensembles can have a large ensemble size but do not capture nonlinearity while stochastic physics ensembles capture nonlinearity but are limited to small ensemble sizes.

5.7 Goal Error Ensembles and the Central Limit Theorem

We have shown in Section 5.5.2 that a posteriori goal errors are approximatelye normally distributed even for non-normal distributions of local errors. This behavior can be explained with the Central Limit Theorem[1] (CLT). The most common variant of a CLT (e.g., von Storch and Zwiers 1999) states that the average distribution of a sum of independent random variables is asymptotically normally distributed, regardless of the distributions of the respective random variables.

If $\mathbf{X}_k$, $k = 1, 2, ...$ is an infinite series of independent and identically distributed random variables with $\mathcal{E}(\mathbf{X}_k) = \mu$ and $Var(\mathbf{X}_k) = \sigma^2$ then the average $\frac{1}{n}\sum_{k=1}^{n} \mathbf{X}_k$ is asymptotically normally distributed. That is

$$\lim_{n\to\infty} \frac{\frac{1}{n}\sum_{k=1}^{n}(\mathbf{X}_k - \mu)}{1/\sqrt{n}\sigma} \sim \mathcal{N}(0,1) \tag{5.33}$$

We can interpret the spatial scalar product $\langle ., . \rangle_\Omega$ between adjoint solution $\mathbf{q}_\Delta^{*k}$ at time step k, $k = 1, ..., N$, and local random process $\mathcal{P}(\tilde{\mathbf{p}})$ at the same time step as a random variable $\mathbf{Y}_k$

$$\mathbf{Y}_k := \left\langle \mathbf{q}_\Delta^{*k}, \frac{1}{\Delta t}\mathcal{P}(\tilde{\mathbf{p}}) \right\rangle_\Omega . \tag{5.34}$$

[1] There are a multitude of Central Limit Theorems, each one describing the conditions for cumulative random processes that are necessary to lead to a Normal distribution. The first version of a CLT was formulated by de Moivre in 1733. A historical overview can be found in (Cam 1986). The term “Zentraler Grenzwertsatz” was first used in (Polya 1920).

The independent random variables $\mathbf{Y}_k$ are based on the same underlying random process $\mathcal{N}$ and have finite means and variances. They do not have constant values for mean and variance so we cannot simply use (5.33) to derive a Normal distribution with zero mean an standard deviation $\sigma = 1$. We can still argue that for many time steps it appears reasonable that the distribution of $\sum_{k=1,..,N} \mathbf{Y}_k$ is asymptotically Normally distributed. It is surprising to see in Figure 5.10 that the tendency to become Normally distributed is already so pronounced after 6 hours because the adjoint solution $\mathbf{q}_\triangle^{*k}$ masks out many elements of the random processes for the last time steps (the adjoint variable is zero at time step $k = N$ everywhere, except on the region of the goal). A posteriori ensembles always yield Normal distributions with a zero mean which in turn means a symmetric error distribution around a systematically wrong goal approximation $J_\triangle$ and no indication of asymmetric model biases. This is a consequence of our symmetric Gaussian local error random processes. Local error random processes with memory or dependent realizations of random processes can lead to non-Normal distributions. The general conclusion is: goal error distributions with non-zero mean imply that not all local errors can be described by random processes without memory. This fits very well with the Mori Zwanzig formalism introduced in Section 5.3: our method estimates the part of the local errors that is produced by unresolved processes that can be modelled purely stochastically. The method cannot estimate the consequence of the local error processes with memory and the deterministic approximation errors. The error that is introduced by the deterministic influence of the unresolved processes and the memory effect can become larger at some point than the stochastic noise. In that case our stochastic assessment of the uncertainty of the numerical solution has to be extended to include asymmetries and memory effects or be augmented by deterministic methods similar to those proposed in Chapter 3.
The same argument holds for simple stochastic physics ensembles: if the stochastic perturbation distribution is symmetric, with constant mean and without memory, the resulting ensemble will be Normal. There are literature results that show that stochastic perturbations can affect the mean state (Shutts 2005). These stochastic perturbations either are spatially and temporally correlated (cellular automata) or are inserted into nonlinear functions that result in effectively skewed distributions, which in the end lead to model biases. The results support our conclusion that the principle of the Central Limit Theorem effectively limits the effect of symmetric independent random perturbations to a Gaussian blurring of the model approximation.

5.8 Conclusion and Outlook

A posteriori goal ensembles are a simple way to obtain an estimate of goal approximation uncertainty. The concept of learning to specify a local error estimator is closely related to the deterministic concept of Section 5.3. The stochastic description of local errors allows us to create a much simpler and more robust learning algorithm. A posteriori goal ensembles are unique compared to other ensembles because they are produced from only one forward evaluation of the model. We believe that the influence and structure of local error processes on resulting goal error probability distributions should be investigated further.

It is possible to extend a posterior goal ensembles to total model error. We have determined our local error random process $\mathcal{P}$ for goal approximation errors by comparing different solutions of the numerical model. The counterpart for total model error is to compare the model solution with measurements to train a new local error random process $\mathcal{P}_2$. However, this is problematic: models usually do not start from the identical same state as reality. In contrast, the initialization of GFD models is complex and involves complicated routines that infer optimal initial states from given data. We suggest to approximate local error realizations as the differences between the tendencies of the model for the first few time steps and the tendencies as derived from real data. If the method that is used to derive the discrete initial conditions from data is not a variant of 4DVAR the first time steps of the model might not be representative for the error evolution because the model will experience initialization shock and drift. In this case, we suggest to use the last few time steps of the assimilation time window to do the learning. For complex GFD models, approximation and total model error tend to overlap because the underlying models do not converge in a classical sense. Our method may be an option to separate the two by choosing a different local reference truth. To quantify total model error, the exact choice of local reference truth requires further reserch.

It is also possible to use a posteriori goal ensembles to quantify the predictability of the numerical solution. The increasing spread of the a posteriori ensemble means that the initial data constraint gets weaker and the system approaches the equilibrium background statistics. Most physically relevant quantities in equilibrium or forced-dissipative systems without trend are bounded and the time scales of usability for our method are sufficiently small to assume a system in quasi-equilibrium. This means we can identify properties of the system through equilibrium statistics as mean μ_{prior} and standard deviation σ_{prior}. Predictability means that knowledge of initial data leads to results that are better then the properties of the equilibrium statistics. We can quantify predictability as a function of the standard deviation σ of the goal error distributions and the background standard deviation σ_{prior}. We argue that predictions of the numerical system stop making sense if $\sigma > \sigma_{prior}$. If a system is in its mean state and its numerical uncertainty σ is equal to the background standard deviation σ_{prior} the probability of the numerical solution to be in any state of the system is identical to the background distribution. While ensemble methods do not command a specific relationship between σ and σ_{prior} they support the ability to construct a consistent concept of numerical predictability.

Our algorithm is a new combination of stochastic ideas and numerical methods and enables goal error estimation for any GFD model with AD abilities.

Summary of Chapter 5

- We extend the method of dual weight error estimation to a stochastic description of local model error. This leads to our concept of a posteriori goal error ensembles for relevant physical quantities, calculated from a single model solution.
- We present a learning algorithm for the local error random process that uses local differences between solutions on different resolutions.
- The algorithm is evaluated for a shallow water model and two test cases and shows consistently good results.
- We use the learned local error random process to a) perturb initial conditions for ICOSWM and to b) perturb the model formulation of ICOSWM and obtain two forward ensembles of goal approximations.
- We examine the consequences of the Central Limit Theorem for goal error distributions.

Chapter 6

Conclusions and Outlook

6.1 The Quintessence

To our best knowledge – for the spherical shallow water equations or any global model of Geophysical Fluid Dynamics – we are the first

- to estimate deterministic time-dependent goal approximation errors with empirical local error estimators.
- to estimate stochastic time-dependent a posteriori goal ensembles from a single model evaluation.
- to employ methods of algorithmic learning for (automatic) goal approximation error estimation.
- to use Algorithmic Differentiation tools to obtain the required sensitivities for dual weight error estimation for error correction purposes.

