HAL
archives-ouvertes.fr

A large-scale atmospheric chemistry-transport model for massively parallel architectures

Alexis Praga

▶ **To cite this version:**

Alexis Praga. A large-scale atmospheric chemistry-transport model for massively parallel architectures. Computer Science [cs]. Université Toulouse III - Paul Sabatier, 2015. English. <tel-01178394>

HAL Id: tel-01178394

https://tel.archives-ouvertes.fr/tel-01178394

Submitted on 19 Jul 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *30 janvier 2015* **par :**

Alexis Praga

Un modèle de transport et de chimie atmosphérique à grande échelle adapté aux calculateurs massivement parallèles

---

JURY

F. Lefevre	Directeur de Recherche CNRS	Rapporteur
V.-H. Peuch	Senior Scientist	Rapporteur
D. Hauglustaine	Directeur de Recherche CNRS	Rapporteur
J.-L. Attie	Professeur des Universités	Examinateur
B. Josse	Ingénieur	Examinatrice
D. Cariolle	Ingénieur Général des Ponts et Chaussées	Directeur de thèse
L. Giraud	Directeur de Recherche Inria	Co-directeur de thèse

École doctorale et spécialité :
SDU2E : Océan, Atmosphère et Surfaces Continentales
Unité de Recherche :
CERFACS
Directeur(s) de Thèse :
Daniel Cariolle et *Luc Giraud*
Rapporteurs :
Franck Lefèvre, *Vincent-Henri Peuch* et *Didier Hauglustaine*

COPYRIGHT

Un modèle de transport et de chimie atmosphérique à grande échelle adapté aux calculateurs massivement parallèles

Alexis Praga
2015

This work is licensed under a Creative Commons Attributions 3.0 License. To view a copy of the license, go to: http://creativecommons.org/licenses/by/3.0/

À mes parents

Remerciements

Une thèse peut paraître un travail bien solitaire. Cependant, derrière la scène, il y a un grand nombre de personnes sans qui je n'aurais pas été capable d'aller jusqu'au bout, grâce à leur soutien moral, et parfois physique !

Tout d'abord, je remercie les rapporteurs, Vincent-Henri Peuch, Didier Hauglustain et Franck Lefèvre, pour avoir pris le temps de lire et de comprendre mon manuscrit. Sans oublier les autres membres du jury, Béatrice Josse et Jean-Luc Attié, pour m'avoir consacré un peu de leur temps ainsi que pour leurs questions pertinentes. Ensuite, je tiens à remercier le CERFACS ainsi que Météo-France pour cette opportunité de thèse et leur soutien financier.

Durant ces trois années, les (car il y en a plusieurs) directeurs ont là pour assurer leur guidance et leur soutien de vieux briscards de la recherche au jeune thésard quelque peu perdu dans le labyrinthe de ses simulations que j'étais. Donc un grand merci à Daniel et Luc pour leurs conseils avisés et leur présence quand il le fallait, notamment lors de la dernière ligne droite.

L'équipe AE a été évidemment un pan indispensable de cette thèse, avec le travail d'Emmanuel, le soutien de bon aloi d'Odile ainsi que les encouragements de Roberto, Hannah et Hélène. Qui dit thèse informatique, dit également support informatique et l'équipe CSG a su être à la hauteur avec la réactivité de Fabrice, Gérard, Maurice et Patrick, sans oublier le support impeccable d'Isabelle. Côté administration, on peut mentionner Chantal et Michèle qui ne se contentent pas de soutenir le CERFACS mais qui sont aussi des personnes formidables.

À tous ceux-là, il faut également ajouter tous les amis, toujours là pour remonter le moral quand il faut. Bien évidemment la fidèle Adèle qui supporta sans broncher les sautes d'humeurs du Pangolin, mais aussi le brave Joris, source d'innombrables discussions geek et Dimitris, toujours là pour me rappeler à l'ordre. Une pensée également pour Lisa, une source inépuisable de bon conseils, Sophie, le rayon de soleil du vendredi après-midi ainsi qu'Audrey qui nous a plus d'une fois sorti de notre morne torpeur de thésards. Mais il faut aussi remercier Thomas qui fut un ami sur lequel on peut compter, Marie n^{o} 1 pour les causeries à rallonge et à des heures indûse, Marie n^{o} 2 pour nos prises de becs et son soutien malgré tout.

Hors des murs du CERFACS, on ne peut oublier de mentionner mes parents, sans qui toute cette aventure n'aurait pas eu lieu. Merci également à ma grand-mère, pour tout ce qu'elle m'a apporté. Enfin, mais non des moindres, je remercie ma chère Laure qui fut source de tant d'encouragements et de réconfort que cette page ne suffirait pas à décrire.

Abstract

We present in this thesis the development of a large-scale bi-dimensional atmospheric transport scheme designed for parallel architectures with scalability in mind. The current version, named Pangolin, contains a bi-dimensional advection and a simple linear chemistry scheme for stratospheric ozone and will serve as a basis for a future Chemistry Transport Model (CTM). For mass-preservation, a van Leer finite-volume scheme was chosen for advection and extended to 2D with operator splitting. To ensure mass preservation, winds are corrected in a preprocessing step. We aim at addressing the "pole issue" of the traditional regular latitude-longitude by presenting a new quasi-area-preserving grid mapping the sphere uniformly. The parallelization of the model is based on the advection operator and a custom domain-decomposition algorithm is presented here to attain load-balancing in a message-passing context. To run efficiently on current and future parallel architectures, algebraic features of the grid are exploited in the advection scheme and parallelization algorithm to favor the cheaper costs of flops versus data movement. The model is validated on algebraic test cases and compared to other state-of-the-art schemes using a recent benchmark. Pangolin is also compared to the CTM of Météo-France, MOCAGE, using a linear ozone scheme and isentropic coordinates.

Résumé

Cette thèse présente un modèle bi-dimensionnel pour le transport atmosphérique à grande échelle, nommé Pangolin, conçu pour passer à l'échelle sur les achitectures parallèles. La version actuelle comporte une advection 2D ainsi qu'un schéma linéaire de chimie et servira de base pour un modèle de chimie-transport (MCT). Pour obtenir la conservation de la masse, un schéma en volume-finis de type van Leer a été retenu pour l'advection et étendu au cas 2D en utilisant des opérateurs alternés. La conservation de la masse est assurée en corrigeant les vents en amont. Nous proposons une solution au problème "des pôles" de la grille régulière latitude-longitude grâce à une nouvelle grille préservant approximativement les aires des cellules et couvrant la sphère uniformément. La parallélisation du modèle se base sur l'advection et utilise un algorithme de décomposition de domaines spécialement adapté à la grille. Cela permet d'obtenir l'équilibrage de la charge de calcul avec MPI, une librairie d'échanges de messages. Pour que les performances soient à la hauteur sur les architectures parallèles actuelles et futures, les propriétés analytiques de la grille sont exploitées pour le schéma d'advection et la parallélisation en privilégiant le moindre coût des flops par rapport aux mouvement de données. Le modèle est validé sur des cas tests analytiques et comparé à des schémas de transport à l'aide d'un comparatif récemment publié. Pangolin est aussi comparé au MCT de Météo-France via un schéma linéaire d'ozone et l'utilisation de coordonnées isentropes.

Contents

Introduction

L'humanité s'est toujours intéressée à la prédiction du temps pour le lendemain, ainsi qu'à des échelles de temps plus longues. Depuis le milieu du 20^{eme} siècle, des modèles numériques ont été utilisés et continuent d'être développés afin de prédire l'évolution de l'ensemble des composantes de l'atmosphère. Ces modèles se divisent en deux catégories selon l'échelle de temps considérée. Les modèles de climat globaux permettent de prédire l'état moyen de l'atmosphère sur de longues périodes de temps, de l'ordre de plusieurs années ou plus. Ils fonctionnent à l'échelle globale, par opposition à l'échelle régionale, et nécessitent donc une résolution plus grossière. Comme ils peuvent être utilisés pour estimer le réchauffement climatique sur une période d'un siècle, leur coût informatique est très élevé. La seconde catégorie comporte les modèles de prévision météorologique, qui sont ce qu'on associe en général avec la « météo ». Ces modèles visent l'échelle régionale avec une échéance temporelle plus réduite, de quelques heures à quelques jours. Ici, la précision et la rapidité d'exécution sont des facteurs déterminants. Cependant, la frontière entre ces deux catégories est en train de s'estomper : les modèles globaux utilisent des résolutions de plus en plus fines alors que les modèles régionaux augmentent la taille de leur domaines. Comme ces modèles sont très complexes, ils sont divisés en « blocs » dans la pratique, où chaque bloc gère un sous-ensemble de processus physiques. Pour cette thèse, nous nous concentrons sur un de ces sous-ensembles, le Modèle de Chimie-Transport (MCT).

Un modèle de chimie-transport modélise la chimie de l'atmosphère. Il s'agit d'une tâche ardue par le nombre de processus, comme le montre la Fig. 1. Dans un modèle de la physique et chimie de l'atmosphère, on se concentre généralement sur les processus principaux décrits dans la Fig. 2, qui contribuent à l'évolution spatiale et temporelle des espèces chimiques atmosphériques. Les modèles de chimie-transport ont différentes applications, telles que l'estimation de l'impact des émissions anthropiques. Lorsque des polluants sont émis en Asie, comme ce fut le cas lors de l'incident à la centrale de Fukushima Daiichi, il peut être vital de modéliser le transport au dessus de l'océan Pacifique. Les MCTs peuvent aussi être utilisés pour étudier les cycles biochimiques comme la création ou le transport des particules de sulfate lors d'éruptions volcaniques, comme ce fut le cas avec l'Eyjafjallajökull. Ces modèles peuvent aussi servir à établir des simulations dites *paléochimiques* de la qualité de l'air à des époques préhistoriques pour évaluer l'impact d'évènements extrêmes (tels que les impacts d'astéroïdes ou les éruptions volcaniques) sur la chimie et le climat. Un autre champ d'application est la prévision du « temps chimique », où la composition de l'atmosphère est estimée sur plusieurs jours en utilisant des prévisions météorologiques ainsi que des cadastres d'émissions. Dans ce cas, on se concentre sur la couche limite, la stratosphère et la troposphère.

Cela permet de prédire l'indice d'UV ou les pics de pollution. Plusieurs pays en Europe ont mis en place de tels systèmes. En France, le gouvernement a installé le projet « Prév'air » pour permettre au grand public d'accéder à des prévisions de qualité de l'air jusqu'à deux jours à l'aide des modèles MOCAGE et CHIMERE. Les résultats sont visible sur Internet (www2.prevair.org). Les MCTs permettent également d'étudier les interactions entre la chimie et le changement climatique. Ainsi, l'IGAC est un comité qui étudie entre autres l'impact de l'ozone sur le changement climatique et les effets des aérosols sur les nuages (voir par exemple IGAC (2006)).

Pour cette thèse, nous nous focalisons sur les schémas numériques d'advection et de chimie d'un MCT. En particulier, le but de la thèse est de développer la base d'un futur MCT en se concentrant sur l'advection de traceurs impliqués dans la chimie atmosphérique. Le développement de méthodes numériques pour le transport est un domaine de recherche amorcé dans les années 60 et qui demeure très actif aujourd'hui. Il y a principalement deux axes de recherche : le premier étudie les méthodes numériques, le second se concentre sur l'adaptation de ces méthodes aux coordonnées sphériques, qui possèdent certaines spécificités. La grille régulière latitude-longitude est traditionnellement utilisée mais cette grille a l'inconvénient d'avoir deux points singuliers (les pôles) ainsi qu'une convergence des méridiens aux pôles. Cela impacte fortement les performances informatiques et a conduit récemment à étudier des grilles alternatives. Dans l'état actuel, trouver une combinaison parfaite entre le schéma d'advection et la grille sous-jacente reste un problème ouvert.
La chimie atmosphérique met en jeu des dizaines de milliers d'espèces chimiques. Comme inclure dans un modèle numérique toutes ces espèces serait trop coûteux, on se restreint en pratique à une centaine d'espèces. D'un point de vue mathématique, il faut résoudre un système d'équations dit *raide*, c'est-à-dire que les méthodes explicites sont inefficaces. De plus, la chimie est locale contrairement à l'advection, donc en pratique cela justifie de traiter la chimie et l'advection séparément dans un MCT.

Le choix d'un modèle dépend fortement de l'architecture informatique sur laquelle il s'exécute. Alors que les modèles mathématiques existent depuis le début du 20^{eme} siècle, les premiers modèles opérationnels de prévisions météorologique sont apparus dans les années 50. L'augmentation de la puissance de calcul a conduit à une complexification des modèles ainsi qu'à l'utilisation de résolutions de plus en plus fines. Cela a en retour imposé une forte contrainte sur les performances des modèles. L'augmentation de la résolution permet d'améliorer la précision à la fois de l'advection et de la chimie, ce qui est notamment utile pour les MCGs, qui visent maintenant une échelle régionale. De plus, les modèles de prévisions météorologiques nécessitent que le modèle soit suffisamment rapide pour pouvoir être intégré plusieurs fois par jour. Deux révolutions informatiques ont permis cette course à la performance. La première fut l'avènement des machines vectorielles, qui ont été depuis remplacées par des clusters de processeurs pouvant comporter plusieurs centaines de milliers de cœurs. Dans le contexte actuel, l'utilisation de ces processeurs multi-cœurs a aussi changé profondément la manière dont sont envisagées les performances en parallèle. On privilégie actuellement le passage à l'échelle des applications, c'est-à-dire l'utilisation efficace d'un nombre très important et toujours croissant de cœurs.

Motivation

Les schémas d'advection dans les modèles opérationnels actuels peuvent être divisés en deux catégories. En premier lieu, on peut regrouper les modèles efficaces numériquement mais qui ne préservent pas la masse de manière intrinsèque : ce sont les schémas de type semi-Lagrangiens. Ils sont efficaces

notamment parce qu'ils permettent des pas de temps plus longs que les schémas Eulériens. Pour assurer que la masse est bien conservée de manière globale, des correctifs *a posteriori* sont en général appliqués. Cependant, la conservation n'est pas assurée localement et ces correctifs ont un coût numérique, qui impactera probablement les performances en parallèle. Dans la seconde catégorie, on trouve les schémas conservatifs en point de grille, principalement de type volumes-finis. Leur efficacité est cependant limitée par le problème « des pôles » mentionné précédemment, qui diminue fortement le pas de temps.

Cette thèse vise à résoudre ces deux problèmatiques en même temps en fournissant un modèle d'advection-chimie conservatif et efficace. De plus, nous imposons que le modèle dispose d'une implantation parallèle performante et qui anticipe les développements futurs des architectures parallèles, principalement en diminuant le coût mémoire du modèle. Nous visons également à utiliser efficacement les architectures parallèles de Météo-France. De plus, le but à moyen terme est de créer un MCT complet afin de fournir une alternative à MOCAGE, le MCT de Météo-France. C'est pourquoi ce manuscrit présente Pangolin (en anglais *a PArallel implementatioN of a larGe scale multi-dimensiOnaL chemIstry-traNsport scheme*), une infrastructure parallèle et efficace pour l'advection et la chimie atmosphérique.

Plan

- Le chapitre 1 offre un aperçu des différents schémas d'advection et introduit le schéma en volumes-finis choisi pour Pangolin. Dans une seconde partie, nous étudions les différentes grilles disponibles sur la sphère. Puis, nous introduisons une nouvelle grille préservant approximativement les aires des cellules qui supprime le problème « des pôles ». L'implantation du schéma volumes-finis sur cette grille est également détaillée.

- Le chapitre 2 examine les performances du schéma d'advection 2D sur des cas tests analytiques. Pangolin est aussi comparé à d'autres schémas de transport à l'aide d'une suite de tests récemment publiée.

- Le chapitre 3 présente la stratégie utilisée pour paralléliser le modèle, avec notamment un algorithme de décomposition de domaine spécifiquement adapté à notre grille afin d'avoir une parallélisation efficace dans un contexte d'échange de messages. Les performances en parallèles pour l'advection 2D sont ensuite validées. Enfin, l'impact de la future chimie est estimé et nous discutons des différentes stratégies pour résoudre les problèmes qui en découlent.

- Le chapitre 4 compare les résultats de Pangolin en 2D en coordonnées isentropiques à ceux de MOCAGE en 3D sur une simulation de distribution de l'ozone stratosphérique sur un mois. Le même schéma linéaire d'ozone est utilisé dans les deux modèles pour que la comparaison soit opportune.

Introduction

Context

Humans have always been interesting in predicting the weather, either for tomorrow or for longer timescales. Since the middle of the 20th century, numerical models are being used and under constant development to forecast the state of the atmosphere. They currently fall into two categories, depending on the time-scale. Global Climate Models (GCMs) aims at predicting the average state of the atmosphere over large time periods, on the order of several years or more. They focus on the global scale, as opposed to the regional scale, and use in practice a coarser space resolution. As they can be used for example to estimate global warming over a century, they are computationally expensive. On the other hand, Numerical Weather Prediction models (NWPs) is what the general public associates with 'weather prediction'. It focuses on the regional scale with a smaller time-scale, usually hourly to weekly. Accuracy and speed are the prime factors. However, the boundary between these categories is beginning to blur as global models are using finer resolutions and regional models are using a larger scale. These models are also extremely complex so in practice, they are divided in 'blocks', where each block manages a smaller set of physical processes. In this thesis, we focus on one of these subsets, the CTM.

A CTM focuses on modelling the chemistry of the atmosphere. It is a complex task with many processes, as illustrated by the Fig. 1. On Fig. 2 are presented the main processes that should appear in a physico-chemistry model of the atmosphere, which contribute to the spatial and temporal evolution of the atmospheric species. CTMs have various applications, such as the estimation of the impact of anthropogenic emissions. When pollutants are emitted in Asia, like in the Fukushima Daiichi incident, it can be crucial to model the transport over the Pacific Ocean. They are also used to study biochemical cycles, such as the formation or transport of sulfate aerosol particles from volcanic eruptions such as the eruption of Eyjafjallajökull. They can be used to run paleochemical simulations of air quality in prehistoric times to evaluate the impact on chemistry and climate of extreme events like asteroid impacts or volcanic eruptions. Another active field of CTMs is *chemical weather forecasting*, where the composition of atmosphere is forecast for several days using weather forecasts and emissions inventories. In this case, the focus is on the boundary layer, stratosphere and troposphere and can be used to predict the UV index or peaks of pollution. Several countries in Europe are implementing such systems. The French government initiated the "Prev'air" system to offer air quality forecasts for the current day and up to two days using the MOCAGE and CHIMERE models. The results are publicly available on the web (`www2.prevair.org`). CTMs

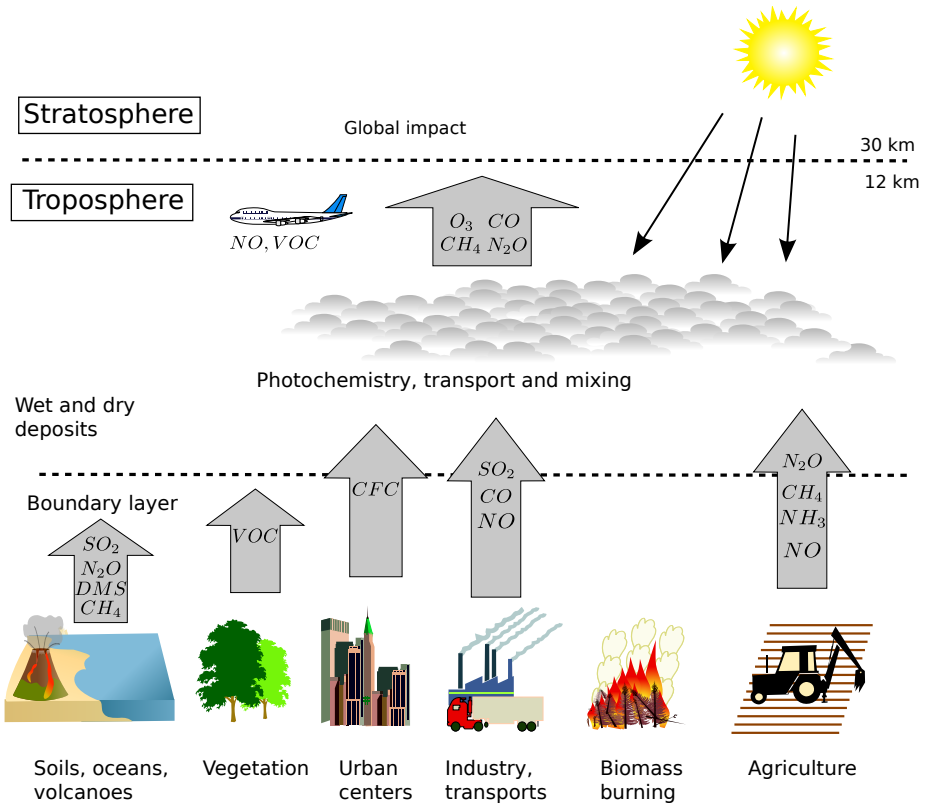


Figure 1: Main processes for atmospheric chemistry (inspired from Delmas et al. (2005), with pictures from Wikipedia). VOC stands for Volatile Organic Compound, DMS stands for dimethyl sulfide, a biological sulfur compound. CFC stands for chlorofluorocarbons, a compound massively used in the industry, which contributes to ozone depletion.

can also be used to study the interactions between chemistry and climate change. For example the IGAC committee studies, among other topics, the impact of ozone on climate change or the effect of aerosols on clouds (see for example IGAC (2006)).