We have created an algorithm that estimates goal approximation error as an aggregation of local model errors. Local model errors can be described deterministically and stochastically. We have presented algorithms that learn the properties of these descriptions for a given model and flow. The concept of local model error learning can also be used to equip stochastic physics ensembles with information on local model error. Our algorithm is sufficiently general to be extended to total model error in the future.

Goal error estimation through learning combines deterministic numerical methods with probabilistic approaches. Our algorithm is an important step towards automatic full error bars for GFD models by creating error bars for approximation error without user knowledge of the model's discretization. A posteriori ensembles that are derived from a single model solution can – if used carefully – challenge classical forward ensembles.

6.2 The Answers to the Research Questions

The guiding research questions (Section 2.4) are divided into two categories, following the two possible interpretations of local model errors (Section 2.2).

1. Deterministic Error Correction of Goal Approximation Errors for GFD Models

- Can empirical functionals of the flow state be used for the estimation of goal approximation errors?

Yes. We have successfully used smoothness measures with a scaling weight to estimate approximation errors for regional potential energy with ICOSWM.

- How can the algorithm learn to train these functionals?

We use a learning period of one time step to calculate goal approximations on two different resolutions. We determine the scaling factor of the empirical local error estimators by comparing the error estimate with the difference between these goal approximations.

- Is the parameter set of these functionals flow-regime-dependent?

Yes. The scaling factor is flow-regime dependent. In case of topographic inhomogeneity the learning should be done in regions that exhibit similar flow states as the target region.

- Is the parameter set of these functionals goal-dependent?

No. For goals that depend on the same parts of the state vector the sensitivities "handle" the specification of the goal.

- Is the parameter set of these functionals resolution-dependent?

Yes. For different resolutions we have to re-use our learning algorithm.

- How do we obtain the sensitivities automatically and efficiently?

We have constructed an differentiation-enabled version of ICOSWM.

- How long are the error estimates of our algorithm useful?

For our solid body rotation test case, we can track the error evolution well for 24 hours. Over that period we can improve the approximated goals more than 50%. For the zonal flow against a mountain, we can track the error for the length of the initial perturbation, around 12 hours.

2. Stochastic Uncertainty Quantification of Goal Approximation Errors

- Can a local error random process $\mathcal{P}$ be used for the estimation of goal approximation errors?

Yes. We have successfully used local error random processes with a learned standard deviation to estimate approximation errors for regional potential energy with ICOSWM.

- How can the algorithm learn the properties of the correct stochastic process?

We look at solution differences at a grid-cell level to calculate properties of an error distribution (e.g., mean and variance for a Gaussian distribution).

- Is the stochastic process flow-regime dependent?

Yes. The dependency is weaker compared to the deterministic learning attempts. Results using the parameter set of the solid body rotation are useful for the zonal flow against the mountain and vice versa.

- Is the stochastic process goal-dependent?

No. The results are similar as for the deterministic approach. The automatic sensitivities take care of the exact specification of the goal.

- Is the stochastic process resolution-dependent?

Yes. With increasing resolution the standard deviation of the local error random process decreases.

- How long is the goal error ensemble of our algorithm useful?

The spread of our error ensembles increases in time, so it seems that we can theoretically estimate the errors infinitely. When the error bounds become bigger than the natural signal, these error bounds are not too useful anymore. For both analyzed test cases we estimate the reasonable time frame to be at least 60 hours.

- Can we use the local error learning algorithm to use classical ensembles to estimate goal approximation error?

Yes. We show that it is possible to use a stochastic physics ensemble to estimate goal approximation error. The results are very similar to our a posteriori goals. It is not possible to use initial condition ensembles to estimate goal approximation error because of the inherent tendency to decrease ensemble spread.

- How does the computational cost of a posteriori goal ensembles compare to that of a stochastic physics forward ensemble?

The a posteriori ensemble is much cheaper to obtain than a classical stochastic physics ensemble (for a large number of ensemble members and few goals).

6.3 The Correct Interpretation of Local Model Errors

We have suggested two strategies for goal error estimation based on two different descriptions of local model errors, stochastic and deterministic. Which of the two is the "correct" description? The answer is simple: local model errors are the consequence of both local error random processes and local errors as a deterministic functional of the solution.
We have shown in Section 3.4 that local error production is spatially correlated and structurally similar to the underlying flow. We have also argued in Section 5.6 that structural model biases imply a local error process with either deterministic or memory component. This holds also for stochastic physics experiments that exhibit similar properties as our a posteriori ensembles. At the same time, the results in Section 5.5 show that a local error random process is an efficient tool to quantify decreasing confidence in the numerical solution, consistent with local model properties. This growing uncertainty covers and blurs the potential model bias but ultimately cannot fully compensate for the deterministic effects. The need for both descriptions is in perfect agreement with the Mori Zwanzig formalism in Section 2.2.

We conclude that the stochastic description provides efficient error estimates consistent with learned properties of the model that also cover deterministic effects for short runs. We encourage the usage of a posteriori ensembles because they are a simple and cheap way to quantify the numerical uncertainty that can be deduced from local model properties. The direction of any systematic bias, however, cannot be learned from this simple stochastic approach. For longer runs with distinct model bias, we need the deterministic interpretation of local errors or more complex local random processes with memory.

6.4 The Next Steps

There is one natural technical extension of the work presented in this thesis: the continued evaluation of the algorithm for different models, different goals, and more demanding test cases.
There are also a lot of conceptual extensions of this work. For the deterministic approach: the smoothness measures of Section 3.3.3 can be refined, extended and combined. Additionally, the learning algorithm for the deterministic algorithm seems to be insufficient. The stochastic learning concept brought forward in Section 5.3.3 could be applied to the deterministic estimators, too. To do this, one should determine random process properties as mean and variance for the smoothness measures themselves. By comparing the random process properties to the learned local random process properties the parameters of the deterministic estimators could be robustly determined. That

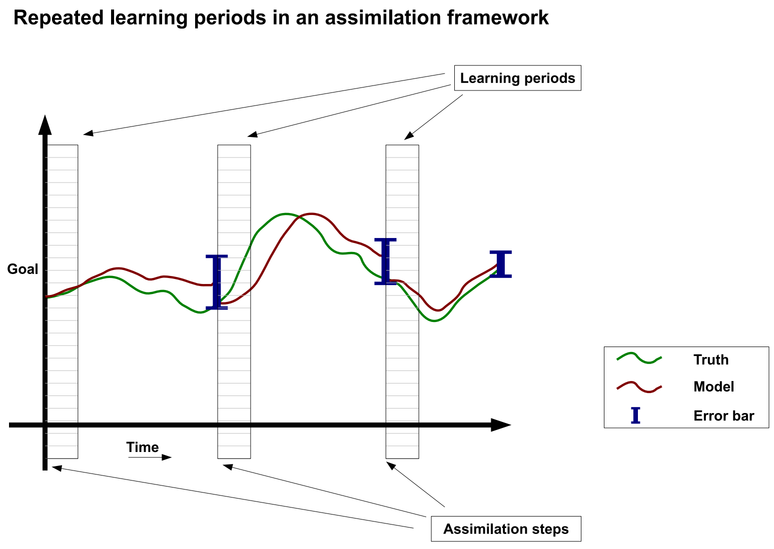


Figure 6.1: A sketch of a working environment for this algorithm. The properties of the error estimation technique are routinely gauged at the beginning of forecast time frames. At the end of the forecast period, an error estimate is calculated for important physical quantities (goals).

method could potentially be more stable than the original learning algorithm because it uses local information. For the stochastic approach: the type of random process should be investigated more closely. Knowledge on topography and possible flow state should be incorporated into the specification of the local model error random process. The effect of memory and asymmetry is potentially large to enable a posteriori ensembles with goal error distributions that are not Normal.
For both parts: our error algorithm depends on a solution of the goal-dependent adjoint problem. The construction of simplified adjoints could help to reduce the computational cost of this method enormously. We believe that simplified low order adjoint solutions could be used to estimate the influence of local errors on the goal error (Hinze and Volkwein 2005).

To apply our algorithm to real world problems we suggest a more elaborate framework. Given a data assimilation system that initializes the model at the beginning of recurring assimilation windows, we propose a similar error estimation window approach, see Figure 6.4. At the beginning of each time window, our learning algorithm determines the flow specific local error parameter set for a given local error random process / empirical functional. For that type of application, the stochastic approach is more promising because of its superior learning algorithm. The time window approach for error estimation leads to suitable approximation error estimates for the respective time frames. If the learned properties differ strongly between two time windows this is an indication of the current quality of the goal error approximations. In this combined data assimilation and error estimation environment, our local error learning algorithm can quantify approximation uncertainties for important physical quantities of interest. Our algorithm could also be used to produce estimates of the total model error uncertainty if the difficulties of defining a correct local reference truth for total model error can be overcome. The possibility to systematically use a posteriori ensemble spread to determine limits of predictability should be investigated further.

Algorithm Proposal for Error Estimation Time Window Approach

Define a class of random processes $\mathcal{P}(\mathbf{p})$ that can describe the distribution of local (total) model errors.

At each time frame:

1. Initialize model with data assimilation scheme.
2. Determine the specific properties $\tilde{\mathbf{p}}$ of the random process $\mathcal{P}$ for either local model error or local total model error in a short learning period.
3. Use the local error random process $\mathcal{P}(\tilde{\mathbf{p}})$ as perturbation random process to create goal approximation ensembles for relevant goals.

6.5 Concluding Remarks

The increasing complexity in computational models can only efficiently be countered if the algorithms themselves become smarter. Our method of goal error estimation through learning is a new approach to uncertainty quantification in GFD models. The algorithm is simple, easy to extend and – comparatively – straightforward to apply to new models. It offers a new perspective on the connection between ensemble techniques and numerical error. Goal error estimation through learning is a first step towards the necessary ability of GFD models to automatically generate error bars for every calculated physical quantity.

Appendix A

The Development of the Differentiation-Enabled Shallow Water Model ICOSWM-AD

We introduce the concept of Algorithmic Differentiation (AD). We describe the two conceptual modes of AD, forward and reverse mode, and the two possible technical approaches to AD, source code transformation and operator overloading. The development of a differentiation-enabled shallow water version ICOSWM is one step towards a differentiation-enabled version of the 3D general circulation model ICON. For the development of a differentiation-enabled shallow water version ICOSWM-AD we implement reverse mode AD with an operator overloading approach. We motivate the use of computational graphs to understand the concept of AD. We show a computational graph of ICOSWM and a corresponding computational graph of reverse mode ICOSWM. We introduce our version of memory checkpointing.

The Concept: Algorithmic Differentiation

A growing number of Earth system model applications need high dimensional gradients due to several reasons. First, the dependency of key quantities on a set of parameters is needed during the construction process of a model to construct realistic models. Second, the dependency of cost/distance functions on controls variables (initial condition or forcings) is needed to optimize the model for forecast-type applications. Automatic differentiation is an algorithmic concept that allows us to evaluate the gradients of any function specified by computational programs (code) with respect to any control variable within this program (if the derivatives exist). Two other options for computing gradients are symbolic derivatives and numerical derivatives by the method of finite differences. A full blown GFD application can not be differentiated in a symbolic way. Algorithmic Differentiation is also superior to the standard numerical method of evaluating derivatives (the method of finite differences) because it deals better with truncation error. For these reasons AD tools and GFD models have been developed concertedly since the 1990s (Marotzke et al. 1999).

The underlying principle of Algorithmic Differentiation is the consequent usage of the chain rule for all computational operations. Every computational program - complex as it might be - is a sequence of simple elemental operations E. For most of those elemental operations the exact (symbolic) derivative is known. We can interpret GFD models as a the application of an operator F on a control vector $\mathbf{q}^0$, resulting in a final state $\mathbf{q}^N$ (N time steps)

$$\mathbf{q}^N = F(\mathbf{q}^0). \tag{A.1}$$

For algebraic simplicity we assume that the control $\mathbf{q}^0$ acts in the first time step (initial condition control) but the same derivation can also be done if the control acts on each time step. The solution operator F for GFD models is usually iterative, it solves a new state vector $\mathbf{q}^{i+1}$ from a state vector $\mathbf{q}^i$ as

$$\mathbf{q}^{i+1} = F^i(\mathbf{q}^1). \tag{A.2}$$

The operator F is therefore a concatenation of operators F^i

$$F = F^N \circ F^{N-1} \circ ... \circ F^2 \circ F^1 \tag{A.3}$$

Each time step operator F^i is again a concatenation of the aforementioned elemental operators E. Given M elemental operations per time step this allows to write

$$F^i = E^M \circ E^{M-1} \circ ... \circ E^2 \circ E^1. \tag{A.4}$$

If we now create a cost function of the solution $\mathbf{q} = (\mathbf{q}^0, ..., \mathbf{q}^N)$ we can rewrite this function also symbolically as a function of the initial state vector $\mathbf{q}^0$

$$J(\mathbf{q}) = H(\mathbf{q}^0). \tag{A.5}$$

To optimize J most numerical routines need the derivative $\dfrac{dJ}{d\mathbf{q}_0}$. We write the derivative as chain rule to calculate the derivative of J that incorporates all time step operators F^i

$$\begin{aligned} \frac{dJ}{d\mathbf{q}^0} &= \frac{dH}{d\mathbf{q}^N} \cdot \frac{d\mathbf{q}^N}{d\mathbf{q}^{N-1}} \cdot ... \cdot \frac{d\mathbf{q}^1}{d\mathbf{q}^0} && \text{(A.6)} \\ &= \frac{dH}{d\mathbf{q}^N} \cdot \frac{dF^{N-1}}{d\mathbf{q}^{N-1}} \cdot ... \frac{dF^0}{d\mathbf{q}^0}. && \text{(A.7)} \end{aligned}$$

We write the ith derivatives as chain rule of the elemental operations E

$$\frac{dF^i}{d\mathbf{q}^i} = \frac{dF^i}{dE^M} \cdot \frac{dE^M}{dE^{M-1}} \cdot ... \cdot \frac{dE^1}{d\mathbf{q}^i}. \tag{A.8}$$

We summarize: the numerical evaluation of a cost function J can be split up into single elemental operations. The derivative of such a cost function with respect to a control vector $\mathbf{q}^0$ can be calculated through the chain rule of derivatives of each elemental operation. Algorithmic Differentiation tools interpret model code to calculate this derivative of cost functions automatically and exact up to machine precision.
There are two major conceptual modes of Algorithmic Differentiation: tangent linear mode (forward) and adjoint mode (reverse/backward). Tangent linear mode propagates initial derivatives through the chain rule from right to left, i.e., from the beginning of a computation to the end. Tangent linear mode is useful if large numbers of outputs are dependent on a low number of controls because each dimension of control needs one forward run. Reverse mode propagates initial derivatives through the chain rule from left to right, i.e., from the end of a computation to the beginning. Reverse mode is useful if low numbers of outputs are dependent on a large number of controls because each dimension of outputs needs one backward run.
There are also two major technical modes of Algorithmic Differentiation: source code transformation (SCT) and operator overloading (OO). Source code transformation produces new source code that still incorporates the original model evaluation and at the same time the necessary code for derivative propagation. The advantage of this process is the possibility to optimize the AD code afterwards by hand. Operator overloading means that we introduce "new" real numbers and elementary mathematical operations to also calculate derivatives. Operator overloading is easy to implement and much more robust but – at the same time – it is much harder to optimize.