In this thesis, we focus on the advection and chemistry schemes of a CTM. In particular, this thesis deals with developing the foundation of a CTM with an emphasis on multi-tracers advection. Development of numerical methods for advection has been a very active area of research since the 1960s and is still continuing today. The research falls into two categories: the first deals with the numerical methods themselves, while the second focuses on adapting these methods to the spherical geometry of the atmosphere, which possesses unique features. Usually referred to as the 'pole issue', the main issues with the traditional regular latitude-longitude grids arise from the convergence of meridians at the pole and the poles are singularities. This strongly impacts computational performances and has recently led to research alternatives with more uniform grids. At the moment, there is no perfect combination of an advection scheme on a spherical grid.
The chemistry in a CTM involves tens of thousands of chemical species. As it is too expensive

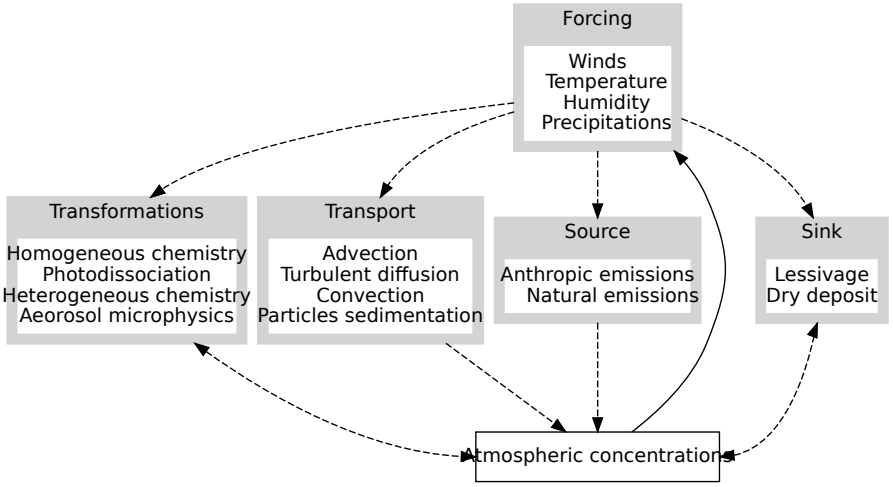


Figure 2: Main processes involved in the evolution of the chemical concentrations (either gaz or particles). They should be represented, at least minimally, in a model of the physics and chemistry of the atmosphere (inspired from Delmas et al. (2005)).

to consider all of them, a hundred of them are considered in practice, along with a thousand of chemical reactions. Mathematically, this results in a *stiff* set of equations, *i.e.*, where explicit methods are inefficient. On top of that, chemistry is completely local to each cell of the grid, contrary to advection. These factors justify in practice a separation of chemistry and advection in a CTM.

The choice of a model is highly dependant on the computer architecture it will run on. While mathematical models for weather forecast were developed at the beginning of the 20th century, the first operational weather forecast appeared in the 1950s. The increase in computation power led to more complex models and with finer resolutions ever since. These improvements have in turn put a huge pressure on computational performances. On one hand, increasing resolution is beneficial to the accuracy of both advection and chemistry and is especially useful for GCMS, which are aiming now for a regional scale. On the other hand, NWFs require computational efficiency as the model must finish fast enough to be able to run several times per day. This race to performance came from two revolutions in computer architectures. The first was the appearance of vector processors, which have been now supplanted by clusters of processors (up to hundred of thousands of cores). In the current context, using many-cores has also changed the way parallel performances are envisioned. The current focus is the ability of an application to handle an increasing number of cores, also known as *scalability*.

Motivation

Advection schemes in current operational models can be roughly put into two categories. The first contains efficient models which are not intrinsically mass-preserving known as semi-Lagrangian models. Their efficiency comes in particular from the larger time-step than their Eulerian counterparts. Mass fixers are used in practice to ensure that mass is conserved globally, without ensuring

local conservation. Unfortunately, these fixers have an additional computation cost, which is likely to impact parallel performances. The second category contains conservative grid-point schemes, which are usually finite-volume schemes. Efficiency is however severely limited due to the 'pole problem', which constraints the time-step.

This thesis aims at addressing these two issues at once by having a conservative and efficient model for advection and chemistry. On top of that, we require the model to have an efficient parallel implementation and to anticipate future development of parallel architectures, mostly by decreasing the memory footprint. In a near future, we also aim to develop a fully-flegded CTM as an alternative to MOCAGE, the CTM of Météo-France, and to exploit the parallel architectures of Météo-France. Hence, we present Pangolin, a PArallel implementatioN of a larGe scale multi-dimensiOnaL chemIstry-traNsport scheme, which serves as an efficient parallel framework for atmospheric advection and chemistry.

Overview

- Chapter 1 gives an overview of the different advection schemes and introduces the finite-volume scheme chosen for this thesis. In a second part, we study the different spherical grids available. Then we introduce a new quasi-area preserving grid to remove the 'pole issue' and show how the scheme is implemented on this grid.

- Chapter 2 examines the performance of the our bi-dimensional advection scheme on algebraic test cases. Pangolin is also compared to state-of-the-art transport schemes using a recent testing-suite.

- Chapter 3 presents the parallelization strategy and explains how a custom domain-decomposition algorithm was developed for an efficient parallelization using the message-passing paradigm. The performances are validated in parallel for 2D advection. The impact of adding the chemistry is estimated and future strategies are also proposed.

- Chapter 4 compares and discusses a 2D run of Pangolin using isentropic coordinates to a 3D run of MOCAGE to simulate stratospheric ozone distribution over a month. A linear ozone scheme is used in both models.

Chapter 1

Transport

1.1 Introduction

This chapter focuses on the transport scheme in a CTM. As illustrated in the introduction (see Fig. 2), the transport is actually a set of several processes: advection, turbulent diffusion, convection and particles sedimentation. Some of these processes can be represented directly, *i.e.*, the model solves the corresponding equations to find the evolution of the quantities over time. It is the case for advection, where large-scale winds move the chemical species away from their sources, and for particles sedimentation. Other processes are too complex to be represented directly and a *parametrization* is used to estimate their average impact on the tracers inside the cells of a model. This is the case for turbulent diffusion and convection as their spatial and temporal scales are too small for the model.

In this thesis, we focus on horizontal advection, which is a core process for CTMs. It is especially important for our model as it serves as the framework for our parallelization strategy (presented in Chapter 3). After presenting key properties for an advection scheme in Section 1.2, we then give an overview of the different numerical methods used in past and current transport schemes to solve the mass and continuity equations for tracers (Section 1.3). Mass conservation is a key property in CTMs so we examine finite-volume schemes which allow for local and global conservation. The description of the scheme chosen in Pangolin follows. A major drawback for finite-volume schemes and Eulerian schemes is the 'pole problem' on the traditional regular latitude-longitude grid. Active research is currently performed on alternative grids to avoid this issue. This issue, along with the current alternative grids that have been so far studied to solve it, are presented in Section 1.5. Within Pangolin, we introduce a new quasi-area preserving grid which maps uniformly the sphere and alleviates the pole issue. The new grid is presented in Section 1.6, along with the necessary adaption of the finite-volume scheme.

1.2 Properties of the scheme

Here, we examine desirable features of a transport scheme in atmospheric applications. The first property is accuracy, which quantifies how close we are to the exact solution in a given norm. There are several possibilities to define a norm, but in practice, the most common norms are

(see Williamson et al. (1992)):

$$l_1 = \frac{\mathcal{I}(|q - q_E|)}{\mathcal{I}(|q_E|)} \qquad l_2 = \frac{\sqrt{\mathcal{I}((q - q_E)^2)}}{\sqrt{\mathcal{I}(q_E^2)}} \qquad l_\infty = \frac{\max|q - q_E|}{\max|q_E|}, \tag{1.1}$$

where q is the tracer distribution (depending on space and time) and q_E the exact distribution. $\mathcal{I}$ is the integral of the distribution over the sphere. Using these norms, we can compute an absolute error at a given resolution or the error for several resolutions to examine the convergence rate. A formal order of accuracy can also be obtained by using truncated Taylor series in the original equations. After removing the cancelling terms, the lowest order term gives the order of the scheme. However, it does not provide any guarantee about accuracy at a given resolution, nor near discontinuities. For global models, the objective is rather to decrease the error at a given resolution, rather than improving the formal order of accuracy.
Another issue is that exact solutions are not always available. In real-case scenario, we only have access to the results of other models and observational data. Observations are non-uniform in space and in time (every few hours in practice). That is why a scheme is first tested on algebraic test cases representative of atmospheric transport, where the algebraic solution is known. Such tests are detailed in Chapter 2. Once the model has been validated on these test cases, 'real' data can be used, as shown in Chapter 4.

When choosing a numerical scheme, its stability should be examined carefully. For Eulerian formulations, we typically have a Courant-Friedrichs-Lewy (CFL) condition restricting the time-step to avoid numerical divergence during the simulation. On a regular latitude-longitude grid, this condition severely restricts the time-step due to the convergence of meridians at the pole. This issue led to different solutions, as explained in Section 1.3. A related desirable feature is the geometric flexibility, *i.e*, how dependent it is on the grid. Mass preservation is another important feature. If we consider the total dry air mass in the atmosphere, there are only very small variations due to physical processes so the lack of mass preservation leads to non-physical solutions. It is also important to preserve the mass of long-lived tracers such as stratospheric ozone. Highly reactive substances such as reactive chlorine must also preserve the sum of their compound.

The equations to solve for the transport are the air mass and tracer mass continuity equations[1]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0, \tag{1.2}$$

$$\frac{\partial \rho q}{\partial t} + \nabla \cdot (\rho q \mathbf{V}) = 0, \tag{1.3}$$

where ρ is the air density, q the tracer mixing ratio and $\mathbf{V}$ the winds vector field. If the grid and the time-step are identical for the air and tracer transport, then the tracer transport should default to the air transport when $q = 1$. Otherwise, this can lead to spurious mass creation or deletion, the so-called *mass-wind inconsistency*.

The transport scheme should preserve the shape of the tracer distribution. In particular, the scheme should be range-preserving and positivity preserving. It avoids negative ratio or ratio outside the physical range, which can cause issues with the chemistry or parametrization later on. Spurious extrema should also be avoided (monotonicity constraint). Also, the scheme should be non-oscillatory, meaning large gradients should not create 'wiggles' in the distribution.

[1] Vectors as written in bold.

When advecting several tracers, the transport scheme should preserve the different relations between species. It is especially important for long-lived species in the atmosphere. At least, linear relations should be preserved.

The usefulness of a numerical scheme depends for a large part on its computational cost and the computational power available. Since the first CTMs, the increase in resolution and in complexity of physical parametrizations led to new challenges. The evolution of computational power and architectures is explained in more details in Section 3.1. As we are now in an area of supercomputers, current and future algorithms must be designed with parallelism in mind to cope with the current architectures (going up to the petascale, and aiming for exascale). It is not sufficient to parallelize a CTM by simply separating the transport and chemistry, or by separating the transport of each tracer. The transport of one tracer should be designed itself with parallel efficiency in mind. In particular, balancing the accuracy and scalability is no obvious task: a decrease in accuracy can improve parallel efficiency. The gain in performances can then be used for finer resolutions as a way to compensate the initial loss of accuracy.

1.3 Historical perspective

Finite-difference

Historically, finite-difference were the first scheme to be studied. They adopt a grid-point approach, where data is discretized at a series of points on which the different quantities are evaluated. The spatial derivatives are estimated from a Taylor serie: using more terms leads to higher-order schemes but also increases the size of the stencil. Some examples are given in Table 1.1. Once the derivatives have been replaced by their approximations in the initial equations, it results into the finite-difference numerical scheme. It should be noted than Lele (1992) developed 'compact' finite-different schemes, which allows for high-order schemes. They may prove more interesting for parallelization, as argued by Dixon and Tan (2003) in the context of convection-diffusion.

To apply these schemes to the sphere, the first idea was to use a map projection for a part of the sphere, and then use the Cartesian coordinates on this projection. However, this strategy is difficult to apply to the whole sphere, so spherical curvilinear coordinates were used in practice. Grids other than the regular latitude-longitude grid were considered at the time, but were eventually abandoned for different reasons. However, the current trend is to examine these alternative grids again with different schemes to avoid the limitations of the regular latitude-longitude grid. Here, we only highlight the reasons why there were abandoned as their features are detailed in Section 1.5. A combination of conformal projections with two polar stereographic projections was considered by Phillips (1957). This grid is of the 'composite' type but was not adopted due to conservation and noise concerns. Another approach used a coordinate system for each side of a polyhedron, which was used by Sadourny (1972) on a cube. However, it was found that noise appeared at the boundaries of the cube. Sadourny et al. (1968) used an icosahedral grid but it was found that higher-order schemes led to noise issues. To solve the 'pole issue', Kurihara (1965) used a reduced grid but different phase errors were found. Another solution to the pole issue is to use spatial filters on the regular latitude-longitude grid, an approach which proved to be successful and is still in use today.

Table 1.1: Examples of first and second-order approximation of the partial derivative $\frac{\partial q}{\partial x}$.

Type	order	Approximation
forward	1	$(q_{i+1} - q_i)/\Delta x$
	2	$(-3q_i + 4q_{i+1} - q_{i+2})/(2\Delta x)$
backward	1	$(q_i - q_{i-1})/\Delta x$
	2	$(q_{i-2} - 4q_{i-1} + 3q_i)/(2\Delta x)$
centered	2	$(q_{i+1} - q_{i-1})/\Delta x$

Spectral models

Spectral models are an alternative approach to grid-point schemes. They were extremely popular between 1990 and 2000, especially for operational NWP. The idea for spectral models can be highlighted if we consider a 1D function. If the function is periodic, it can be represented as a infinite Fourier serie, which is truncated in practice. With this Fourier representation, computing the derivatives is easily done from the Fourier coefficients. To adapt this 1D approach to the sphere, the trigonometric base functions are replaced by base functions one the sphere called 'spherical harmonics'. For linear advection, a certain type of truncation allows for an isentropic representation in spectral space, therefore bypassing the 'pole issue'. It should be noted that truncating the Fourier series implies a lower limit on resolution. For example, in the European Centre for Medium-Range Weather Forecasts (ECMWF) model, the truncation requires a resolution of 40km (Buizza et al. (2008)).

The efficiency of spectral models can be improved with several methods. Using reduced grids (see Section 1.5), the number of points can be reduced up to 30%. One example is given by Hortal and Simmons (1991). Initially, spherical harmonic transforms were too expensive to be used in practice: as their cost is in $O(K^3)$, where K is the spectral truncation. The development of fast transforms (FFTs) led to an affordable cost of $O(K^2 \log K)$ instead. In dynamical cores, spectral models can be improved by coupling semi-Lagrangian schemes (see below) for transport and spectral transform methods. It allows to switch from a Gaussian grid to a linear one. A Gaussian grid is a grid with enough latitude circles ($K \geq 3T+1$) so we can compute the product of two fields without aliasing[2]. A linear grid only has $K \geq 2T + 1$ as a constraint. It ensures we can switch to the spectral space and still recover the initial values if we take the inverse transformation. More details about this and spectral models can be found for example in Hack and Jakob (1992).
However, parallelizing spectral models is difficult, mostly due to the Fast Fourier Transforms (FFTs) and Legendre transforms. An example of a parallel FFT algorithm can be found for example in Swarztrauber (1987). An analysis and comparison of parallel methods for the spectral transform can also be found in Foster and Worley (1997).

[2] *i.e*, transforming each field into the Fourier space, multiply the result and switch back to the initial space.

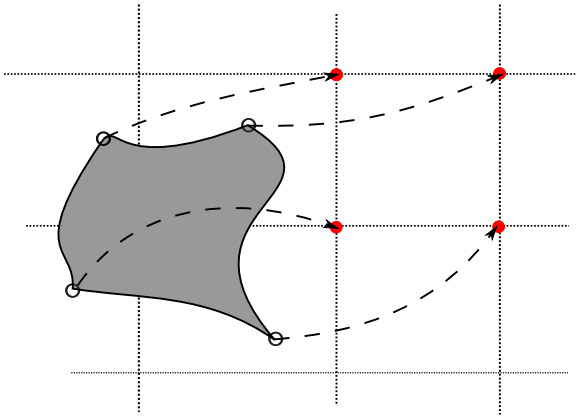


Figure 1.1: Illustration of a time-step in a forward semi-Lagrangian scheme.

Semi-Lagrangian

In an Eulerian formulation, we use either the advective or the flux-form of the equations. For the continuity equation, there are noted respectively:

$$\frac{\partial \rho}{\partial t} + \mathrm{div}(\rho \mathbf{V}) = 0, \tag{1.4}$$

$$\frac{\partial \rho}{\partial t} + \mathbf{V} \cdot \mathrm{div}(\rho) = 0. \tag{1.5}$$

Lagrangian models use this formulation instead:

$$\frac{D\rho}{Dt} = 0, \tag{1.6}$$

where $\frac{D}{Dt} = \frac{\partial}{dt} + \mathbf{V} \cdot \mathrm{div}$ is the total Lagrangian derivative. In an Eulerian formulation, we consider the cells of a fixed grid and examine the change of the tracer concentration in each cell. In a Lagrangian formulation, we follow moving cells during all the simulation. It allows for very accurate results but the resolution is not uniform in the case of deformational flows. This led to the development of semi-Lagrangian schemes, where we track the same cell for a given time-step and then interpolate it on the fixed grid. This is illustrated on Fig.1.1. This strategy is for 'forward' schemes but the most common version is 'backward', where the initial position of the cell is computed from the trajectories.