To understand AD it is useful to think of a computational program as a realization of a graph: variables and operations are vertices, dependencies are represented as directed edges. This concept can be applied to any computational program, no matter its inherent complexity. Automatic differentiation transforms the elements of a computational graph but keeps the structure of the graph. The nodes for real numbers are replaced by nodes that also include the adjoint / tangent component. The edges are augmented with local partial derivatives with respect to local dependencies. The two modes of Algorithmic Differentiation are now much easier to understand: forward mode means that we propagate tangents along with original values through the computational graph. Reverse mode means we propagate through the graph once for the original components and then once backwards for the adjoint components. More information on Automatic differentiation can be found in books (e.g., Griewank 2000; Bischof et al. 2008), on a central webpage [1] or various survey papers (e.g., Bischof et al. 2002).

[1]http://www.autodiff.org

	Technical mode $\longrightarrow$	
Conceptual $\downarrow$	Source code transformation	Operator Overloading
Forward	not planned	ICOSWM-AD
Reverse	*planned for ICON 3D Ocean*	**ICOSWM-AD** & *planned for ICON 3D Ocean*

Table A.1: An overview of the four possible AD strategies and its (planned) realizations within the ICON framework. The version that is used throughout this thesis is in bold.

GFD applications are usually interested in reverse mode because for typical data assimilation applications the dimension of the output is one (the cost function) and the number of controls large (e.g., dimensionality of discrete initial condition vector). The same holds for the error estimation algorithm of this thesis: we need the sensitivities of a few important goals with respect to a high dimensional control vector.

The Tool: the Differentiation-Enabled NAG Fortran Compiler

The development of the differentiation-enabled version of ICOSWM for this thesis was done in cooperation with the RWTH Aachen university and the CompAD project. The mission statement of CompAD is “to put Algorithmic Differentiation into the NAG-Ware Fortran compiler”, for more details see the project's webpage[2].
The differentiation-enabled NAG Fortran compiler combines a two stage semantical transformation with a set of runtime support libraries in a hybrid approach to AD that blends source transformation capabilities and overloading techniques. More details can be found in (Naumann and Riehme 2006).
Practically, the tool consists of two pieces, the compiler and modules. There are modules for each AD mode (i.e., forward / reverse, SCT/OO). The compiler takes care of operator overloading / overloaded data types. The module - if linked into your project - provides a set of routines that can be used to determine which variable should be differentiated with respect to which. Every real variable in the code is transformed into a variable of type CompAD-type automatically by the compiler. The specific structure of this CompAD-type depends on the chosen AD-mode. The changed variables include the information about the propagated gradients. These derivative components can be accessed via routines supplied by the module.

[2]http://wiki.stce.rwth-aachen.de/twiki/bin/view/Projects/CompAD/WebHome

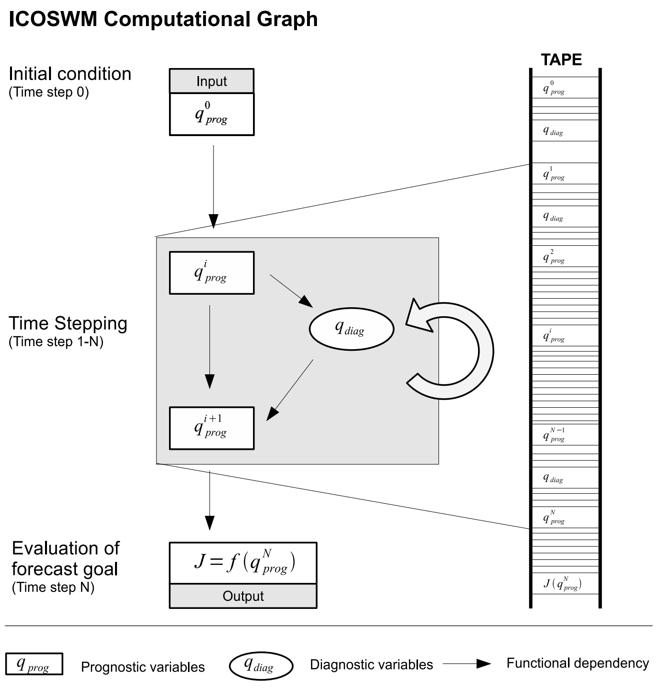


Figure A.1: A sketch of the computational graph of ICOSWM. The left part of this figure shows standard forward evaluation. The right part introduces the concept of a TAPE that records every operation throughout the program.

The Model: ICOSWM

The shallow water equations on the sphere are

$$\frac{\partial \mathbf{v}}{\partial t} = (\xi + f)\mathbf{k} \times \mathbf{v} - \nabla(gh + \frac{1}{2}|\mathbf{v}|^2) \qquad \text{(A.9)}$$
$$\frac{\partial h}{\partial t} + \nabla \cdot (h\mathbf{v}) = 0.$$

Here $\mathbf{v}$ is the horizontal velocity, ξ the vorticity, f the Coriolis parameter, $g = 9.81$ the gravitational constant and h the height of the fluid surface. The initial conditions are $\mathbf{v}(t_0) = \mathbf{v}_0$ and $h(t_0) = h_0$. We consider (3.1) on a time interval $T := [t_0, t_n]$ and with periodic spatial boundary conditions. The state vector $\mathbf{q} = (h, \mathbf{v})$ consists of the prognostic fields height and velocity.
Our numerical framework is ICOSWM[3], a shallow water model on a triangular spherical grid with C-type staggering of the variables. ICOSWM uses a hybrid finite volume / finite difference method to approximate the SWE (3.1). ICOSWM calculates a solution

[3]http://www.icon.enes.org/swm/index.html

vector $\mathbf{q}_{prog} = (h, v_n)$ of prognostic variables h, the height field in the cell centres of our triangular grid, and v_n, the normal velocities at the mid points of the triangular edges. The solution process is sequential in nature, the discrete model yields discrete time slices $\mathbf{q}^n_{prog}$ for each time step. The time stepping schemes include Runge-Kutta, Semi-Implicit and Adam Bashford. For further details see (Giorgetta et al. 2009; Ripodas et al. 2009).
The starting version of ICOSWM is ICOSWM 1.05[4]. The finalized version is checked into the ZMAW SVN system as

http://svn.zmaw.de/svn/icon/branches/icon-1.0.5_AD/

ICOSWM produces the state vector $\mathbf{q}_\Delta = (h_\Delta, \mathbf{v_n})$ that consists of the prognostic fields height and velocity. We now introduce the concept of *prognostic* and *diagnostic* variables that is very common to GFD models. The state vector $\mathbf{q}_\Delta = \mathbf{q}_{prog}$ consists of prognostic variables. The program produces at each time step diagnostic variables $\mathbf{q}_{diag}$ that are derived quantities such as kinetic energy, vorticity or reconstructed zonal and meridional velocities. These diagnostic variables are only intermediate steps for the routines, they can be calculated from the prognostic variables at each time step. The concept is used for conceptual simplicity during the programming. If we account for both types of variables the main structure of ICOSWM is the following (see Figure A.1):

1. **Initialization**: the prognostic variables h and v_n are initialized from external data or analytic functions at the beginning.

2. **Time stepping**: ICOSWM computes a set of so-called diagnostic variables which also represent physical quantities such as kinetic energy, vorticity or velocities in geographical coordinates at the cell center. The time stepping can also be separated into the two components height field and velocities as has been shown in chapter 4:

   - Implicit step: calculate new surface height h^{i+1} by solving linear free surface equation

   $$A^i(h^i, v^i) \cdot h^{i+1} = b^i(h^i, v^i) \tag{A.10}$$

   - Explicit step: update velocity

   $$v^{i+1} = h(h^{i+1}, v^i) \tag{A.11}$$

   The code of ICOSWM is structured in a similar two step way per time step. First, it calculates a new set of diagnostic variables. Second, it uses the diagnostic variables and the prognostic variables of time step i to calculate the new prognostic

[4]http://svn.zmaw.de/svn/icon/branches/icon-1.0.5_fr/

variables at time step $i+1$.

$$\mathbf{q}_{prog}^{i+1} = f(\mathbf{q}_{prog}^{i}, \mathbf{q}_{diag}), \tag{A.12}$$

with f a combination of the implicit and explicit step. This procedure is repeated for all time steps.

3. **Post-processing**: We calculate a goal approximation from the solution of the last time step, $\mathbf{q}_{prog}^{N}$. After the last time step our error algorithm sets in and tries to estimate the error of the approximated goal.

ICOSWM does not save all intermediate time steps. The model uses two instances of the prognostic variables $\mathbf{q}_{prog}$ and one instance of the diagnostic variable $\mathbf{q}_{diag}$ and exchanges the two prognostic instances each time step. This is a standard procedure for GFD models because the memory requirement to save all intermediate time steps is much too high (and mass storage devices are much too slow). At the same time, this is a significant problem for reverse mode AD because reverse mode AD requires the full forward solution for the backward run.

The Implementation of ICOSWM-AD

We construct a simplified reverse mode diagram of ICOSWM (Figure A.2). The NAGware compiler uses the concept of a TAPE to realize reverse mode AD. The TAPE is a recorder that records any variable instance and any action that occurs throughout the run of a computational model. The TAPE saves the complete computational graph. It is activated at the beginning of program, stopped at the end and interpreted to go backward through the computational graph. If the TAPE is activated we speak of an active forward run, if the TAPE is deactivated we speak of a passive forward run. A conceptual version of the ICOSWM Fortran program with original syntax for the TAPE commands looks like:

Forward model run

```
CALL TAPE_INIT
CALL TAPE_TURN_ON
CALL ICON_INIT(controls)
CALL ICON_TIMELOOP
CALL GOAL_CALCULATION (goal)
CALL TAPE_TURN_OFF
```

Backward model run

```
CALL SEED (goal,1)
CALL TAPE_INTERPRETER
derivatives = ACCESS_DERIV (controls)
```

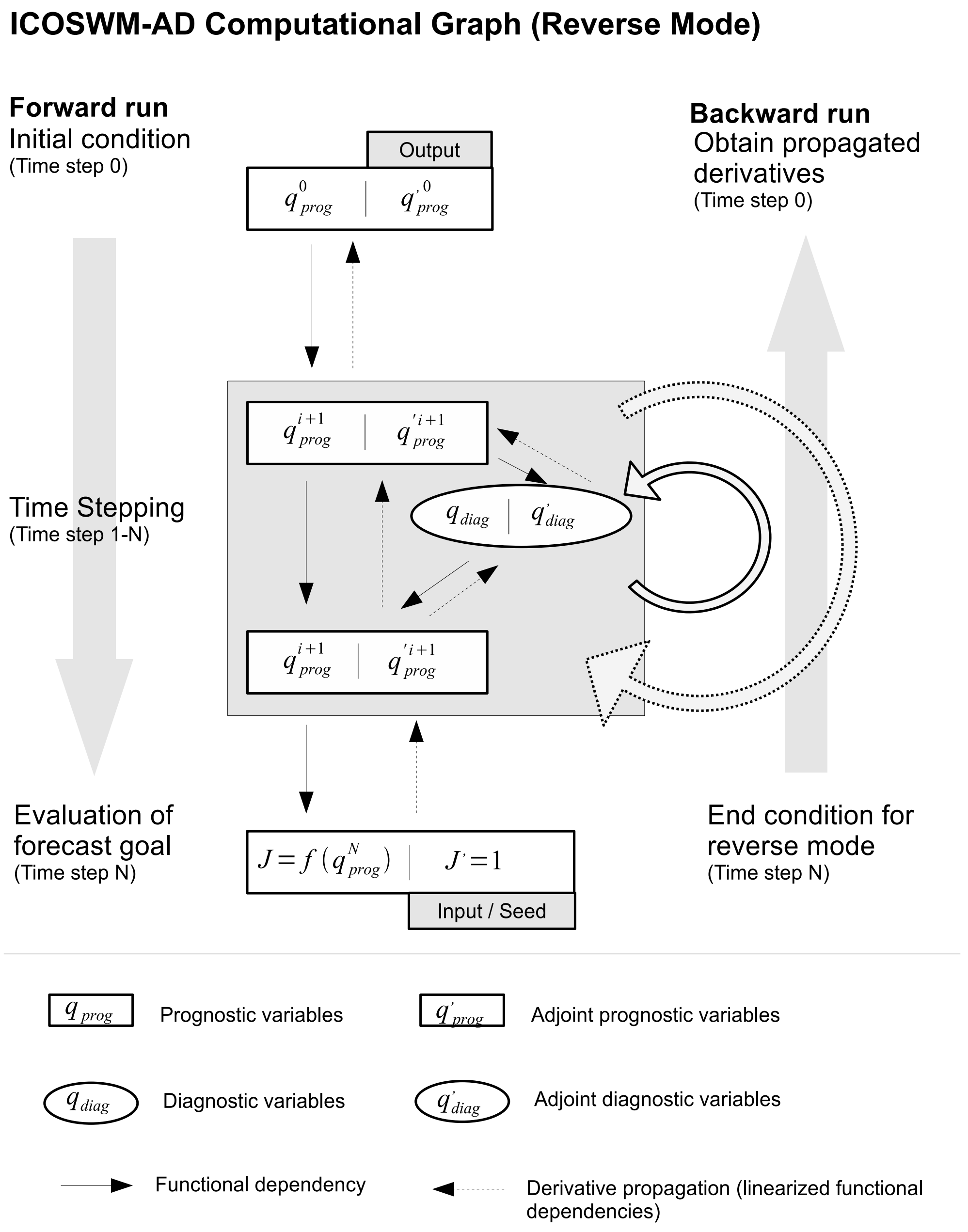

ICOSWM-AD Computational Graph (Reverse Mode)
Forward run
Initial condition
(Time step 0)
Output
q^0_{prog} | q'^0_{prog}
Backward run
Obtain propagated derivatives
(Time step 0)
q^{i+1}_{prog} | q'^{i+1}_{prog}
q_{diag} | q'_{diag}
Time Stepping
(Time step 1-N)
Evaluation of forecast goal
(Time step N)
$J=f(q^N_{prog})$ | $J'=1$
Input / Seed
End condition for reverse mode
(Time step N)
Prognostic variables
Adjoint prognostic variables
Diagnostic variables
Adjoint diagnostic variables
Functional dependency
Derivative propagation (linearized functional dependencies)

Figure A.2: A sketch of the computational graph of ICOSWM-AD, for reverse mode derivative propagation. The left part shows the forward evaluation, starting from the initial state, continuing with iterative time stepping and concluding with the goal calculation. The right part shows the corresponding propagation of derivatives from the seeds of the final goal to the adjoint variables of the initial conditions.

The routines `TAPE_INIT` and `TAPE_TURN_OFF` do exactly what they promise to do. `TAPE_INIT` creates the data construct TAPE that records all variable instances and all types of operations performed on these instances; `TAPE_TURN_OFF` stops recording. The routine `SEED (A,B)` initializes the backward run: the adjoint variable corresponding to variable `A` is initialized with the value `B`. The routine `TAPE_INTERPRETER` is equivalent to the full backward run: the derivative components are propagated through the reverse mode computational graph. The routine `ACCESS_DERIV (A)` returns the propagated derivatives saved in the adjoint variables of variable A. We see that both for seeding and accessing the derivatives we use the standard forward variable to reach the connected adjoint variable. There is no need in the code for explicit and separate adjoint variables.