The main advantage of semi-Lagrangian schemes is their weaker stability criteria. According to Smolarkiewicz and Pudykiewicz (1992), it can be written in 1D as:

$$||\frac{\partial v}{\partial x}||\Delta t < 1 \tag{1.7}$$

and can be interpreted geometrically as the fact that trajectories do not cross. This is less restrictive

Centre	Reference
Canadian Meteorological Center (CMC)	Côté et al. (1998)
ECMWF	Ritchie and Temperton (1995)
United Kingdom Meteorological Office	Davies et al. (2005)
Danish Meteorological Institute	Nair and Machenhauer (2002)
China Meteorological Association	Chen et al. (2008)
Meteo-France	Bubnová et al. (1995)
Hydrometcentre of Russia	Tolstykh (2001)
Japanese Meteorological Agency (JMA)	JMA (2013)

Table 1.2: Operational models using semi-Lagrangian approaches

than the CFL condition (CFL) generally used for 1D Eulerian models:

$$\max\left(\frac{u\Delta t}{\Delta x}\right) < 1. \tag{1.8}$$

In practice, this gain in the time-step for grid-point semi-Lagrangian can be used to improve the accuracy of the model (with a finer resolution or improved parametrizations). This led to a wide adoption for operational models (see Table 1.2 for a list of examples). Yet, semi-Lagrangian do not preserve mass inherently. To address that issue, mass fixers can be used to adjust the scheme such as mass is preserved globally. Examples of this method are given in Priestley (1993); Moorthi et al. (1993); Williamson and Rasch (1994). Another way is to use cell-integrated methods which are conservative by nature, an approach explained in Section 1.3. Another disadvantage of the semi-Lagrangian formulation is the interpolation step: not only is it expensive, but it creates numerical diffusion.

Finite-volume schemes

Conservation properties and scalability are two current trends for the next-generation climate and weather models. As such, spectral and semi-Lagrangian methods are now being reconsidered: the former as they are not easily scalable, the latter as they are not inherently conservative. Finite-volume schemes are well-suited to solve conservation laws, *i.e.*, the air and tracer continuity equations for CTMs. Finite-volume schemes are also very useful for dynamical cores, which solve the conservation of total energy, angular momentum and entropy.

Finite-volume schemes are derived by integrating the flux-form equations over a control volume and in time. In the case of CTMs, the air or tracer mass is then updated by the fluxes through the control volume during a time-step. Choosing a proper control volume remains an open question but finite-volume schemes can be adapted to a wide range of geometry. A review of most common grids for atmospheric transport is presented in Section 1.5. Most of the current finite-volume schemes have an order equal or lower than three. It is possible to use higher-order schemes such as WENO (Jiang and Wu (1999)) but they require a large number of cells, which will impact scalability. It should also be noted that finite-volume schemes have a wide range of application and the choices made for Pangolin are very specific to the atmospheric transport. For a more general review, the reader can refer to LeVeque (2002).

Finite-volume schemes can fall into two categories: flux-form and semi-Lagrangian. Flux-form schemes are based on the flux-form equation, in our case Eq.(1.5), while semi-Lagrangian schemes are based on the Lagrangian form, shown in Eq.(1.6). It can be shown that these two formulations are equivalent (see Machenhauer et al. (2009)). The difference between the two stems from the approximations made for each approach, such as the mass integral and the trajectories.

Semi-Lagrangian

While in 'traditional' semi-Lagrangian methods, values are interpolated from grid-points, the finite-volume version advects average values over the cell area. The discretization of the equations leads to the following scheme: the average value over the regular cell can be computed from the integrated value over the irregular departure cell. So the main difficulty is to compute this integrated value over the departure cell explicitly. These schemes are referred to as Departure Cell-Integrated Semi-Lagrangian (DCISL).

To solve the 3D continuity equation, we can simplify the problem by considering that horizontal and vertical advection are separated. This means that we use vertically integrated variables and only focus on the horizontal advection. It is also possible to consider a full 3D transport for semi-Lagrangian finite-volume schemes but this increases the complexity, especially for mass preservation. 2D schemes can be divided into two categories: fully 2D and 'cascade' schemes.

We first detail fully 2D schemes. To approximate the integral, the first step is to define the geometry of the departure cell, which is difficult to compute efficiently due to the spherical geometry. We present some solutions in 2D but the extension to the spherical geometry is presented in Machenhauer et al. (2009). In 2D, one solution is to find the four departure points from the trajectories[3] and link them with straight lines to create a polygon (see Rančić (1992)). Another possibility is to use quadrilaterals with sides parallel to the x- and y-axis as in Nair and Machenhauer (2002). The second step is to reconstruct the distribution of the transported variable in this new Lagrangian cell. Common reconstructions use constant, first-order or even higher-order polynomials.

Another approach is to use a 'cascade' scheme which splits the 2D integral into two 1D integrals. For that, an intermediate grid must be constructed. The algorithm is as follows: first, the integral is computed from the Eulerian cells to the intermediate grid. Then the second integral is computed from the intermediate grid to the Lagrangian cells. More details and illustrations about this scheme can be found in Purser and Leslie (1991). Cascade schemes have this advantage that they can be easily extended to the 3D case.

Flux-form

Another finite-volume approach is to consider instead that the mass is modified according to the fluxes passing through the walls of cells. As for DCISL schemes, there are full 2D schemes as well as operator-splitting schemes, where 1D operators are applied successively. Furthermore, flux-form schemes can easily be extended to 3D using operator splitting than their semi-Lagrangian counterpart. We present in more details this approach in the next section.

[3]The trajectories can be computed either starting from the Eulerian grid ('backward') or to the Eulerian grid ('forward'). A common trajectory algorithm is presented by Staniforth and Côté (1991).

1.4 Flux-form finite-volume schemes

Here, we focus on solving the equations (1.2) and (1.3) using flux-form finite-volume schemes. The first equation describes the conservation of 'moist air', that is, atmospheric air with all of its chemical components. The second equation describes the conservation of a tracer, defined by the ratio of the tracer mass over the air mass in a given volume. The accuracy of moist air is especially important in meteorology as the flow of air mass determines pressure and weather dynamics. Furthermore, the conservation of linear relations between chemical tracers is especially important for CTMs.

To derive the flux-form version, we consider 2D Eulerian cells without making assumptions on their shape. The cell area is noted $\mathcal{A}$ and its boundary $\partial\mathcal{A}$. First, we integrate both the continuity and tracer conservations (1.2) and (1.3) over a cell grid:

$$\frac{\partial m}{\partial t} = \frac{\partial}{\partial t}\int_{\mathcal{A}} \rho = \int_{\partial\mathcal{A}} (\rho \mathbf{V} \cdot \mathbf{n}\, \mathrm{d}S), \tag{1.9}$$

$$\frac{\partial mq}{\partial t} = \frac{\partial}{\partial t}\int_{\mathcal{A}} \rho q = \int_{\partial\mathcal{A}} (\rho q \mathbf{V} \cdot \mathbf{n}\, \mathrm{d}S). \tag{1.10}$$

Then, integrating over a time-step yields:

$$m^{n+1} = m^n + \int_{t_n}^{t_{n+1}} \Big(\int_{\partial\mathcal{A}} \rho \mathbf{V} \cdot \mathbf{n}\, \mathrm{d}S \Big) \mathrm{d}t, \tag{1.11}$$

$$[mq]^{n+1} = [mq]^n + \int_{t_n}^{t_{n+1}} \Big(\int_{\partial\mathcal{A}} \rho q \mathbf{V} \cdot \mathbf{n}\, \mathrm{d}S \Big) \mathrm{d}t, \tag{1.12}$$

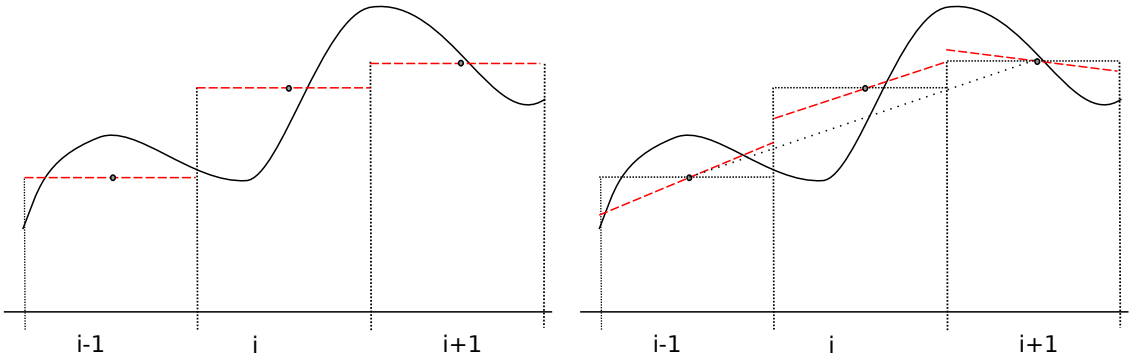
where m^{n+1} is the average air mass in the considered cell at time t_n and $[mq]^{n+1}$ the average tracer mass at the same time. At this point, no hypothesis were made about the shape of the cells. As an illustration, we consider the 1D version of these equations on a regular latitude-longitude grid. If we consider the i-th cell, with its borders noted $i-1/2$ and $i+1/2$, we have:

$$m_i^{n+1} = m_i^n + U_{i-1/2} - U_{i+1/2}, \tag{1.13}$$

$$[m_i q_i]^{n+1} = [m_i q_i]^n + F_{i-1/2} - F_{i+1/2}, \tag{1.14}$$

where $U_{i-1/2}$ is the flux of air mass passing through the cell wall $i-1/2$ and $F_{i-1/2}$ is the flux of tracer mass passing through the same cell wall. However, it should be noted that other geometries are possible and flux-form can be used in a variety of grids. In particular, Section 1.4 explains how it was adapted to our non-regular grid.

The accuracy of the scheme depends on the accuracy of the fluxes computation, namely U and F. While the air mass fluxes U can be computed easily knowing the winds and the size of the cell borders, there is no exact formula for F as we only have discretized data at our disposal. It is then approximated as the mean ratio $\hat{q}$ value passing through the cell border times the air mass flux: $F = \hat{q}\,U$. Therefore, we have to reconstruct the tracer mass distribution in the cell to be able to integrate it. Different reconstruction are highlighted below in the 1D case. The extension to 2D is detailed in Section 1.4.



(a) Godunov's method.

(b) van Leer's scheme I: the slope is simply computed by finite differences.

Figure 1.2: Constant and first-order reconstruction of the grid distribution (dark line). Cell average are shown as dotted lines and the reconstruction is the dashed red line.

Sub-grid distributions

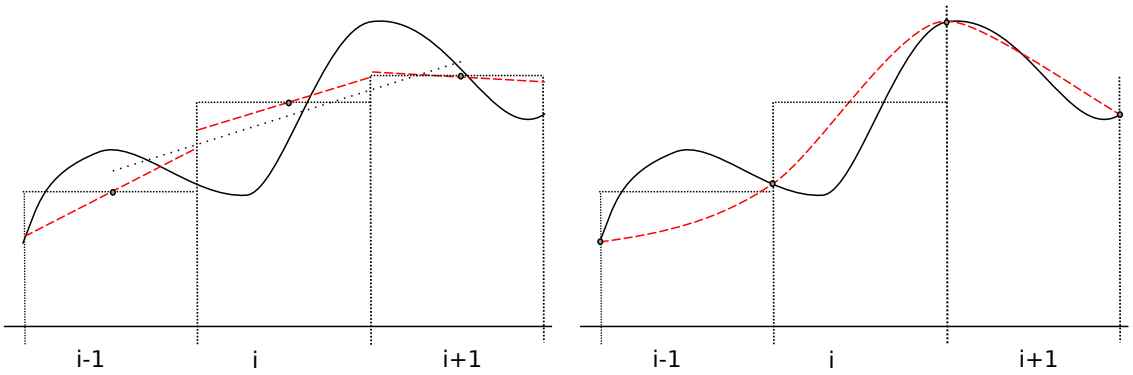
First-order

To reconstruct the mass distribution, polynomials are the more common representation used in practice. The first proposal was made by Godunov et al. (1961) where the distribution in each cell is reconstructed as a constant function. It is a very simple requirement to preserve the shape of the distribution but it is also very diffusive. To improve on that, the use of a first-order polynomial was proposed by van Leer (1977). As the reconstruction is locally linear for each cell, there is no continuity between the cells. Computing the slope can be done for example with finite-differences (scheme I in van Leer (1977)) or with a least-square fit (scheme III in van Leer (1977), also called *slopes scheme* in the GCM community). The idea for Godunov's scheme and the two van Leer's schemes is highlighted on Fig. 1.2(a), 1.2(b) and 1.3(a) respectively.

The scheme was subsequently improved in van Leer (1979), who proposed a monotonicity algorithm to reduce numerical oscillations. The slope is limited to prevent that the new distribution in a cell does not exceed the mean value of the adjacent cells. This is illustrated in Fig. 1.5.

PPM and Prather's scheme

It is also possible to use quadratic polynomials to reconstruct the mass distribution, as shown by van Leer (1977) (scheme V in the original paper). Van Leer only used finite differences to evaluate the coefficients of the polynomial but this led to the Piecewise Parabolic Method (PPM). A first version, proposed by Laprise and Plante (1995), defined the coefficients of the parabola by enforcing mass preservation in the current cell and its two neighbours. A second version was proposed by Colella and Woodward (1984), where the west and east cell values are used as initial conditions. This allows the reconstructed distribution to be continuous across cell borders and the result is shown in Fig. 1.3(b). Polynomials are only one choice among many, as long as the chosen function can preserve mass. For example, Xiao (2002) used rational functions and Zerroukat et al. (2002) used parabolic splines.



(a) Van Leer's scheme III: the slope is computed with a least square method.

(b) PPM: the reconstruction is smooth and uses the west and east value in the cell (shown as dots).

Figure 1.3: First and second-order reconstruction of the grid distribution (dark line). Cell average are shown as dotted lines and the reconstruction is the dashed red line.

To ensure monotonicity, a simple filter was proposed by Colella and Woodward (1984). Other filters exist, for example in Lin and Rood (1996), the filter was modified as to prevent undershooting only. An example for more advanced filters can be found in Zerroukat et al. (2005), which aims at preventing the clipping of peaks while removing grid-scale noise anyway. The PPM scheme can be extended to 2D, as shown by Rančić (1992) but is quite expensive from a computational point of view. Furthermore, using 1D filters in this case does not remove all negative values so additional filters are needed.

Another well-known second-order reconstruction is the Second-Order Moments (SOM) scheme introduced by Prather (1986). The reconstruction is written as:

$$\begin{aligned} f(x,y,z) = a_0 &+ a_1 K_x(x) + a_2 K_y(y) + a_3 K_z(z) \\ &+ a_4 K_{xx}(x) + a_5 K_{yy}(y) + a_6 K_{zz}(z) \\ &+ a_7 K_{xy}(x,y) + a_8 K_{xz}(x,z) + a_9 K_{yz}(y,z), \end{aligned}$$

where the K are polynomial orthogonal functions. In particular, K_x is a first-order polynomial in x, K_{xx} second-order and K_{xy} a first-order in x and y. This scheme is more accurate compared to the previous schemes and it preserves second-order moments (the 6 coefficients a_i with $i = 4 \dots 9$). However, it is also more computationally expensive and needs to store 10 tracers (the a_i, $i = 1 \dots 10$).

2D flux-form schemes

The simplest approach to extend the 1D schemes mentioned previously to the 2D case is to simply combine the two 1D operators simultaneously. For simplicity, we note $Q^n = [mq]^n$ the tracer mass at the time-step n. The update of tracer mass during one time-step is then:

$$Q^{n+1} = Q^n + H_{zonal}(Q^n) + H_{merid}(Q^n), \tag{1.15}$$

where H_{zonal} (resp. H_{merid}) is the 1D operator corresponding to the zonal (resp. meridional) advection. This approach is stable for 1D reconstruction as shown by Leonard et al. (1996) but not for the 2D reconstruction. Therefore, we have to apply each operator sequentially:

$$\widehat{Q}_{zonal} = Q^n + H_{zonal}(Q^n), \tag{1.16}$$

$$Q^{n+1} = \widehat{Q}_{zonal} + H_{merid}(\widehat{Q}_{zonal}). \tag{1.17}$$

Unfortunately, this approach introduces a directional bias due to its non-symmetrical nature. A simple way to avoid that is to alternate the operators at each time-step, also called operator-splitting. For example, the first time-step will use H_{zonal} then H_{merid} while the next will apply H_{merid} then H_{zonal}. Another solution is to define a symmetric scheme:

$$Q^{n+1} = Q^n + H_{zonal}\Big(\frac{Q^n + \widehat{Q}_{merid}}{2}\Big) + H_{merid}\Big(\frac{Q^n + \widehat{Q}_{zonal}}{2}\Big). \tag{1.18}$$

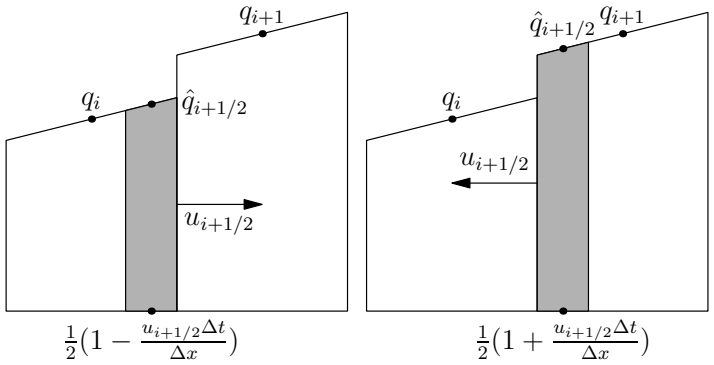
Yet, all these schemes introduce a splitting error: a spatially uniform field with divergence-free winds will not be preserved, as the field will be deformed in one direction by the scheme. Lin and Rood (1996) proposed an algorithm to address that issue by compensating the error due to the deformation.

Operator-splitting approaches are computationally efficient and easy to extend to several dimensions. But if we examine them from a semi-Lagrangian point-of-view, we cannot integrate on the exact departure area exactly due to an inconsistency due to the splitting. To try to solve that issue, the use of fully 2D fluxes was studied (see for example Rasch (1994); Dukowicz and Baumgardner (2000)). But the fluxes in the cross-directions must be written explicitly, contrary to the operator-splitting approach, where there are 'automatically' accounted for. It is also a more expensive approach.

Scheme chosen in Pangolin

In Pangolin, our goal is not to invent a new scheme, but rather implement an already-existing but efficient scheme on our custom grid (presented in Section 1.6). The most important requirements for our scheme is its flexibility to adapt to our non-regular grid, mass preservation and scalability on parallel computers. We have chosen a flux-form finite-volume approach as it preserves tracer mass globally and locally. It is also considerably easier to parallelize than the semi-Lagrangian alternative. The parallelization issue is then reduced to a domain decomposition problem detailed in Section 3.2.

The current version of Pangolin is bi-dimensional but a future version should deal with 3D advection. The operator-splitting approach was chosen for simplicity reasons. Among the different flux-form schemes, the Van Leer scheme I was chosen. While it may be less accurate than higher-order schemes such as PPM, higher-order reconstruction increases data volume in a parallelization based on a message-passing approach (see Section 3.2). We strive to reduce communications, so a first-order reconstruction in each direction is a good compromise. On top of that, in an operator-splitting algorithm, the formal order cannot be greater than 2. As shown in the test results of Section 2.2, we aim at achieving the same accuracy than higher-order schemes with finer resolutions. Finally, it should be remembered that Pangolin is essentially a parallel framework using a custom quasi area-preserving grid, on which several schemes can be plugged. The results showed here



(a) Positive winds (b) Negative winds

Figure 1.4: van Leer scheme for positive and negative winds. The distribution of the tracer is shown as broken lines. The grey area is the quantity of tracer passing through the interface during a time-step.

result from a given transport scheme but this scheme can be replaced according to the needs of the modellers.