There are two problems with this approach. First, memory requirement: the TAPE saves all variables in all instances and all operations between them in memory. We have already mentioned in Section A that this is not feasible for high resolution GFD applications. This is a general problem for the reverse mode of Algorithmic Differentiation because we need to save every value of the forward model evaluation for a backward sweep. The memory requirements are especially high for implicit time scheme solvers that overwrite certain fields many times during an iterative process. In our case, the TAPE saves all instances of the fields (see Figure A.1). For ICOSWM this means we have memory demands in the order of tens of Gigabyte for only a few time steps on resolutions larger than $\Delta 5$ (Table 3.1). The second problem is that for the application of error estimation in this thesis we need the derivatives of the goal with respect to ALL local changes, not only the initial controls.
The memory problem is common to both TAPE (operator overloading) and SCT approaches. There is a solution, the concept of checkpoints. During a passive forward evaluation, only certain states of the system are saved in memory (called checkpoints). During the backward propagation, these checkpoints are used to actively recalculate the missing parts of the solution that could not be saved in memory. Checkpointing therefore replaces memory demand by additional load on the CPU (some parts of the integration are done redundantly). There is a theory of optimal checkpointing (Kowarz and Walther 2006), optimal online checkpointing and more. For ICOSWM, we are in the comfortable situation that the computational graph gets very "thin" at certain points during the computation: all information transfer between one time step and the next is done via the prognostic variables. If forward information transfer is only done via these edges of the computational graph, this means that adjoint propagation is also done only along these edges. This means we only have to save the prognostic variables as checkpoints. For ICOSWM is a 2D model it is possible to save all prognostic variables $\mathbf{q}_{prog}$ in the memory (for runs of limited length). This means: we do one passive forward run and save all prognostic fields. We then restart from the last check point

and do a reverse mode sweep time step after time step. This allows efficient usage of the concept of checkpoints: we only need to recompute each time step once. The second problem is very specific to our model and application. For goal error estimation we need the full adjoint sensitivity for all prognostic variables to get weights for local error estimates. This is usually not the case for GFD applications. 4DVar and other optimization routines only need the final derivative with respect to the controls. The derivatives of the goal with respect to the prognostic variables at any intermediate time step are identical to the intermediate propagated derivatives when we differentiate the goal with respect to the initial conditions of these prognostic variables. This is good news, we solve the same adjoint problem as all typical applications. The only additional thing we have to do is to access and save all intermediate adjoint values, the full adjoint solution.
At this point, Equation (A.12) is of crucial importance: the dependency of the prognostic variables in one time step on the previous one is not just a simple function of the previous one but includes the dependency on the diagnostic variables. During the forward evaluation of one time step i to $i+1$, the values of the prognostic variable $\mathbf{q}_{prog}^{i}$ do not change. During the backward propagation of the same time step from $i+1$ to i the adjoint values $\mathbf{q}_{prog}^{;i}$ do change because the derivatives are added:

$$\frac{d\mathbf{q}_{prog}^{i+1}}{\mathbf{q}_{prog}^{i+1}} = \frac{\partial f}{\partial \mathbf{q}_{prog}^{i}} + \frac{\partial f}{\partial \mathbf{q}_{diag}}. \tag{A.13}$$

We have to make sure to access the adjoint solutions of the correct original instance of prognostic variables before the calculation of the diagnostic variables, see Figure A.3. This is an important point for all attempts to access the full adjoint solution in systems with diagnostic variables.

The Full Algorithm of ICOSWM-AD

We can now introduce a simplified version of the actual ICOSWM-AD algorithm including checkpointing. During a first passive forward sweep ICOSWM-AD saves the checkpoints for all time steps but the last one with the routine `SAVE_CHECKPOINT` ($\mathbf{q}_{prog}^{i}$). The last time step is solved with activated TAPE. We seed the goal with a partial derivative of one and the CompAD module routine `TAPE_INTERPRETER` propagates this information backwards (to the beginning of the last time step). We then load each time step with the routine `LOAD_CHECKPOINT` ($\mathbf{q}_{prog}^{i}$), solve the time step with activated TAPE, seed with the propagated derivatives of the next time step and repeat backward sweep with the routine `TAPE_INTERPRETER`. We use the accessed derivatives for our goal error estimation that are also used for seeding the previous time step. We see a sketch of this checkpointing behavior in Figure A.3.

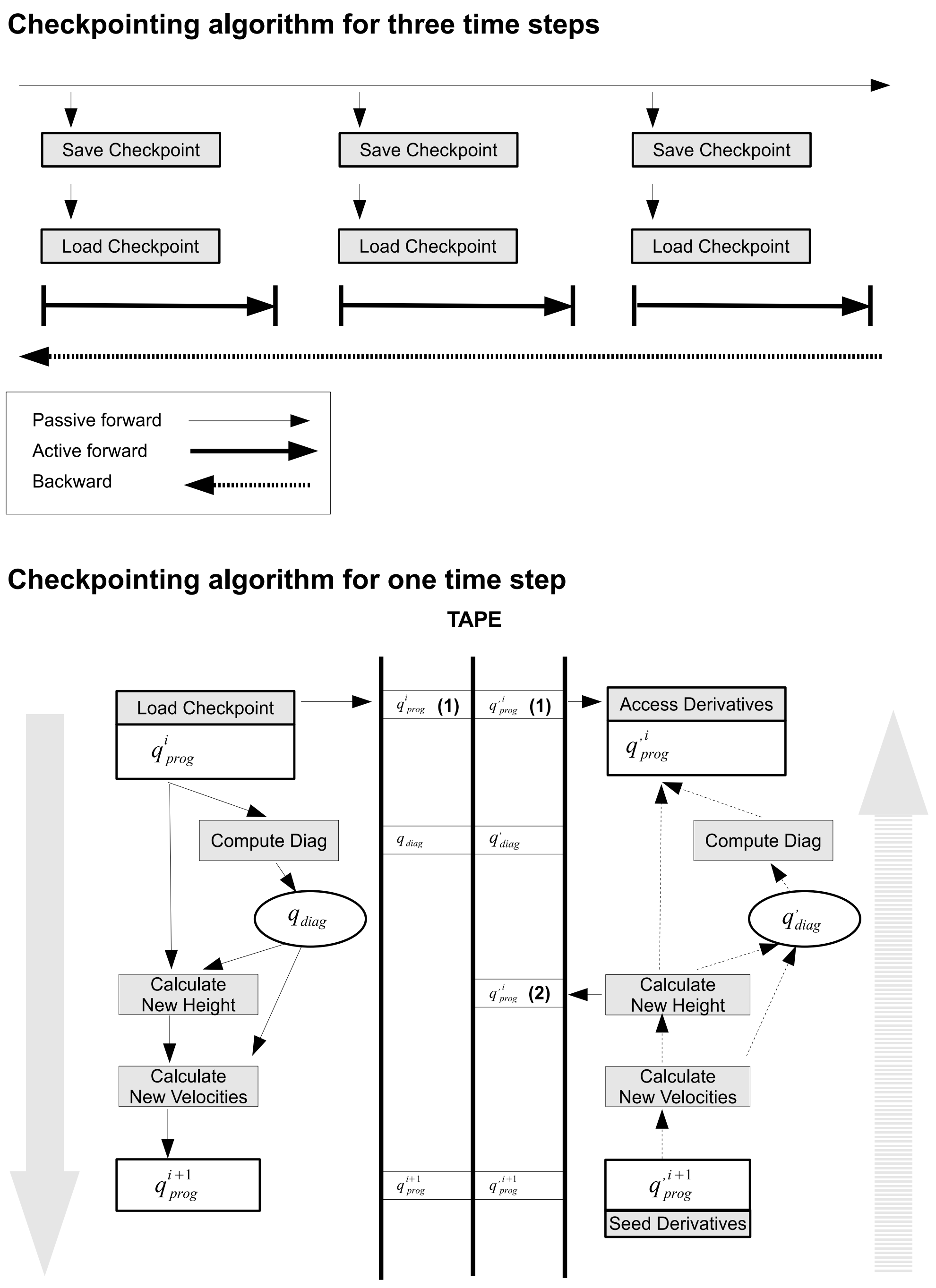


Figure A.3: A sketch of the checkpointing concept in ICOSWM-AD. The top part shows the general concept. The bottom part shows the active computation of one time step. The left part shows the forward evaluation, starting from one checkpoint of state $\mathbf{q}^{i}_{prog}$. The right part shows the corresponding propagation of derivatives from the seeds of state $\mathbf{q}'^{i+1}_{prog}$.