We now summarize the different steps of the scheme. The equations to solve are Eq. (1.2) and Eq. (1.3). In 1D, they are discretized to arrive to the form presented in Eq. (1.13) and (1.14). Eq. (1.14) can be omitted for non-divergent flows since the divergence of the mass fluxes is null and the mass inside a cell is constant: $m^{n+1} = m^n$.
To compute the fluxes $f_{i+1/2}$, we have to find the mean value $\widehat{q}_{i+1/2}$, given by the van Leer I scheme:

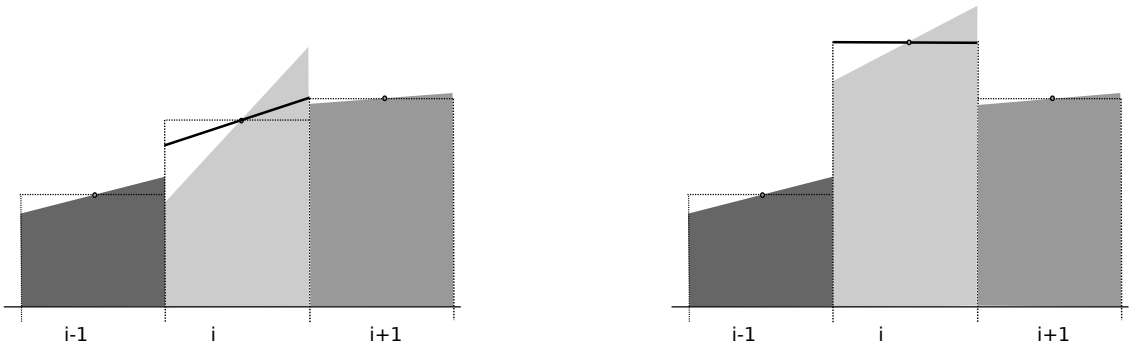
$$\widehat{q}_{i+1/2} = \begin{cases} q_i + \left(1 - \frac{u_{i+1/2}\Delta t}{\Delta x}\right)(\delta q)_i & \text{if} \quad u_{i+1/2} > 0, \\ q_{i+1} - \left(1 + \frac{u_{i+1/2}\Delta t}{\Delta x}\right)(\delta q)_{i+1} & \text{otherwise,} \end{cases} \tag{1.19}$$

This reconstruction is illustrated on Fig. 1.4, where the broken lines represent the tracer distribution for cells i and $i+1$. Depending on the wind direction, the tracer mass passing through the interface $i+1/2$ either comes from cell i when $u_{i+1/2} > 0$ or $i+1$, when $u_{i+1/2} < 0$. Computing the slope can be done with finite-difference: $(\delta q)_i = \frac{1}{2}(q_{i+1} - q_{i-1})$. We use in practice the slope limiter of van Leer (1979):

$$(\delta q)_i = \min\left(\frac{1}{2}|q_{i+1} - q_{i-1}|, 2|q_{i+1} - q_i|, 2|q_i - q_{i-1}|\right) \times \text{sign}(q_{i+1} - q_i), \tag{1.20}$$

if q_i lies in between q_{i-1} and q_{i+1}, and $(\delta q)_i = 0$ otherwise. An example of the limiter is shown on fig. 1.5.

To extend the 1D algorithm to 2D, we use the simplest operator-splitting algorithm: first, the tracer and air mass are advected in the zonal direction, then in the meridional direction as in Eq. (1.16) and (1.17).



(a) The limiter ensures the value is cell i is not greater than the average of its two neighbours (dotted line).

(b) The slope in cell i is set to 0 as the cell average is a local maximum.

Figure 1.5: van Leer's slope limiter in the 1D case. The limited slope is shown as a dark line, while the reconstruction is shown as broken lines.

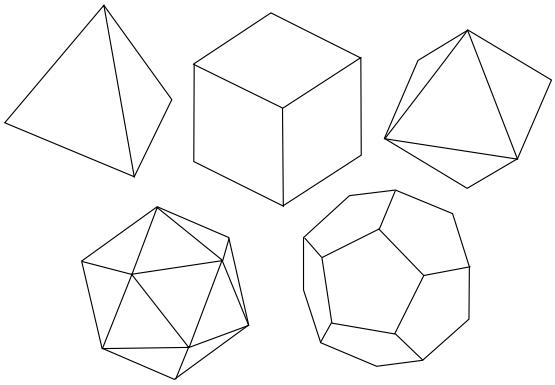


Figure 1.6: The five platonic solids. From left to right and top to bottom: tetrahedron, cube, octahedron, dodecahedron, icosahedron.

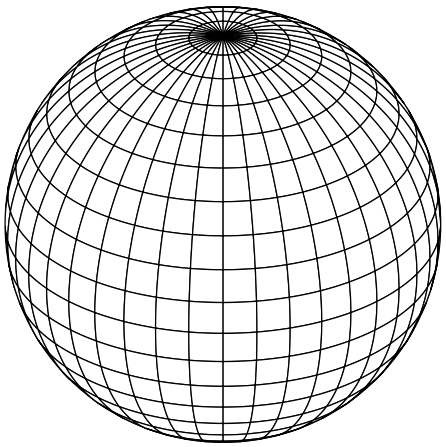


Figure 1.7: Regular latitude-longitude grid with 20 latitudes.

1.5 Choosing a grid

Introduction

There are several ways of gridding the sphere and it is still an active topic of research. In general, the choice of the grid impacts the discretization and the scheme. More precisely, it can impact key properties of the scheme such as mass conservation or accuracy. In some cases, the grid structure can even generate noise. On top of that, the sphere presents unique challenges: finding an uniform resolution without singularities or clustering is not trivial. The choice of the grid also impacts computational efficiency and scalability on parallel architectures.

It can be shown there are only five ways to tile the sphere uniformly. For that, we have to use one of the five platonic solids (shown in Fig. 1.6) and do a gnomonic projection of each face onto the sphere. However, the resulting grids would be too coarse as platonic solids have at most 20 faces. This leads us to make compromise to tile the sphere: we cannot have uniformity and orthogonality and to be free of singularities. Singularities are not necessarily a drawback but some grids, especially the regular latitude-longitude grid, have cell clustering near these singularities. Then CFL conditions in Eulerian formulations imply poor scaling when the resolution increases. For that reason, uniform or quasi-uniform grids are preferred. Orthogonality is another interesting property but is less important for CTM than for dynamical cores (see Staniforth and Thuburn (2012) for more information about this and other impacts of grids on dynamical cores). Yet, it helps making neighbour computation and thus flux computation easier in our case. Another possible constraint is the geometry of the cells. Quadrilaterals are the most common cells in practice but some grids can use more 'exotic' shape, which can restrict the choice of the scheme.

Here, we highlight different categories of grids with their pros and cons, while the grids themselves are explained in more details in Section 1.5. We can generate orthogonal quadrilateral grids with a conformal projection but it implies singularities on the sphere and some resolution clustering. Regular latitude-longitude and conformally projected cubic grids are two examples of that. Singularities can be avoided with composite grids while achieving quasi-uniformity. The downside is the increased cost of the overlapped regions. Overlaps also impact scalability significantly due to the increase communication in these regions. It is also not easy to preserve mass accurately and efficiently on the overlapped region. If we give up on orthogonality, we can avoid both clustering and overlaps and still use quadrilaterals. The gnomonically projected cubic grid is one example of that. Other approaches use an orthogonal dual (for example, by dividing the quadrilaterals into 'kite'-shaped polygons). If we give up quadrilaterals, we can manage the quasi-uniformity and have an orthogonal dual with icosahedral grids.

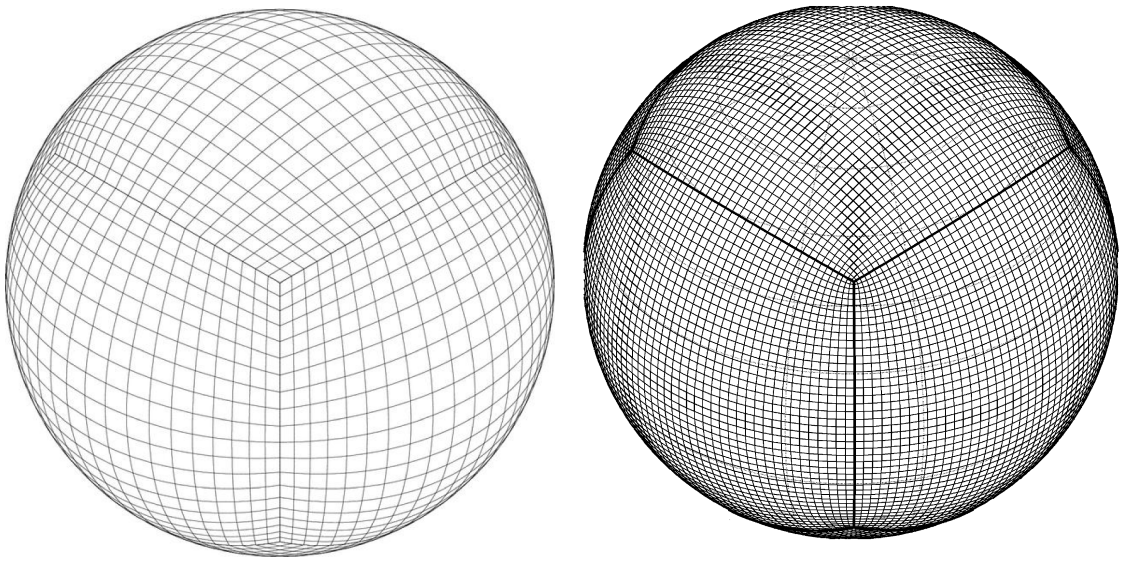
Review

Latitude-longitude

This grid is the most popular as it has a rectangular structure, is orthogonal and is very natural to use. However, the convergence of meridians at the poles (see Fig. 1.7) is an important problem. It imposes a severe constraint on the time-step in Eulerian formulations. The resolution is anisotropic so there is a decrease in computation efficiency. It also leads to poor scaling concerning communications around the poles. To deal with the time-step constraint, this grid is now often used with semi-Lagrangian approaches, which are less sensible to stability requirements. As an illustration, the National Center for Atmospheric Research (NCAR) in their CAM3 model implemented a flux-form semi-Lagrangian model. An improvement of the latitude-longitude grid was proposed by Kurihara (1965), where the number of cells at a latitude decreases as the latitude gets closer to the pole (see Section 1.6).

Cubed-sphere grids

Cubed-sphere grids can fall into two categories. The first uses the gnomonic projection projection and was proposed by Sadourny (1972): Cartesian coordinates are used on each face of the cube and then projected gnomonically. This approach used finite-differences and did not enjoy much popularity at the time but it generated renewed interest with Rančić et al. (1996) or Harris et al.



(a) Gnomonically projected (b) Conformally projected

Figure 1.8: Equi-distant cubic grids (from Putman and Lin (2007)).

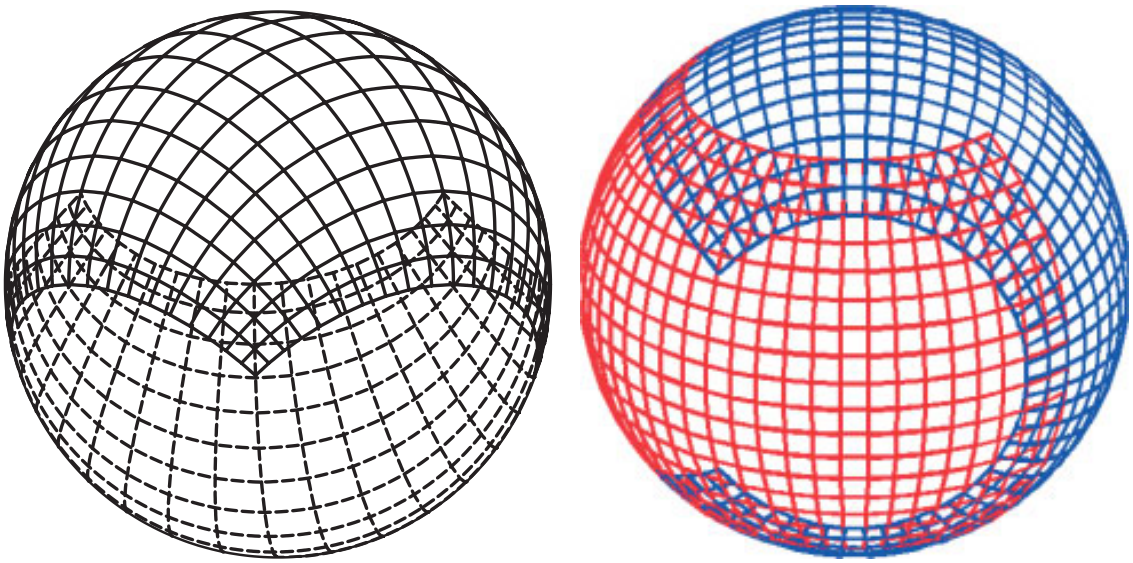
(2011). The equi-distant version of this grid uses an uniform Cartesian grid on each face, leading to a non-uniform grid once projected. The equi-angular version of this grid does the opposite: uniformly spaced meridians are projected, resulting into an non-uniform Cartesian grid. An illustration is shown in Fig. 1.8(a).

The ratio of the maximum grid length over the minimum is 5.2 for the equi-distant version and 1.3 for the equi-angular version. As a reference, this can be compared to the composite grid of Phillips (1962) which claims a ratio of 1.2 (see below). Each face of the grid is free of singularities but is not orthogonal. On the sphere, the eight vertices of the cube are still singularities though. If the uniformity requirement is relaxed, some orthogonality can be gained. The balance was studied by Putman and Lin (2007). If a conformal projection is used, each face is free of singularities and also preserves orthogonality. Yet the eight vertices of the cube are still singularities and resolution clustering can be seen around them as shown on Fig. 1.8(b).

Composite grids

One way to avoid singularities is to 'patch' several, partially overlapping, grids to create a composite (or overset) grid. Phillips (1957) proposed a grid separated into a tropical belt and two square polar stereographic grids. The ratio maximal/minimal grid length is then 1.373 but can be reduced to 1.2. This grid was improved by Browning (1989) by removing the tropical belt, leading to two overlapping stereographic coordinates, one on each hemisphere. This is illustrated on Fig. 1.9(a). In this configuration, the previous ratio is then increased to 2.

Another composite grid is the Yin-Yang grid introduced by Kageyama and Sato (2004), which consists of two latitude-longitude grids 'sewn' together (see Fig. 1.9(b)) and normal to each-other. This grid is quasi-uniform without singular points but possesses an overlapping region for which



(a) Two-patch grid with stereographic coordinates for each patch

(b) Yin-Yang grid

Figure 1.9: Composite grids (taken from Williamson (2007)).

interpolation is needed. Global mass conservation can be achieved using the algorithm presented in Peng et al. (2006) which ensures the fluxes at the boundary are identical on each 'patch'. The Yin-Yang grid is almost as efficient as the grid of Phillips (1957) but is easier to use in practice. As an example of a practical application, a forecasting model using the Yin-Yang grid is being developed at the CMC (see Qaddouri and Lee (2011)).

Icosahedral grid

Icosahedral grids are part of a larger set of grids called geodesic. The idea is to approximate the surface of the Earth by refining a polyhedron. In practice, a icosahedron is the most common choice, thus leading to icosahedral grids. The Deutscher Wetterdienst (DWD) uses this kind of grid, as shown in Majewski et al. (2002). There are different ways to generate the grid for a given resolution. The easiest way is to start with the icosahedron and subdivides each triangle until the desired resolution is achieved. Tomita et al. (2001) proposed a way to improve the grid using spring dynamics. By default, the grid points do not correspond to the center of gravity of the corresponding control volumes. They suggested to link the grid points with springs in order to place them at the equilibrium. Their approach helps to reduce the noise due to numerical integration.

However, it seems harder to use high-order finite-difference approximations on this grid, as well as for finite-volume schemes. Giraldo (2000) proposed a Lagrange-Galerkin finite element approach to solve the issue. Another related grid is the hexahedral grid used in the spectral element dynamical core of Giraldo and Rosmond (2003). The grid is generated by subdividing each face of an hexahedron (polyhedra with six faces) into quadrilaterals with the desired resolution and project them on the sphere.

Other grids

The Fibonacci grid is one example of the more exotic grids and was developed by Swinbank and Purser (2006). Its advantages are its uniformity and isotropy. An illustration is shown on Fig. 1.10. The grid is constructed with a Delaunay triangulation using the grid points. When examining the dual mesh, most of the cells are irregular hexagons but there are some pentagons and heptagons as well. The grid itself possesses analytical features which are exploited to reduce the memory footprint and improve scalability.

1.6 Pangolin grid

Our motivation for the model is to solve the 'pole issue' mentioned previously in an Eulerian context as we focus on flux-form finite-volume approaches. For that, we have chosen to use a custom reduced grid. Orthogonality is lost and the poles are two singularities but we achieve a rectangular structure and have quasi-uniformity.

Reduced grids

Reduced latitude-longitude grids were examined in order to solve the clustering around the poles in the regular latitude-longitude grids. The idea is to decrease the number of cells as the latitudes get closer to the pole to have an uniform grid. The first reduced grid was proposed by Kurihara (1965) for finite-difference schemes, on which the number of cells at colatitude i is $4(i-1)$ (see Fig. 1.11). Cell width (or longitudinal spacing) is constant for a given colatitude and the cell height (or latitudinal spacing) is also constant. Rasch (1994) proposed a different approach: starting from the Equator, we go towards the pole. If cells are 'too close' to each other, there are merged two by two. The criteria for the merge in the paper is the distance between two consecutive cells should be less than 2/3 of the distance at the Equator. This can also be seen as a composite grid and is illustrated on Fig. 1.12.

Gaussian grids are one example of grids where the latitudinal spacing is not constant but uses the Gaussian quadrature instead. A Gaussian reduced grid was proposed by Hortal and Simmons (1991) for spectral methods. An illustration of the 'fully' reduced grid is shown in Fig. 1.13. It should be noted that the grid used in practice is quite similar to the reduced version in Hortal and Simmons (1991) but with a different number of cells nevertheless (see for example the ECMWF grids). One example of practical use is the dynamical core developed by the JMA in their Global Spectral Model (see JMA (2013) for more details).

Description

For the grid in Pangolin, we follow the same idea as Kurihara (1965): the cell height is constant and at a given latitude, cell width is constant. We will show our grid has a very lightweight description: only the number of latitudes on both hemisphere N_{lat} and the number of cells at the pole n_1 are needed. While Kurihara (1965) used one grid point for the pole itself, we have chosen not to have a cell centered on the pole. It should be noted that, while Kurihara (1965) set the middle of its cells at the Equator, we have chosen to place the walls of the cell at the Equator. Choosing the number of cells on the first (and last) colatitude is arbitrary. For our grid, we have chosen to take the regular latitude-longitude grid as a reference. If we consider square cells at the Equator, as

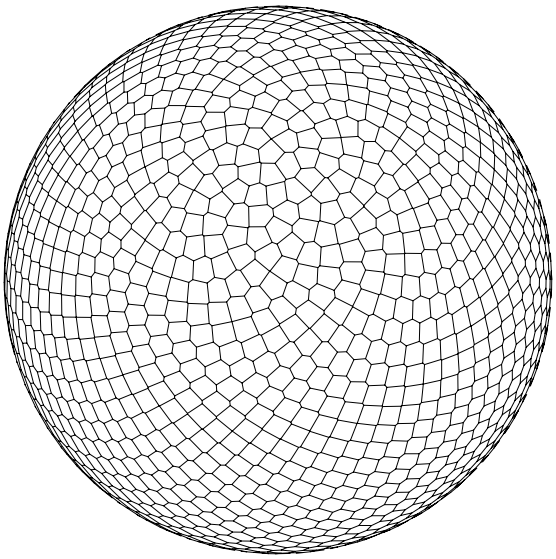


Figure 1.10: Fibonacci grid (from Swinbank and Purser (2006)).

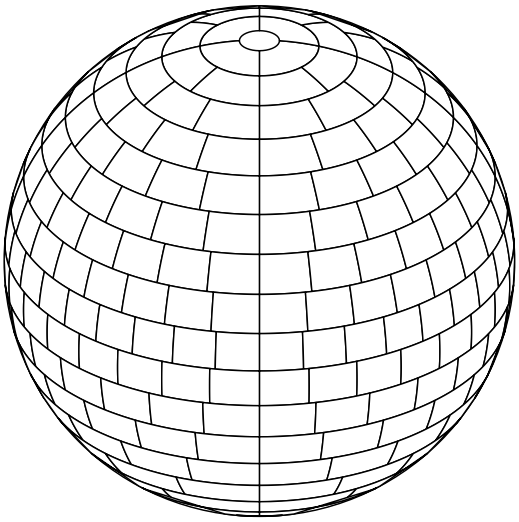


Figure 1.11: Reduced grid by Kurihara (1965) with 20 latitudes (reconstructed).

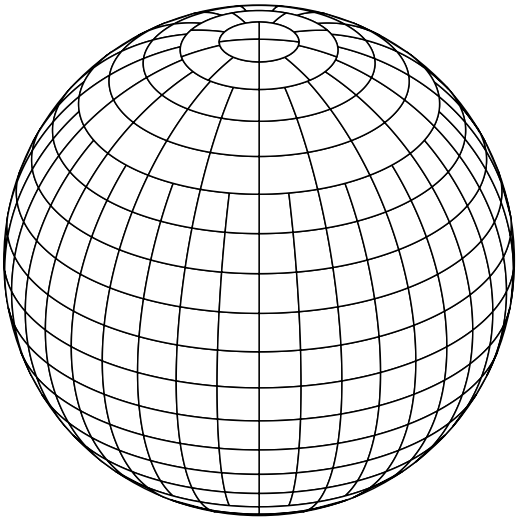


Figure 1.12: Reduced grid by Rasch with 20 latitudes and 40 longitudes at the Equator (reconstructed from Rasch (1994)).

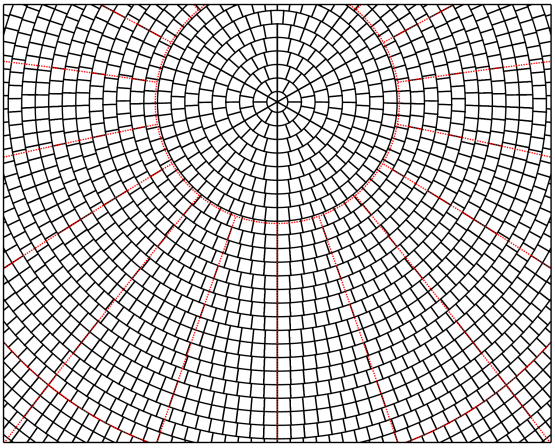


Figure 1.13: The 'fully' reduced grid by Hortal (reconstructed from Rasch (1994)). Due to the very fine resolution, the grid is represented with a Lambert equal-area projection, centered at latitude 80°.

regular latitude-longitude grid usually have, writing the area preservation formula leads to $n_1 = 3$ for small latitudinal spacing (see the proof in Appendix B.1).

To construct the grid, we only need to consider the Northern hemisphere, as the Southern hemisphere is its symmetric in respect to the Equator. We now derive the formula for the number of cells n_i at colatitude i. We no longer require square cells at the Equator as before. Let us note the cell (i, j) as being defined on $\Omega_{ij} = [\lambda_j, \lambda_{j+1}] \times [\phi_{i-1}, \phi_i]$. where $\phi_i = i\Delta\phi_i$ is the colatitude and $\lambda_{ij} = j\Delta\lambda_i$ the longitude. $\Delta\phi_i$ and $\Delta\lambda_i$ are the latitudinal and longitudinal spacings respectively. The area of cell (i, j) is also noted $\mathcal{A}_i$, as it does not depend on j. This leads to the following cell area formula in spherical coordinates:

$$\mathcal{A}_i = \iint_{\Omega_{ij}} r^2 \sin\phi \mathrm{d}\lambda \mathrm{d}\phi = r^2 \Delta\lambda_i \big(\cos(\phi_{i-1}) - \cos(\phi_{i-1} + \Delta\phi_i)\big).$$

Areas are preserved so $\mathcal{A}_i = \mathcal{A}_1$:

$$\Delta\lambda_i = \Delta\lambda_1 \frac{1 - \cos(\Delta\phi_1)}{2\sin\left(\frac{\Delta\phi_i}{2}\right)\sin\left(\left(i - \frac{1}{2}\right)\Delta\phi_i\right)}.$$

The number of cells at colatitude i can be defined by $n_i = \lfloor \frac{2\pi}{\Delta\lambda_i} \rfloor$ so:

$$\frac{n_i}{n_1} = \left\lfloor \frac{\Delta\lambda_1}{\Delta\lambda_i} \right\rfloor = \left\lfloor \frac{2\sin\left(\frac{\Delta\phi_i}{2}\right)\sin\left(\left(i - \frac{1}{2}\right)\Delta\phi_i\right)}{1 - \cos(\Delta\phi_1)} \right\rfloor.$$

Now let us assume $\forall i$, $\Delta\phi_i$ and $(i - \frac{1}{2})\Delta\phi_i$ are small enough so the sinus can be approximated by their angle:

$$\frac{n_i}{n_1} \approx \left\lfloor \frac{2\frac{\Delta\phi_i}{2}\left(i - \frac{1}{2}\right)\Delta\phi_i}{\frac{\Delta\phi_1^2}{2}} \right\rfloor = 2i - 1.$$

If the Southern hemisphere is constructed as the symmetric of the Northern hemisphere, this leads to:

$$n_i = nb\_cells(i) = \begin{cases} 3(2i-1) & \text{if } 1 \le i \le N_{\text{lat}}/2, \\ nb\_cells(N_{\text{lat}} - i + 1) & \text{otherwise.} \end{cases} \tag{1.21}$$

It follows that the total number of cells on the grid is $6N_{\text{lat}}^2$. As an illustration, the grid is shown on Fig. 1.14.

To derived the number of cells in Eq. (1.21) from the area preservation formula, two approximations were made. The first is to consider the latitudinal spacing small enough, a valid assumption for resolution of 1° of latitude. In this case, we only make an error of only 0.001% when approximating $\sin(\Delta\phi_i/2)$ by $\Delta\Phi_i/2$ for all i. The other approximation is to consider that latitudes are close to the North pole, *i.e*, for all i $(i - 1/2)\Delta\phi_i \ll 1$. In practice, the relative error is less than 1% up to to 75°but increases for lower latitudes, with a maximum of 56% at the Equator. On top of that, the resolution given by this approximation is higher than it should be at the Equator. This is illustrated on Fig. 1.15, where the linear approximation is compared to the 'exact' number of

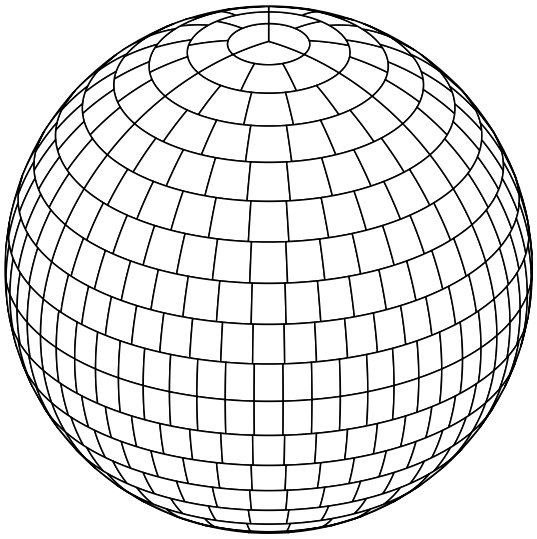


Figure 1.14: Pangolin grid with 20 latitudes.

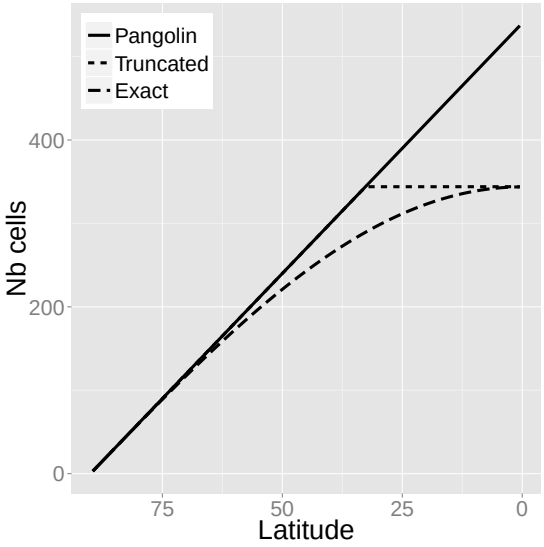


Figure 1.15: Comparison of the number of cells on Pangolin for our approximation (solid line), the truncated approximation and the 'exact' version. The grid has 90 latitudes on one hemisphere.

cells (*i.e*, without approximation). One solution to decrease the number of cells at the Equator is to truncate the number of cells at some point (dashed line on the figure). To truly preserve the cell areas, we should also use a variable cell height. However, the error in the cell area preservation was found to be acceptable so both of these two possible improvements were not implemented in Pangolin.

Neighbours of a cell

From Fig. 1.14, it can be seen that, while the Pangolin grid is not unstructured, computing the meridional neighbours of a cell are not immediate. On the other hand, zonal neighbours are easily computed. With periodical boundaries, the zonal neighbours of cell (i, j) are simply:

$$\begin{cases} (i, j-1) & \text{if } j > 1, \\ (i, n_i) & \text{otherwise,} \end{cases} \quad \text{and} \quad \begin{cases} (i, j+1) & \text{if } j < n_i, \\ (i, 1) & \text{otherwise.} \end{cases} \tag{1.22}$$

To find the meridional neighbour, we have to exploit the symmetries of the grid. From Eq. (1.21), we can deduce the grid has 4 axes of symmetry: the Equator and the meridians $\lambda = 0$, 120 and 240°. The grid can then be split into six identical zones, as shown on Fig. 1.16. In the rest of this paragraph, we only consider the first zone. For zones 2 and 3, the neighbours are found by adding the proper offset. On the South hemisphere (zone 4 to 6), north and south neighbours are simply

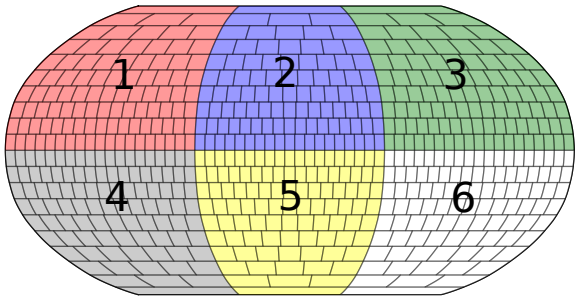


Figure 1.16: The six identical zones in the Pangolin grid with 20 latitudes with a Robinson projection.

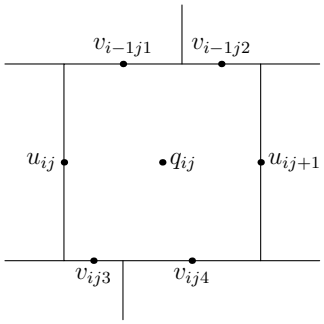


Figure 1.17: Discretization of zonal and meridional winds (u and v respectively) and tracer data q for cell (i, j) in Pangolin.

inverted. So the meridional neighbours on zone 1 for the cell (i, j) are $\{(i-1, j_1), \ldots, (i-1, j_2)\}$ (northern) and $\{(i+1, j_3), \ldots, (i+1, j_4)\}$ (southern), with:

$$\begin{aligned} j_1 &= \left\lfloor \frac{n_{i-1}}{n_i}(j-1)+1 \right\rfloor, \; j_2 = \left\lceil \frac{n_{i-1}}{n_i} j \right\rceil \\ \text{and} \quad j_3 &= \left\lfloor \frac{n_{i+1}}{n_i}(j-1)+1 \right\rfloor, \; j_4 = \left\lceil \frac{n_{i+1}}{n_i} j \right\rceil. \end{aligned} \tag{1.23}$$

From Eq. (1.23), it can be shown that most cells have two northern neighbours and two southern neighbours. Special cases include:

- the center of each zone, where a cell has one (resp. three) northern neighbours and three (resp. one) southern neighbour (s) on the Northern (resp. Southern hemisphere),
- the westmost cells on each zone, with one (resp. two) northern neighbours and two (resp. one) southern neighbours on the Northern (resp. Southern hemisphere),
- the eastmost cells on each zone, with the same number of neighbours as the westmost cells.

To decrease memory costs, the neighbours of the cells are not stored, contrary to unstructured grids, but are computed explicitly when needed. The computation is efficient as it involves only integer computation with some rounding. Due to the current trend of decreasing memory per core (see Section 3.1), we are emphasizing CPU computation to the expensive of memory. From a more general point-of-view, the algebraic features of the grid have been exploited as much as possible, especially in the parallelization process (see Section 3.2).

Pangolin and other reduced grids

While we did not present all reduced grids, we believe the chosen panel is representative of the different possible choices. On Table 1.3 is shown the gains in the number of cells when using the different reduced grids presented before as opposed to a regular latitude-longitude grid. Kurihara's

Table 1.3: Gains of the different reduced grids versus the regular latitude-longitude grid. The gains noted (*) were computed using the same resolution at the Equator for comparison (1° in longitude).

Grid	Reference	Gain
Reduced Gaussian	Hortal and Simmons (1991)	34.5%
Reduced	Kurihara (1965)	50% (*)
Reduced/composite	Rasch (1994)	34.5% (*)
Pangolin	this manuscript	33% (*)

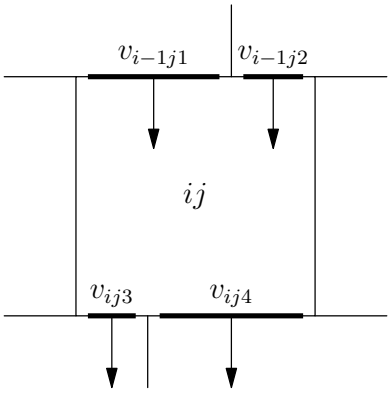


Figure 1.18: Meridional interfaces (bold lines) and fluxes (dark arrows) for cell (i, j).

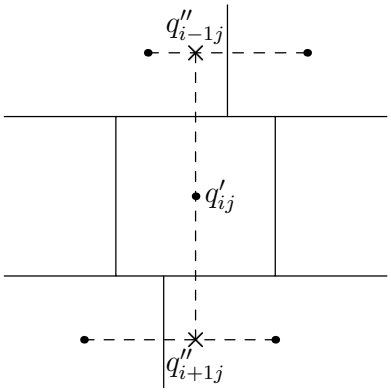


Figure 1.19: Linear interpolation to compute the meridional gradient.

reduced grid is quite similar to the Pangolin grid but has approximately twice the number of cells for each latitude. The grid by Rasch and Hortal has similar gains compared to the Pangolin grid. Rasch's grid allows for a more efficient computation of the neighbours: it only requires a distance computed once of every latitude and the neighbours are fewer and somewhat easier to compute. Hortal's is less relevant in our case as it is specifically designed for spectral models. However, if we were to adapt it to a finite-volume approach, it would require to store the neighbours as for an unstructured grid, thus increasing memory footprint. The most important requirement remains its parallel efficiency and the domain decomposition algorithm is an important aspect. The advantage of Pangolin is its custom algorithm, which exploits the analytical features as much as possible (see Section 3.2).

Adapting the scheme to the grid

To discretize the winds and tracer data, we use the same approach as the D-grid of Arakawa and Lamb (1977), illustrated on Fig 1.17. In practice, data is either generated at the proper position in the case of analytical tests (Chapter 2) or interpolated onto the grid when using 'real' data (Chapter 4).

While there is not particular difficulty to compute the zonal fluxes, the varying number of meridional neighbours means we have to compute the contribution of each neighbours. In practice, this means we have compute the interface between a cell and its meridional neighbours. As the

neighbours are given by Eq. (1.23) and knowing that cell width is constant at a given latitude, the size of the interfaces can be easily computed. Computing the meridional gradient is also not immediate. We use a linear interpolation to set a north and south ratio and compute the meridional gradient with finite-differences, as in Eq. (1.20). The interpolation is shown on Fig. 1.19. The sequential algorithm for 2D advection is summarized in Algorithm. 1 and 2. The parallel version is presented in Section 3.2.

Algorithm 1 Sequential zonal advection

1: Compute zonal gradient
2: Compute zonal fluxes
3: Update ratios

Algorithm 2 Sequential meridional advection

1: Compute zonal gradient
2: Compute meridional gradient
3: Compute meridional fluxes
4: Update ratios

To ensure mass preservation, winds are corrected such as the winds divergence is null. For 2D wins, we correct first meridional winds. If we consider a latitude circle, it constitutes a closed contour and the divergence on it must be null. So we compute the sum of all meridional fluxes on this circle and subtract the mean values from all meridional wind such as to achieve a null divergence. Then zonal winds are corrected such as the sum of all fluxes in each cell is null. To that end, we browse sequentially all cells at a given latitude and correct each east wind such as the sum of fluxes is null. The zonal wind at $\lambda = 0$ is used as an initial condition, *i.e*, is not modified by the algorithm.

During a time-step, the tracer mass is updated locally according to the fluxes in each cell. However, we should avoid unphysical fluxes which empty the cells of all (or more) of its tracer. This is usually done using a CFL condition. As we use a dimensional splitting algorithms, we have two 1D CFL conditions: one for zonal advection, one for meridional advection. As a global CFL, we use the most restrictive condition in both directions and for all cells:

$$C = \max_{ij} \left(\frac{u_{ij}\Delta t}{\Delta\phi_{ij}}, \frac{\Delta t \sum_{k\in V_{ij}} v_k \Delta\lambda_k}{\Delta\phi_{ij} \sum_{k\in V_{ij}} \Delta\lambda_k} \right),$$

where V_{ij} is the set of meridional neighbours for the cell (i, j) and $\Delta\lambda_k$ is the interface size between the cell and its neighbours.