Full ICOSWM-AD algorithm with checkpointing

Forward model run

```
CALL ICON_INIT(controls)
DO i LOOP from 1 to N-1
      CALL SAVE_CHECKPOINT ($\mathbf{q}^{i}_{prog}$)
      CALL ICON_TIMESTEP
END LOOP
```

Backward model run

```
CALL TAPE_INIT
CALL LOAD_CHECKPOINT ($\mathbf{q}^{N-1}_{prog}$)
CALL ICON_TIMESTEP
CALL GOAL_CALCULATION (goal)
CALL TAPE_TURN_OFF
CALL SEED (goal,1)
CALL TAPE_INTERPRETER
$\mathbf{q}'^{N-1}_{prog}$= ACCESS_DERIV ($\mathbf{q}^{N-1}_{prog}$)

CALL ICON_INIT(controls)
DO i LOOP from N-1 to 1
      CALL TAPE_INIT
      CALL LOAD_CHECKPOINT ($\mathbf{q}^{i}_{prog}$)
      CALL ICON_TIMESTEP
      CALL TAPE_TURN_OFF
      CALL SEED ($\mathbf{q}^{i+1}_{prog}$,$\mathbf{q}'^{i}_{prog}$)
      CALL TAPE_INTERPRETER
      $\mathbf{q}'^{i}_{prog}$= ACCESS_DERIV ($\mathbf{q}^{i}_{prog}$)
END LOOP
```

ICOSWM shares the grid, the discrete operators, and a big portion of its code with the dynamical kernel of the atmosphere/ocean general circulation model ICON. The construction of ICOSWM-AD is a first step towards a differentiation-enabled version of ICON.

Bibliography

Ainsworth, M. and J. Oden, 1997: A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, **142 (1–2)**, 1–88.

Anderson, J. L. and S. L. Anderson, 1999: A monte carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. *Monthly Weather Review*, **127 (12)**, 2741–2758.

Babuska, I. and W. Rheinboldt, 1978: A-posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering*, **12 (10)**, 1597 – 1615.

Becker, R. and R. Rannacher, 2002: An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, **10**, 1–102.

Bischof, C. H., H. M. Bücker, P. D. Hovland, U. Naumann, and J. Utke, (Eds.) , 2008: *Advances in Automatic Differentiation*, Lecture Notes in Computational Science and Engineering, Vol. 64. Springer, Berlin.

Bischof, C. H., H. M. Bücker, and B. Lang, 2002: Automatic differentiation for computational finance. *Computational Methods in Decision-Making, Economics and Finance*, E. J. Kontoghiorghes, B. Rustem, and S. Siokos, Eds., Kluwer Academic Publishers, Dordrecht, Applied Optimization, Vol. 74, chap. 15, 297–310.

Bonaventura, L. and T. Ringler, 2005: Analysis of discrete shallow-water models on geodesic delaunay grids with c-type staggering. *Monthly Weather Reviews*, **133**, 2351–2373.

Brovkin, V., T. Raddatz, C. H. Reick, M. Claussen, and V. Gayler, 2009: Global biogeophysical interactions between forest and climate. *Bulletin of the American Meteorological Society*, **36**, L07 405.

Buizza, R., M. Milleer, and T. N. Palmer, 1999: Stochastic representation of model uncertainties in the ECMWF ensemble prediction system. *Quarterly Journal of the Royal Meteorological Society*, **125**, 2887–2908.

Cam, L. L., 1986: The central limit theorem around 1935. *Statistical science*, **1 (1)**, 78–96.

Charney, J. G., R. Fjoertoft, and J. von Neumann, 1950: Numerical integration of the barotropic vorticity equation. *Tellus*, **2**, 237 254.

Davis, T. A., 2004: Algorithm 832: UMFPACK V4.3 – An unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, **30 (2)**, 196–199.

Dongarra, J. J., H. W. Meuer, and E. Strohmaier, 2010: Top500 supercomputer sites. *http://www.netlib.org/benchmark/top500.html. (updated every 6 months).*

Ehrendorfer, J. J. T., Martin, 1997: Optimal prediction of forecast error covariances through singular vectors. *Journal of the Atmospheric Sciences*, **54**, 286–313.

Giles, M. B., 1998: On adjoint equations for error analysis and optimal grid adaptation in CFD. *Frontiers of computational fluid dynamics*, 155–169.

Giles, M. B. and N. Pierce, 2000: Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Review*, **42 (2)**, 247–264.

Giles, M. B., N. Pierce, and E.Sueli, 2004: Progress in adjoint error correction for integral functionals. *Computing and Visualization in Science*, **6**, 113–121.

Giorgetta, M., T. Hundertmark, P. Korn, S. Reich, and M. Restelli, 2009: Conservative space and time regularizations for the ICON model. *Berichte zur Erdsystemforschung*, **67**.

Givon, D., R. Kupferman, and A. Stuart, 2004: Extracting macroscopic dynamics: model problems and algorithms. *Nonlinearity*, **17**, R55–R127.

Griewank, A., 2000: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* No. 19 in Frontiers in Appl. Math., SIAM, Philadelphia, PA.

Hinze, M. and S. Volkwein, 2005: Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: Error estimates and suboptimal control. *Dimension Reduction of Large-Scale Systems*, Lecture Notes in Computational Science and Engineering, Vol. 45, chap. 10, 297–310.

Johnson, C., R. Rannacher, and M. Boman, 1995: Numerics and hydrodynamic stability: Toward error control in computational fluid dynamics. *SIAM Journal on Numerical Analysis*, **32 (4)**, 1058–1079.

Jungclaus, J. H., et al., 2010: Climate and carbon-cycle variability over the last millennium. *Climate of the Past*, **6 (5)**, 723–737.

Kalnay, E., 2003: *Atmospheric modeling, data assimilation, and predictability.* Cambridge University Press.

Kalnay, E., H. Li, T. Miyoshi, S. Yang, and J. Ballabera-Poy, 2007: 4-D-Var or ensemble kalman filter? *Tellus A*, **59 (5)**, 758–773.

Kowarz, A. and A. Walther, 2006: Optimal checkpointing for time-stepping procedures in ADOL-C. *International Conference on Computational Science (4)*, 541–549.

Laeuter, M., D. Handorf, and K. Dethloff, 2005: Unsteady analytical solutions of the spherical shallow water equations. *Journal of Computational Physics*, **210**, 535–553.

Majda, A. J., C. Franzke, and D. Crommelin, 2009: Normal forms for reduced stochastic climate models. *Proceedings of the National Academy of Sciences*, **16 (10)**, 3649–3653.

Majda, A. J. and S. Stechmann, 2009: Gravity waves in shear and implications for organized convecion. *Journal of the Atmospheric Sciences*, **66 (9)**, 2579–2599.

Mani, K. and D. J. Mavriplis, 2009: Error estimation and adaptation for functional outputs in time-dependent flow problems. *Journal of Computational Physics*, **229**, 415–440.

Marotzke, J., R. Giering, K. Zhang, D. Stammer, C. Hill, and T. Lee, 1999: Construction of the adjoint MIT ocean general circulation model and application to atlantic heat transport sensitivity. *Journal of Geophysical Research*, **104 (C12)**, 29,529 – 29,547.