Chapter 2

Testing suite for advection

In the development of a model, tests are paramount to ensure the accuracy and to check the behaviour of the scheme in real-life configurations. For transport schemes, idealized test cases are used in practice. Each test aims at examining the ability of the scheme to meet one or more of the properties presented in Section 1.2. For that, results of the schemes are compared to the algebraic solution of the test case using error norms. The tests should be representative of atmospheric situations. In particular, they should examine the behaviour of a single tracer, as well as the preservation of the relations between multiple tracers, an important feature for long-lived atmospheric species. Another important aspect of testing a model is to be able to compare it to other models. This can help judging if its performances are worth the implementation cost when compared to other state-of-the-art schemes. Unfortunately, there is no standard testing suite for transport schemes, which makes comparisons difficult. Williamson et al. (1992) proposed a testing suite in the case of shallow-water equations. It is a non-divergent and non-deformational test where the algebraic solution is known at all times. Nair and Machenhauer (2002) and Nair and Jablonowski (2008) presented deformational test cases with an algebraic solution known only at $t = 0$ and after a full period. Recently, Lauritzen et al. (2012) proposed a test-suite for transport schemes and a comparison with state-of-the-art models was presented in a following paper (Lauritzen et al. (2013)).

In this chapter, we present the results of advection in Pangolin using idealized bi-dimensional test cases. Pangolin is tested further in Chapter 4, where a chemistry scheme is added and real data is used. Here, we focus on global test cases dealing with difficult situations encountered in atmospheric transport, such as transport across the poles, deformational and non-divergent flows. In Section 2.1, we present two tests where the algebraic solution is known at any given time. In Section 2.2, Pangolin is compared to other existing transport schemes with the testing suite of Lauritzen et al. (2012) and the subsequent results of presented by Lauritzen et al. (2013).

2.1 Model validation

As we opted for an operator-splitting approach, the first test in validating the models was to check zonal and meridional advection. This was done using 1D test cases, not presented here. These tests allowed us to ensure the cell neighbours were properly computed, especially for meridional advection. It also allowed us to check the advection across the different symmetry axes, especially the Equator. In this section, we examine the behaviour of Pangolin in two 2D idealized test cases,

where the algebraic solution is known at any given time. For all tests presented in this manuscript, we have ensured that the mass is preserved by comparing the initial air mass to the air mass at the end of the simulation. As an example, for a full grid, the total air mass is around 2×10^4 and the variation is of the order of 10^{-9}, so around $10^{-14}\%$.

Solid body rotation

This test is commonly used when comparing advection schemes. The winds in this test case simply advects the initial distribution in such a way that it corresponds to a rotation around an axis. Winds are given by Williamson et al. (1992):

$$u = U_0(\cos\theta\cos\alpha + \sin\theta\cos\lambda\sin\alpha),$$
$$v = -U_0\sin\lambda\sin\alpha,$$

where θ and λ are the latitude and longitude respectively. The axis of rotation is defined by α, the angle of rotation between the z-axis and the polar axis. U_0 is set here such as a full rotation is 5 days. The initial distribution is defined as a cosine bell:

$$q(\lambda,\theta) = \begin{cases} \frac{h_{\max}}{2}\left(1+\cos\left(\frac{\pi r}{R'}\right)\right) & \text{if } r < R', \\ 0 \quad \text{otherwise,} \end{cases}$$

with $R' = R/3$, $h_{\max} = 1$ and R the radius of the Earth. r is the great-circle distance from the center of the cosine bell noted (λ_0, θ_0) to (λ, θ):

$$r(\lambda,\theta) = R\arccos(\sin\theta_0\sin\theta + \cos\theta_0\cos(\lambda-\lambda_0)).$$

The center of the cosine bell is set at $(\lambda_0, \theta_0) = (3\pi/2, 0)$. We follow the recommendations of Williamson et al. (1992) and use different rotation axes: $\alpha = 0, \alpha = 0.05, \alpha = \frac{\pi}{2} - 0.05$ and $\alpha = \frac{\pi}{2}$. The maximal wind speed is set such as a full rotation at the Equator takes 12 days. The time-step for Pangolin is set at 16min, leading to 1080 iterations for a 12-day simulation. The resolution at the Equator is set to $0.75 \times 1.13°$ (longitude-latitude) so the global CFL is 0.83 for the axes $\alpha = 0$ and $\alpha = 0.05$ (0.80 for the axes $\alpha = \frac{\pi}{2} - 0.05$ and $\alpha = \frac{\pi}{2}$).

Results are shown on Fig. 2.1, with the initial distribution and the results after a full rotation. The global error norms are presented on Table 2.1. The first two configurations execute a solid rotation with an axis close to the North Pole. As the Equator is an axis of symmetry, the final distribution of the tracer is, as expected, symmetric on each side of the Equator. It also allows to check that no mass is lost at the interface between the two hemispheres.
The other two configurations advect the tracer distribution over each pole once. This is an important test as Pangolin has only 3 cells at each pole, a potential source of loss of accuracy at the poles when compared to regular latitude-longitude grids. The shape of the distribution after a full rotation is distorted in the zonal direction and the extrema are much reduced. Hourdin (2005) used also the transpolar test case on a regular latitude-longitude grid with Prather's scheme and van Leer's scheme I. The van Leer scheme showed a deformation in the shape of the distribution in the meridional direction. This leads us to conclude that the accuracy of the scheme is the main reason for this deformation and for numerical diffusion. Furthermore, there is a sensibility to the 'tilting' of rotation axis ($\alpha = \frac{\pi}{2} - 0.05$, $\alpha = 0.05$), which can be seen in the slight deformation in

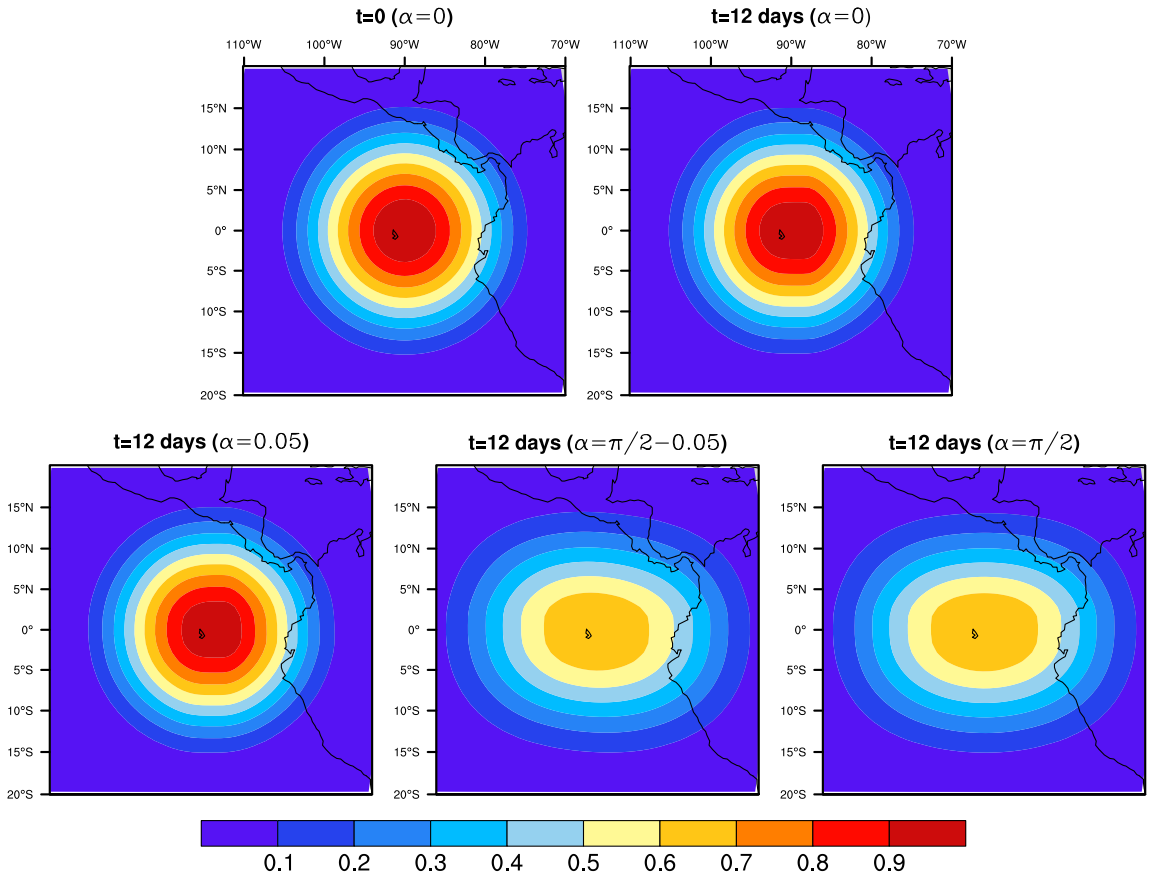


Figure 2.1: Contour plot for the solid body rotation test case at t=0 (upper-left figure) and after a full rotation (other figures). The projection used is a Lambert conformal projection centered at $(3\frac{\pi}{2}, 0)$. The resolution at the Equator is $0.75 \times 1.13°$ (longitude and latitude respectively). The rotation axis is set at $\alpha = 0, \alpha = 0.05, \alpha = \frac{\pi}{2} - 0.05$ and $\alpha = \frac{\pi}{2}$ respectively (from top to bottom and left to right).

one direction of the shape of the distribution ($\alpha = \frac{\pi}{2} - 0.05$). This observation is confirmed by the error norms in Table 2.1.

Snail test

As the previous test did not deform tracer distribution, we examine now the results using the test presented by Hourdin and Armengaud (1999). An initial Gaussian distribution is deformed by a vortex of winds, thus creating small filaments. As such, this test can quantify the accuracy of a

Table 2.1: Error norms for solid rotation, with different rotation axis (defined by α). The error norms are defined in Section 1.2. The algebraic solution is simply the distribution at $t = 0$.

α	l_2	l_∞
0	0.01274	0.01674
0.05	0.01625	0.02745
$\pi/2 - 0.05$	0.28723	0.30884
$\pi/2$	0.28456	0.30635

scheme in preserving these fine structures. Winds are given by:

$$\begin{aligned} u =& 2U_0 \cos\theta \sin\theta \cos^2\left(\frac{\lambda}{2}\right), \\ v =& -U_0 \sin\theta \cos\left(\frac{\lambda}{2}\right) \sin\left(\frac{\lambda}{2}\right), \end{aligned}$$

and derive from the following potential:

$$\Psi = RU_0 \cos^2\left(\frac{\lambda}{2}\right) \cos^2\theta, \tag{2.1}$$

where U_0 is a normalizing speed, R the radius of the Earth. Winds create a rotation around the axis centered at $(0, 0)$ and the period of rotation depends on the position of the considered point:

$$T = \frac{2\pi}{U_0 \cos\theta \cos(\frac{\lambda}{2})}.$$

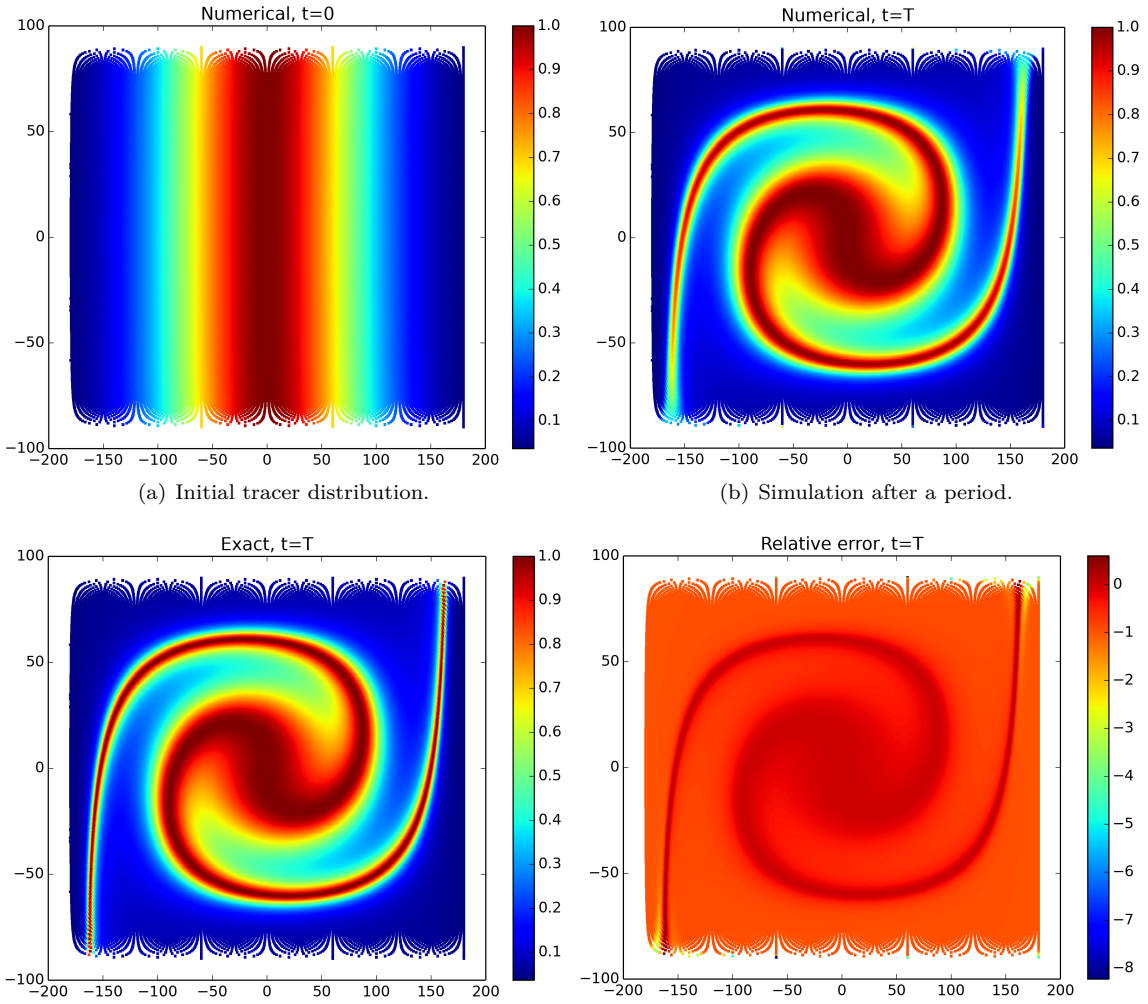
Here, U_0 is chosen such as $T = 12$ days at the center $(0, 0)$. For running Pangolin, the CFL is set at 0.90 such as a full period takes 384 iterations with a time-step of 45 minutes.

The exact solution can be computed algebraically at any time. For that, we use a Lagrangian approach where the trajectories are the iso-potentials. The initial position (λ_0, θ_0) of a given point (λ, θ) is found in a two-step process. First, θ_0 is found by integrating Eq. (2.1):

$$-\alpha \int_0^t \text{sign}\lambda dt = \left[\arcsin \frac{\sin\theta}{\sqrt{1 - \frac{\Psi}{RU_0}}}\right]_{\theta_0}^{\theta},$$

with $\alpha = \sqrt{\frac{\Psi U_0}{R^3}}$. Then, λ_0 is found using the relation $\Psi(\lambda_0, \theta_0) = \Psi(\lambda, \theta)$.

Results are shown on Fig. 2.2(b), while the exact solution given by the aforementioned method is plotted on Fig. 2.2(c), along with the relative error (Fig. 2.2(d)). It can be seen that Pangolin reproduces faithfully the filaments. Extremas are smoothed due to numerical diffusion, which is mostly visible near the poles. Most of the error is localized at the poles, due to numerical diffusion and fewer cells in these regions. We also have to ensure there is no local perturbation in the numerical solution to validate the implementation on the Pangolin grid. For that, we plot the evolution of two error norms, l_2 and l_∞, on Fig. 2.3 every 6 hours. As expected, l_2 increases smoothly. Even though a periodic pattern seems to appear in the plot of l_∞, this was not confirmed



(a) Initial tracer distribution.

(b) Simulation after a period.

(c) Exact solution after a period.

(d) Relative error after a period.

Figure 2.2: Tracer distribution at $t = 0$ and after a full rotation at the center with Hourdin's test case. The data is plotted as scattered points, where each point is the center of a cell on the Pangolin grid. No interpolation is done and the data is plotted on latitude-longitude coordinates. The grid resolution is $0.56 \times 0.37°$ (latitude and longitude respectively).

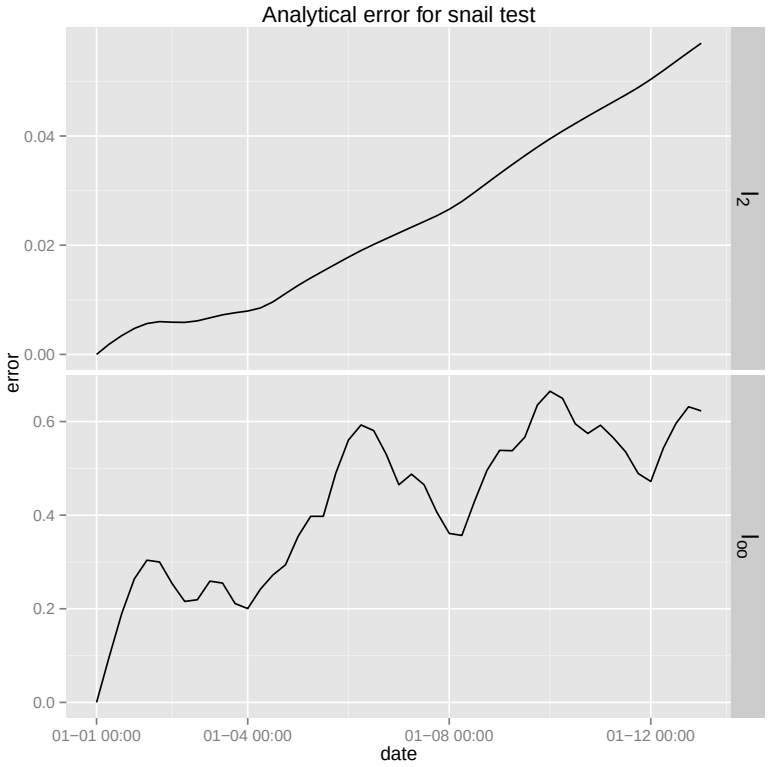


Figure 2.3: Evolution of the l_2 and l_∞ error norms (defined in Eq. (1.1)) for Hourdin's test case during a full rotation (12 days). The errors are computed every 6 hours. Resolution at the Equator is $1.13 \times 0.75°$ (latitude and longitude respectively).

by further experiments with different resolutions. However, the global trend of l_∞ confirms there is no numerical artefact in the scheme.

2.2 Comparison with other models

As mentioned in the introduction, comparison to other models is paramount for validating a new transport scheme. A paper was published in Geoscientific Model Development (GMD) (Praga et al. (2015)) gives detailed comparisons between Pangolin and other state-of-the-art models journal. The comparison uses the testing suite of Lauritzen et al. (2012), with the results of Lauritzen et al. (2013). The full paper is available in Appendix A, so the reader can refer to it for the results. Not all tests from the original paper were used in our comparison as we felt it more relevant to compare Pangolin to schemes with a similar theoretical order.

This led us to a selection of five mass-preserving transport models. These five schemes were compared to Pangolin with three diagnostics: convergence rate, preservation of filaments and preservation of preexisting relations. It should be noted that all the models presented in the tests have a

larger halo than Pangolin: FARSIGHT, CLAW and SLFV-ML use a halo of 2, while CAM-FV and UCISOM have a halo of 3. This difference can explain the difference in accuracy between Pangolin, which uses a halo of 1, and other models. In addition to the tests presented in Appendix A, one more test of the initial testing suite is presented below.

Rough distribution

The goal of this test is to check how well a scheme suppresses over- and undershoots using shape-preserving filter. Extrema should be also preserved. The slotted-cylinder test case is used here to provide a discontinuous initial distribution defined as:

$$q(\lambda,\theta) = \begin{cases} c & \text{if } r_i \leq r \text{ and } |\lambda - \lambda_i| \geq \frac{r}{6R} \text{ for } i = 1,2, \\ c & \text{if } r_1 \leq r \text{ and } |\lambda - \lambda_1| < \frac{r}{6R} \text{ and } \theta - \theta_1 < \frac{-5r}{12R}, \\ c & \text{if } r_2 \leq r \text{ and } |\lambda - \lambda_2| < \frac{r}{6R} \text{ and } \theta - \theta_2 > \frac{-5r}{12R}, \\ b & \text{otherwise,} \end{cases}$$

where $b = 0.1$ is the background value and $c = 1$ the amplitude. Otherwise, the notations are the same as in Appendix A.2. The impact of a shape-preserving limiter for Pangolin is shown on Fig. 2.4 after a full period. As it can be expected, diffusion smoothes the maxima and spreads out the minima. However, it does not completely remove the undershoots for Pangolin. Other models are shown at $t = T/2$ on Fig. 2.5. It should be noted FARSIGHT is not present in this comparison as its data were not available in the original paper. All the other models are given for a fixed resolution at the Equator of 0.75°. Pangolin does not match this resolution but is set to match the number of cells of most models, except CLAW.

Without shape-preserving filter, all models show evidence of undershoots as 'ripples' (light blue in the figure) in the background. CLAW and SLFV-ML also show overshooting (red lines). For Pangolin, CLAW and UCISOM, using a limiter does not completely remove undershooting. However, Lauritzen et al. (2013) noted the limiter of UCISOM was relaxed to avoid too much diffusion, thus explaining the undershooting. For our selection, CLAW is the only model which does not remove overshooting with limiters.

2.3 Conclusion

The tests presented in this chapter show the performances of Pangolin in regard to the different properties discussed in Section 1.2 and compare them to other state-of-the-art advection schemes. Pangolin does compete with the other models for the preservation of linear relations between two tracers. However, due to the numerical diffusion of Pangolin, the impact of the slope limiter on accuracy is found to be small for most of the tests compared to the other models. Nevertheless, the slope limiter efficiently preserves the shape of the distribution, as demonstrated in the slotted cylinder test. Like the other models, Pangolin ensures mass is preserved globally, an important property for chemistry transport models.

Pangolin is theoretically second-order accurate and was compared to schemes with a similar order on diverse grids (see Table A.2 for a summary). It was found the numerical convergence speed of Pangolin is slower than the theoretical order, possibly due to the linear interpolation done when computing the meridional gradient. The other schemes presented here achieve a numerical convergence speed close to two in optimal configurations. However, they use a larger stencil than

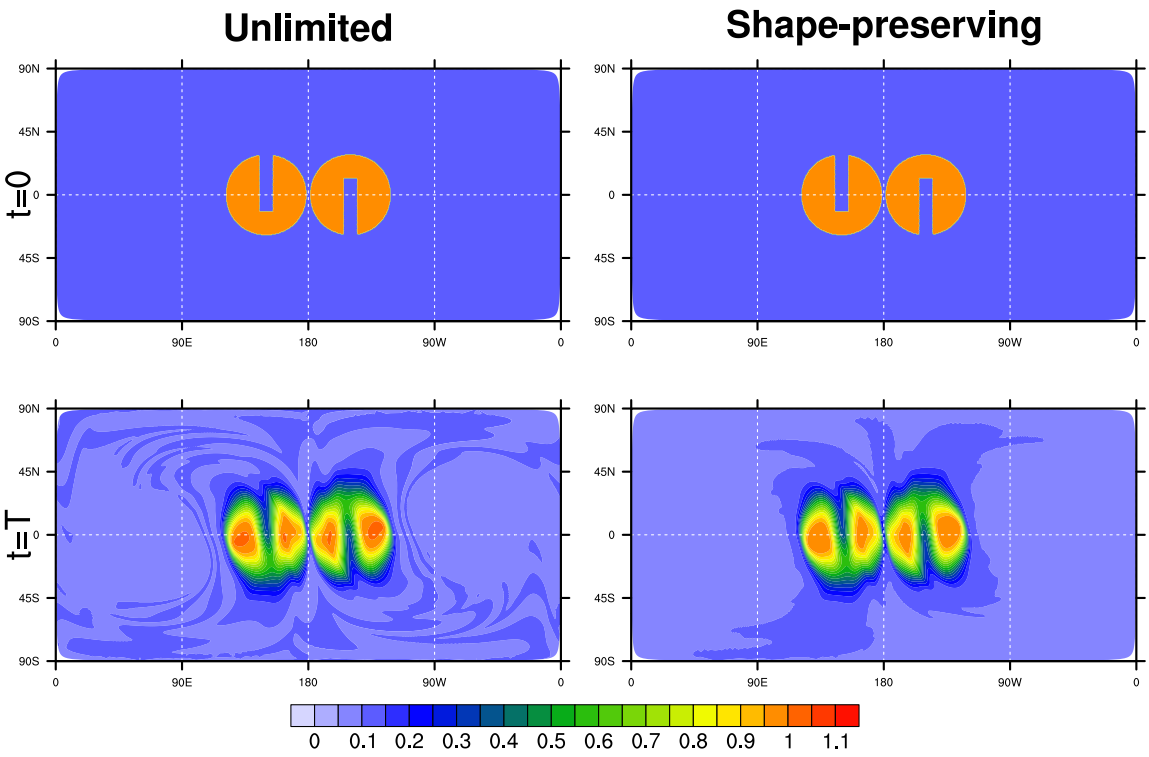


Figure 2.4: Initial and final concentration for the slotted cylinders test case. The model is Pangolin, with $0.56 \times 0.37°$ as resolution on the Equator with slope limitation.

Pangolin. Pangolin was designed with a smaller stencil to reduce communication costs in order to improve scalability. Another measure of accuracy, the preservation of filaments, confirms that Pangolin can match the results of the other models, working on non latitude-longitude grids, provided that a finer grid is used. In conclusion, our strategy is to compensate the loss of accuracy by using finer resolutions and by exploiting and maintaining good scalability to run efficient simulations on current supercomputers.

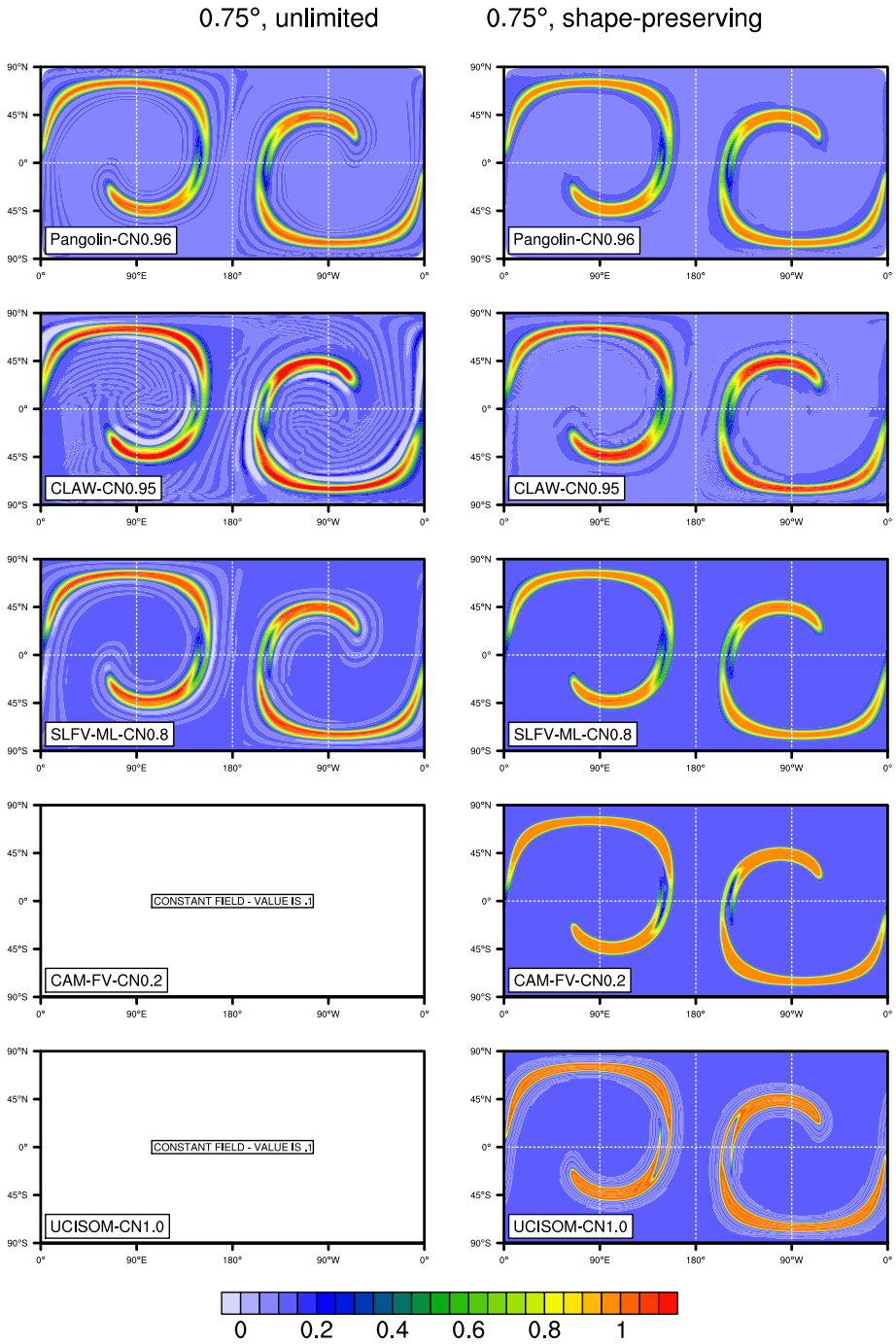


Figure 2.5: Tracer concentration for the slotted cylinders case, at $t = T/2$. The models are shown with both unlimited and shape-preserving versions of the models. All models, except Pangolin which has 0.37°, use a resolution of 0.75° at the Equator. Empty plots correspond to missing data.

Chapter 3

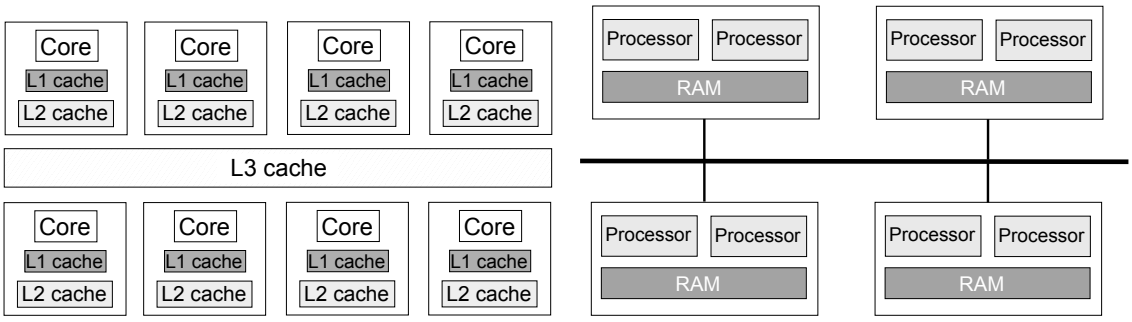
Parallelization

3.1 Introduction

Architectures

CTMs, and in a larger measure climate and weather models, are CPU-consuming applications, where the performances are usually quantified using the number of floating-point operations per seconds (flops). As such, they benefited greatly from the improvements in microprocessor architectures, usually described by Moore's law. Two consequences of this law for single-core microprocessor are the decrease of transistor area and transistor delay. Up to 2004, the increase in clock frequency was kept in check by the decrease in the operating voltage so cooling the chip was still rather inexpensive. It is no longer the case after 2004. Power is now a scarce resource so new processors aim for a better performance per unit power. For performance, the current limiting factor for microprocessor is memory bandwidth and latency and new designs are being studied to attempt to deal with that. Therefore, designing scalable applications must take into account the cost of memory accesses.

The current micro-architectures can be divided into three categories. The first contains 'conventional' microprocessors which focus on improving the performances of single threads, such as AMD's K8 core. Another approach is taken by Sun for its Niagara 2 chip where multiple threads are chosen over single-thread performance.
In the second category, we have Graphics Processing Units (GPUs), which were initially designed for graphics rendering but are now used for general computations. Their design is fundamentally different from CPUs. CPUs aim at decreasing latency, for example with larger caches (memory latency), branch prediction (branch latency) and better arithmetic and logic operation (operation latency). GPUs aim at improving the throughput using smaller caches for example. More energy-efficient than CPUs, they however are better suited for large number of truly independent tasks. NVIDIA is the most popular GPU constructor at the moment and their latest (2013) Kepler GPU have 15 multiprocessors with 192 cores each (NVIDIA (2012)).
Multi-core microprocessors make the third category. They emerged from the limitations of single-core microprocessor and stack on the same chip several cores. As an example of application, in pure MPI programming, a parallel task is associated to a single core. These microprocessors can be further divided into categories according to the communication between cores. The most common design is a 'tree' of different level of caches, with access to the main memory as the top as illustrated



(a) Memory hierarchy in the 8 cores of a Sandybridge processor. L1 and L2 are local to the cores (32KB and 256 KB resp.), while L3 (20MB) is common to the 8 cores. L1 is the small and fastest cache, while L3 is the largest but slowest.

(b) 4 nodes of the clusters are shown here. Each node has 2 Sandy Bridge processors sharing 32GB of RAM. Nodes are interconnected with an Infiniband network with a speed of 5GB/s.

Figure 3.1: An example of a cluster: the Bull cluster at CERFACS. It consists of 158 nodes (4 nodes are shown on (b)). Each node has two Intel Sandy Bridge processors. One of them is shown on (a).

on Fig. 3.1(a). More details and other designs can be found for example in Kogge (2008).

However, a single instance of one of these architectures is not sufficient for massively parallel applications, which require tens or hundred of thousands tasks. To achieve that, supercomputers are used in practice, where several 'computing nodes' are connected together via a fast network to create a cluster. Nodes can be designed specifically for parallel computation, such as the vector processors in the Earth simulator but the majority of clusters uses 'commercial' nodes, *i.e*, with mass-produced hardware. In practice, a node contains one or more microprocessors (multi-core or not) or GPUs in some cases. More details about the current architectures can be found in the "Top 500", which lists the most powerful supercomputers of the world twice a year (see the Top 500 website).

As an example of a cluster, the configuration of the Bull cluster at CERFACS (used later in the chapter) is shown on Fig. 3.1. It can be seen that each node shares a large amount of RAM for its processors but there is no global address shared across the nodes. From the point of view of the nodes, we have a *distributed-memory* model. However, at the level of the processor, we can have a *shared-memory* model. While these are the two main architectures, it should be noted there is an effort to use a global address space, the Partitioned Global Adress Space (PGAS) model. One implementation of this model is the Unified Parallel C (UPC) language (see Carlson et al. (1999)). The choice of the programming model then depends on the memory architecture as explained below.

Programming models

A program is 'parallelized' by splitting the computation into independent tasks. Each task is executed at the same time, between synchronizations. In practice, parallelization can occur at several levels, depending on the *granularity* of the problem. Finding a properly granularity helps to balance perfectly the computational load across all threads (*load balance*), an important goal in parallelization. With fine-grained tasks, it is easier to achieve load balance but at the cost

of overhead due to synchronization for example. On the other hand, coarse-grained tasks can decrease the overhead by agglomerating small tasks into larger one. The granularity can go from 'instruction-level' parallelism to 'thread-level' parallelism. Considering the current architectures, we need to focus on 'thread-level' parallelism. Even so, finding the right granularity depends on the application. Some problems are 'embarrassingly parallel', meaning parallelization is not an issue. However, most application, in particular CTM models considered here, do not fall into this category and require a programming paradigm. The two most popular for cluster of processors are MPI and OpenMP.

MPI

Message Passing Interface (MPI) is an Application Programming Interface (API) for the message-passing paradigm. The idea is to use 'messages' for communication between processes in distributed systems. MPI focuses on entities called processes, with separate adress spaces. MPI can also be used with multiple threads, but the two should not be confused. The message-passing approach was already popular in the 1990s and MPI was a standardization of different approaches. This contributed to its popularity, along with its performance and scalability, even in shared-memory environment. It was not the only approach: Parallel Virtual Machine (PVM) was another solution but has been supplanted by MPI in the 1990s. MPI is a standard and as such, has several implementations (*i.e*, the actual software): Intel MPI and Bull MPI are vendor implementations, while OpenMPI and MPICH are open-source implementations.

Each parallel task is assigned to a MPI process, which cannot access to data from other processes. So blocks of memory (the 'messages') are exchanged between processes when needed. As explained below, these blocks do not necessarily need to be contiguous. Here, we give more details for the different types of communications used in Pangolin. Communications can take the form of point-to-point communications, where a process A sends a message to another process B. But global communications are also possible:

- during a *broadcast*, a process sends the same message to all processes,
- with a *reduction*, a global operation is applied across all processes and the result is gathered onto a single process. For example, we can compute the global tracer mass across all processes,
- a *barrier* synchronizes all processes at a given point in a parallel code.

An important aspect for a MPI process is to be able to avoid waiting for communications — *i.e*, the arrival or departure of a message — to end before resuming its work. This is done using *non-blocking* communications. It should be noted than *non-blocking* only means the calling process can continue, not that communications are *asynchronous*. MPI has several non-blocking send modes but we will concentrate on `MPI_Isend`. Here, the sending process A sends to the target process B a request for communication. While data is being transferred, A can continue its computation, as long as it does not modify the data being sent. Once A has received a confirmation from B, it can then reuse the data being sent. We only mentioned here the relevant parts of MPI for Pangolin, but for more details about MPI communications and MPI in general, the reader can refer to Gropp et al. (1999) for example.

One advantage of MPI is its portability: it can now be used on the majority, if not all, of the current clusters of processors and can also run on desktops. It was also designed for scalability in mind and works well up to hundred of thousands of cores. However, future supercomputers

aims for millions of cores (*exascale*) and MPI 1 and 2 have trouble scaling to these configurations, mainly due to memory consumption, performance and fault tolerance, as explained in Thakur et al. (2010). Another consideration when parallelizing a model is the cost of implementation. Indeed, MPI requires rather important changes in the code to be efficient and alternatives such as OpenMP may be attractive for a faster development and incremental parallelization.

OpenMP

Open Multi-Processing (OpenMP) is an API designed for shared-memory systems. Here, the model used is the *master-slave* model where a *master* thread generates a set of *slaves* threads. Each one of them is assigned a task (either statically or user-defined), then synchronized and terminated. Like MPI, it is very portable and works on most of shared-memory parallel architectures. Unlike MPI, the cost of porting a code is much lower as OpenMP works by adding special directives (the `pragma` keyword in C and `OMP` in Fortran) in the sections to be parallelized. The most common operation is the parallel loop, which allows for load balancing across threads and reduction operations for example. The downside to its ease-of-use is the lack of control of the programmer on data layout, a possible limit to scalability. Nevertheless, its performances were found to be better than MPI on shared-memory systems after optimizations. This has led some applications to use hybrid programming by combining MPI and OpenMP together. Examples can be found in Chorley and Walker (2010); Jin et al. (2011); Guo et al. (2014).

Possible future trends

Using flops as the unit measure, we are now in the *petascale* (10^{15}) area and aiming for exascale (10^{18} flops). In Kogge and Shalf (2004), it was argued the current trends explained in the introduction are going to continue: clock frequency will not increase, power will be the primary constraint, flops will stay a cheap resource while data movement will dominate. Furthermore, parallelism will continue exponentially while memory will not scale according to the computing power. Kogge and Shalf (2004) also argued that only hybrid architectures have a chance of achieving exascale performance.