Marsaglia, G., 1991: Normal (gaussian) random variables for supercomputers. *The Journal of Supercomputing*, **5 (1)**, 49–55.

Meehl, G., et al., 2007: *Global Climate Projections in 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, UK.

Meehl, G. A., G. J. Boer, C. Covey, M. Latif, and R. J. Stouffer, 2000: The coupled model intercomparison project (CMIP). *Bulletin of the American Meteorological Society*, **81**, 313 – 318.

Molteni, F., R. Buizza, T. N. Palmer, and T. Petroliagis, 1996: The ECMWF ensemble prediction system: Methodology and validation. *Quarterly Journal of the Royal Meteorological Society*, **122**, 73–119.

Mori, H., 1965: Transport, collective motion, and brownian motion. *Progress of Theoretical Physics*, **33**, 423–450.

Mori, H., H. Fujisaka, and H. Shigematsu, 1974: A new expansion of the master equation. *Progress of Theoretical Physics*, **51**, 109–122.

Naumann, U. and J. Riehme, 2005: A differentiation-enabled Fortran 95 compiler. *ACM Transactions on Mathematical Software*, **31 (4)**, 458–474.

Naumann, U. and J. Riehme, 2006: Computing adjoints with the nagware fortran 95 compiler. *M. Buecker et. al., editors, Automatic Differentiation: Applications, Theory, and Tools*, **50**, 159–170.

Oden, J. T. and S. Prudhomme, 2002: Estimation of modeling error in computational mechanics. *Journal of Computational Physics*, **182**, 496–515.

Pedlosky, J., 1982: *Geophysical Fluid Dynamics*. Springer-Verlag, 58++ pp.

Polya, G., 1920: Über den Zentralen Grenzwertsatz der Wahrscheinlichkeitsrechnung und das Momentproblem. *Mathematische Zeitschrift*, **8**, 171–180.

Power, P., M. Piggott, F. Fang, G. Gorman, C. Pain, D. Marshall, A. Goddard, and I. Navon, 2006: Adjoint goal-based error norms for adaptive mesh ocean modelling. *Ocean Modelling*, **15**, 3–38.

Prudhomme, S. and J. Oden, 2002: Computable error estimators and adaptive techniques for fluid flow problems. *In: Barth, T.J., Deconinck, H. (Eds.), Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics, Lect. Notes Comput. Sci. Eng. 25, Springer*, **25**, 207–268.

Rasch, P. J., D. B. Coleman, N. Mahowald, D. L. Williamson, S.-J. Lin, B. A. Boville, and P. Hess, 2006: Characteristics of atmospheric transport using three numerical formulations for atmospheric dynamics in a single GCM framework. *Journal of Climate*, **19**, 2243 –2266.

Rauser, F., P. Korn, and J. Marotzke, 2011: Predicting goal error evolution from near-initial-information: A learning algorithm. *Journal of Computational Physics*, **230 (19)**, 7284–7299.

Rauser, F., J. Riehme, U. Naumann, P. Korn, and K. Leppke, 2010: On the use of discrete adjoints in goal error estimation for shallow water equations. *Procedia Computer Science (ICCS 2010 Proceedings)*, **1**, 1.

Ripodas, P., et al., 2009: Icosahedral shallow water model (ICOSWM): results of shallow water test cases and sensitivity to model parameters. *Geoscientific Model Development Discussions*.

Seiffert, R., R. Blender, and K. Fraedrich, 2006: Subscale forcing in a global atmospheric circulation model and stochastic parametrization. *Quarterly Journal of the Royal Meteorological Society*, **132**, 1627 – 1643.

Shutts, G., 2005: A kinetic energy backscatter algorithm for use in ensemble prediction systems. *Quarterly Journal of the Royal Meteorological Society*, **131 (612)**, 3079–3102.

Sonar, T. and E. Sueli, 1998: A dual graph-norm refinement indicator for finite volume approximations of the euler equations. *Numerische Mathematik*, **78**, 619–658.

Stewart, J. R. and T. Hughes, 1998: A tutorial in elementary finite element error analysis: A systematic presentation of a priori and a posteriori error estimates. *Computer methods in applied mechanics and engineering*, **158**, 1–22.

Treut, H. L., R. Somerville, U. Cubasch, Y. Ding, C. Mauritzen, A. Mokssit, T. Peterson, and M. Prather, 2007: *Historical Overview of Climate Change*. Cambridge University Press, Cambridge, UK.

van den Berge, L., F. Selten, W. Wiegerinck, and G. Duane, 2010: A multi-model ensemble method that combines imperfect models through learning. *Earth System Dynamics*.

Venditti, D. A. and D. L. Darmofal, 2000: Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flows. *Journal of Computational Physics*, **164**, 204–227.

von Storch, H. and F. W. Zwiers, 1999: *Statistical Analysis in Climate Research*. Cambridge University Press, Cambridge, UK.

Williamson, D. and J. Drake, 1992: A standard test set for numerical approximations to the shallow water equations in spherical geometry. *Journal of Computational Physics*, **102 (1)**, 211 – 224.

Wunsch, C., P. Heimbach, R. Ponte, and I. Fukumori, 2009: The global general circulation of the ocean estimated by the ecco-consortium. *Oceanography*, **22**, 88 – 103.

Zwanzig, R., 1973: Nonlinear generalized langevin equations. *Journal of Statistical Physics*, **9**, 215–220.

Acknowledgements

Consistency is the last refuge of the unimaginative.
Oscar Wilde

First, I wish to thank Peter (Korn) to be an exceptional supervisor during the last years. I enjoyed the trustful cooperation and I am grateful that the results of this thesis were **not** hidden somewhere before I even started. Second, I wish to thank Jochem (Marotzke) to be an excellent co-advisor. Your insights in the principles of science, scientific communication and writing were very helpful in developing a way to understand and communicate science. Third, I wish to thank Detlef Stammer for chairing my panel and for guidance throughout my panel meetings.

I would like to thank the IMPRS-ESM office for providing an outstanding PhD environment. Special thanks to Antje Weitz and Cornelia Kampmann for their organizational and personal support. Thanks to the Central IT Services for their technical support.

Personal thanks go especially to Dr. Malte Heinemann, Dr. Juliane Otto, Dr. Jonas Bhend, Dr. Julia Pongratz, Dr. Aiko Voigt, Rosi, Fanny, Freja, Steffen, Mario, Ronny, Daniel, Jaison, Eleftheria, Stergios, Maria Paz, Sebastian, Laura and Iris! I hereby officially acknowledge my dear colleague and dearest friend Nils Fischer: we did it! Special greetings to the LA2010 Venice Beach crowd, Peter, Werner & Lorenzo, that was no place for the weary kind! Thanks also to Cafe Alibi for enabling Werner and myself to survive Berlin 2008! Thanks to the original thursday group, the participants of COP15 MUN 2009 and the development team of WOODSTOCK-CM! Thanks also to PT, Christina, Magdalena, Heather, Burkhard, Martin, Fadi, Sophia, Franzi, Stefan, Pauline, Fadi, Christine, Julia, my personal Wednesday evening support group for anonymous scientists!

I do thank Jan Riehme and Uwe Naumann for excellent support concerning the usage and development of the Differentiation-enabled NAGware F95 compiler - and for good times.
I also want to thank Mike Giles for kindly providing example Matlab code in the beginning of this work. I am grateful to Marco Giorgetta for reading the manuscript of

Chapter 3 carefully during the internal review process and for many helpful comments. I thank the external reviewers of Chapters 3 and 4 for helpful insights.

Special thanks to Peter, Peter, Laura, Lorenzo, Kevin, Dirk, Jan, Nils and Heather for proofreading and revision comments!

I wish to say thanks to my family for always being supportive. I also thank Mela, for always being there and believing in me. The last years were slightly tiresome. It's high time to reboot the world.

I dedicate this work to Barney Stinson. True Story.

ISSN 1614-1199