How does that impact us? In the case of Pangolin, advection is used as the basis for parallelization with the MPI library. For that, a custom domain decomposition technique is proposed in Section 3.2. It is a fairly standard technique but communication overhead becomes a concern when the number of cores is too large for a given resolution. Also, the strategy chosen here implies the different processes are equidistant, while in reality data movement is more expensive outside the processor on clusters. At the moment, Pangolin does not have aim at such extreme configurations but focuses rather on current clusters. This is why MPI was chosen, for both scalability and fine control over parallelization. Nevertheless, the warnings of Kogge and Shalf (2004) were heeded and some measures were taken to ensure the adaptability to the future. First, we exploit as possible the algebraic features of the Pangolin grid to reduce the memory cost of Pangolin. This exploits the decreasing cost of flops versus data movement. Another mitigation is to use hybrid programming by combining OpenMP and MPI to use more cores than there are subdomains.

3.2 Parallelization strategy for the CTM

In a CTM, chemistry and advection are separate, independent tasks. Also, advection of each tracer can be done independently of the others. However, this granularity is not enough for a

parallel model as the number of tasks would be limited to the number of advected species (around a hundred). For large-scale parallelization, it is best to focus on a domain decomposition technique, where the computation domain is split into connex subdomains. Each parallel task is then assigned a subdomain. As chemistry is local to each cell, only the advection step requires communication between the subdomains so our parallelization strategy focuses only on advection.

The computational load of advection is proportional to the number of cells of a subdomain due to the Eulerian approach chosen here (Section 1.4). So the goal of the domain decomposition is to create equal-area subdomains to achieve load balancing. We will first focus on parallelizing the advection of one tracer. Multi-tracer advection will simply aggregate the advection of the different tracers. We expect the communication cost to be a linear function of the number of tracers. The underlying hypothesis is that the cost of chemistry is constant for all cells. It is not the case in practice, due to factors such as cloud or the day-night interface. Nevertheless, mitigation for load unbalancing do the chemistry is studied in Section 3.5. For the advantages mentioned before (scalability, fine control, portability), we have chosen to use MPI as the parallel library and paradigm. The code itself is written in Fortran.

Domain decomposition

It should be noted our strategy should not be confused with 'classical' domain decompositions techniques, where boundary conditions are searched algebraically. Here, we aim at splitting our custom quasi-area preserving grid into equal-sized subdomains for the advection scheme. As mesh splitting is a common issue, professional tools are available, such as Scotch (Pellegrini (2012)) or Metis (Karypis and Kumar (1995)). These tools use graph algorithms to create high-quality partitionings. However, such methods are generic and do not take into account the specific structure of the Pangolin grid and this leads to unstructured subdomains as shown on Fig. 3.2(b). As such, it discards the algebraic features of the grid, such as the computation of neighbors cells. We have chosen instead to develop our own partitioning to exploit these algebraic features. The main consequence is that information is recomputed as much as possible, instead of being stored in memory. For example, the neighbours of a cell (given by Eq. (1.23)) and the neighbours of a subdomain (see below) are computed when needed. This follows the trend mentioned before, where flops are cheap versus data movement. We explain here how the partitioning is performed.

We have seen the Pangolin grid has four axes of symmetry, creating six identical zones (Fig. 1.16). First, we focus on splitting one zone. As it contains exactly N_{lat}^2 cells, the optimal number of subdomains is of the form p^2, with p dividing N_{lat}. In this case, all subdomains on a zone have exactly the same area ($(N_{lat}/p)^2$ cells). There are several possibilities for the topology of the p^2 subdomains. The most natural configuration for the subdomains is to use the same topology as the grid itself. On zone 1, subdomains are set on bands, where the i-th band contains $2i-1$ subdomains. This topology can be extended to the whole grid if the total number of subdomains is $6p^2$, with p still dividing N_{lat}: the subdomains are placed on each zone according to the algorithm mentioned before. The topology on the Southern hemisphere is simply symmetric in respect to the Equator. This results in the global decomposition on Fig. 3.2(a). In this optimal configuration, computing the neighbours of a subdomain and the interface between two subdomains is easy. It also helps reducing the number of neighbours of a subdomain (see below for a more detailed comparison with Scotch). It should be noted that the FARSIGHT transport scheme by White and Dongarra (2011) (presented in the testing suite in Section 2.2) has a similar constraint on the number of subdomains. The constraint comes from the cubic grid: each face has a square number of cells so

the total number of the subdomains should be of the form $6p^2$. To ensure each face of the cube has p subdomains of equal area, p has to be a divisor of the number of cells along an edge.

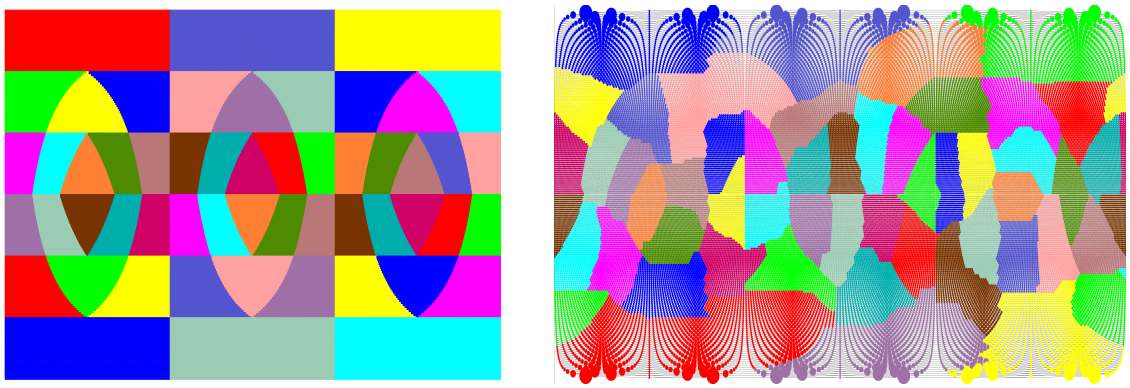
In order to relax the requirement on the number of subdomains (*i.e*, the number of cores), let us focus again on partitioning only one zone. If the number of subdomains p' is not a square, we find the closest best match $q = \lfloor \sqrt{p'} \rfloor$. Then we can apply the algorithm described before to place the q subdomains. The remaining $p' - q$ subdomains are then placed on the last band — *i.e*, the one closest to the Equator. It should be noted that q does not need to divide N_{lat}. In this case, the 'height' of the last band will be changed to include the remainder of the division. On the last band, subdomains have a much smaller size than the other q domains. They are set in the following way: the first $p' - q - 1$ subdomains are rectangular and of equal size, while the last one is trapezoidal with more cells. If the last band is too thin, other bands are slightly reduced to increase the height of the last band. The same strategy is applied to the other five zones to provide a global decomposition. Thus the only constraint is the total number of subdomains is a multiple of 6. An example of this global partitioning is presented on Fig. 3.3(a).
We can still go a bit further into relaxing the constraint on the number of domains. So far, the same strategy was applied for each zone. Instead, the final version of our partitioning only requires the total number of subdomains to be of the form $3p$. This is done by only requiring that zones 1, 2, 3 and 4, 5, 6 have each the same number of subdomains, but the number of subdomains could be different on each hemisphere.

Where does that leave us? Our partitioning works best in some optimal configurations, *i.e*, when the total number of subdomains is $6p^2$ with p dividing N_{lat}. For more flexibility on the number of cores, this condition can be relaxed to a number of subdomains of the form $3p$. However, the limiting factor for speed-up (defined more precisely in Section 3.4) is the size of the largest subdomains, which only changes for optimal configurations. In practice, it means that adding more cores between optimal configurations will not improve the performances. Nevertheless, this provides a flexibility which can be used later for improving load-balancing of the chemistry for example, or for allowing a different number of threads in a hybrid MPI-OpenMP configuration.

Comparing the partitioning by Scotch and Pangolin

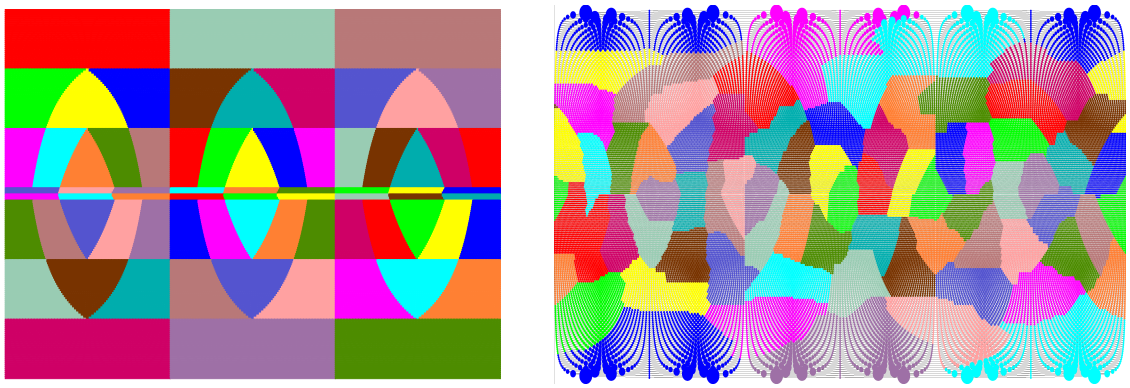
Here we examine in more details the difference between our algebraic partitioning and a general-purpose mesh partitioner, Scotch. For a visual comparison, Fig 3.2 shows the partitioning by Scotch and Pangolin in a optimal case for Pangolin, while Fig 3.3 shows the partitioning in a sub-optimal case for Pangolin. It is immediately apparent there is no underlying structure in the partitioning computed by Scotch. From a performance point of view, it means that both the topology of the subdomains and the interface size between subdomains need to be represented using a custom data structure. As it increases the memory storage, it conflicts with the possible future trends mentioned before and should be avoided.

The quality of the partitioning is quantified more precisely in Table 3.1. Subdomain size is roughly the same for both Scotch and Pangolin in the optimal case for Pangolin. For a suboptimal number of domains, load-balancing is less interesting as can be expected. However, the total number of neighbours is more beneficial for Pangolin: the subdomains created by Scotch have in average 2.5 more neighbours, even in the suboptimal case. This makes our partitioning more interesting as it requires a lower number of neighbours. It will decrease accordingly the number and volume of communications. From a computational point-of-view, the neighbours of a subdomain are computing using only integer arithmetic. While there are special cases when the number of



(a) Pangolin (b) Scotch

Figure 3.2: Comparison of our algebraic partitioning (a) to Scotch (b) for 54 subdomains (optimal case for Pangolin). Each color corresponds to a subdomain (the same color can be used for different subdomains). The total number of cells is 8100. Grids are shown in latitude-longitude coordinates.



(a) Pangolin (b) Scotch

Figure 3.3: Comparison of our algebraic partitioning (a) to the general purpose mesh partitioner Scotch (b) for 72 subdomains (optimal case for Pangolin). Each color corresponds to a subdomain (the same color can be used for different subdomains). The total number of cells is 8100. Grids are shown in latitude-longitude coordinates.

subdomains is sub-optimal, computation still remains extremely cheap. When searching for the interface between two subdomains, the idea is to find the intersection between the extremities of the subdomains and the neighbours cells of the adjacent subdomain. Again, this is mostly integer arithmetic, along with some floating-point computations. Therefore, we exploited as much as possible the algebraic features of the Pangolin grid to use the cheaper cost of flops versus data storage.

	Domain size			Neighbours		
	min	max	avg	min	max	sum
Scotch 54 cores	899	901	900	3	8	304
Pangolin 54 cores	900	900	900	3	6	123
Scotch 72 cores	661	689	675	3	9	416
Pangolin 72 cores	174	841	675	4	6	159

Table 3.1: Comparaison of the quality of the partitioning by Scotch and Pangolin (54 and 72 subdomains respectively). A dual recursive bipartitioning was used for Scotch.

Grid generation As mentioned in Section 1.6, the Pangolin grid is completely defined by the number of latitudes N_{lat} on an hemisphere and the number of cells at the poles. So the cost of storage of the global grid is virtually free, which is a huge advantage over unstructured grids. Another advantage of our algebraic partitioning is to preserve this lightweight structure even after the decomposition. Knowing the number of 'bands' on a zone, we can deduce the shape of the subdomain has a square number of cells or not and then deduce the shape of the subdomain (squared/triangular on normal bands or rectangular/trapezoidal on the last band). In the end, a subdomain is simply defined by two integers: the band number and the position in this band. This is extremely advantageous as the cost of storing subdomain is virtually free, especially when compared to unstructured domains, which must store the cell connectivity as well as the coordinates for all cells of each subdomain.

Advection algorithm

In flux-form finite volume advection schemes, the tracer ratio in a cell is updated according to fluxes at the borders (Section 1.4). In 1D, the air and tracer mass are updated according to Eq. 1.13 and 1.13, with the tracer fluxes defined by Eq. (1.19) and Eq. (1.20). 2D transport is achieved by applying 1D advection operators successively. From a parallelization point of view, the synchronization points are:

- after computing the 1D gradient,
- after computing the 1D fluxes,
- after updating the tracer ratios.

In a message-passing context, data outside the current subdomain is not available. As the advection scheme require some information outside the current subdomain, a layer of additional cells, called *ghost cells*, is created around the subdomain. Data from neighbouring subdomains will then be copied into these ghost cells. Their layout is illustrated on Fig. 3.4. The number of layers of ghost cells depends on the stencil of the scheme used. For the van Leer scheme chosen here, only one layer of ghost cells is needed.

To parallelize each step, the same strategy is used: first, we post a request for non-blocking communications. While communications are taking place, computations are performed at the *interior* of the grid, *i.e*, where no data outside the current subdomain is needed. Then we check

if the communications are finished and resume computation, this time at the *border* of the grid, *i.e*, where data outside the subdomain is needed. Non-blocking communications should be able to 'hide' the communications cost (message transfer and throughput) so that no time is lost waiting for data to be exchanged. Of course, this supposes the communications complete faster than the computation on the interior. Once the subdomains become small enough, this is no longer true so time will be wasted by waiting for communications to complete. The resulting algorithm for zonal advection is rather straightforward and is presented in Algorithm 3. An extra step must be added for the meridional transport: due to semi-structured of the grid, computing the meridional gradient requires an interpolation as shown on Fig. 1.19. The resulting algorithm is shown on Algorithm 4.

Algorithm 3 Parallel 1D zonal advection

Require: All tasks are synchronized
Ensure: Ratios are updated with new values from 1D zonal advection
Starts the communications for ratio in ghost cells
Compute the zonal gradients on the interior
Wait for the end of communications
Compute the zonal gradients on the boundary

Starts the communications for zonal gradient in ghost cells
Compute the zonal fluxes on the interior
Wait for the end of communications
Compute the zonal fluxes on the boundary

Update the boundary and interior ratios

Algorithm 4 Parallel 1D meridional advection

Require: All tasks are synchronized
Ensure: Ratios are updated with new values from 1D meridional advection
Starts communications for ratio in ghost cells
Compute zonal gradient on the interior
Wait for end of communications
Compute zonal gradient on the boundary

Starts communications for zonal gradient in ghost cells
Compute meridional gradient on the interior
Wait for end of communications
Compute meridional gradient on the boundary

Starts communications for meridional gradient in ghost cells
Compute meridional fluxes on the interior
Wait for end of communications
Compute meridional fluxes on the boundary

Update boundary and interior ratios

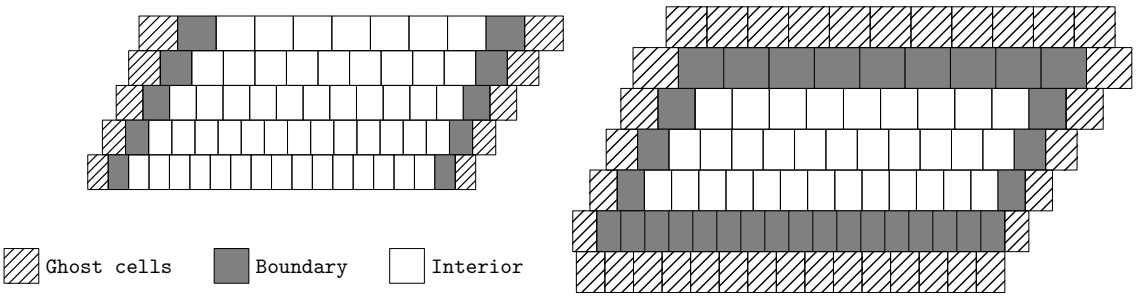


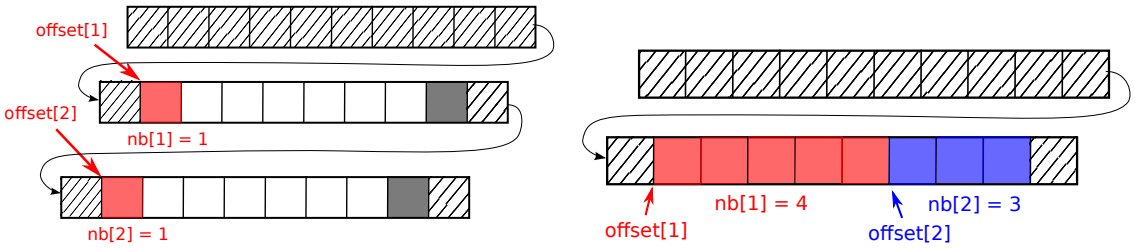
Figure 3.4: Ghost, boundary and interior cells for zonal (left) and meridional (right) advection for a rectangular subdomain.

	Zonal	Meridional
Ghost	$2/n$	$(4n+1)/n^2$
Border	$2/n$	$(4n-1)/n^2$
Interior	$(n-1)/n$	$(n-2)^2/n^2$

Table 3.2: Comparison of the size of ghost, border and interior cells for zonal and meridional advection. A subdomain of $n \times n$ cells is considered here and ghost cells are not counted.

Due to the structure of the Pangolin grid, ghost, border and interior cells have different shape and size for zonal and meridional advection, as shown on Fig 3.4. For zonal advection, a subdomain only needs to communicate with its east and west neighbours, while it requires communications with all of its neighbours for meridional advection. Inside a subdomain, the neighbours of a cells are also different in the zonal and meridional case: a cell has more than two meridional neighbours but has only two zonal neighbours. Also, meridional interpolation requires to have also the east and west cell neighbours, in addition to the north and south neighbours. This explains the 'stair-like' structure for meridional boundary cells. We can draw several conclusions from that. First, most of the computation occurs during the meridional step. Second, meridional advection needs twice the communication volume of zonal advection. Finally, meridional advection is also the most restrictive operation if communications are to be 'hidden' as the ratio number of cells over ghost cells is less favorable than in the zonal case. These differences are highlighted in Table 3.2, which shows the exact ratio between ghost, interior and border cells to the total number of cells (outside ghost cells).

Custom MPI communications As mentioned before, we exploit the features of MPI to hide communication costs with non-blocking communications. Fortunately, we do not have to send contiguous blocks of data by hand as MPI provides user-defined datatypes. As Pangolin has a semi-structured grid, data is stored as an 1D array. Therefore, defining datatypes adapted to ghost cells for example means the indexed structure must be used (`MPI_Type_indexed`). It defines irregularly-spaced block of data, where the blocks can be of different lengths. Two examples on how the datatypes are constructed are shown on Fig. 3.5. For each neighbour of a subdomain, two datatypes must be defined: one for ghost cells — receiving data from the neighbour — and one for interior cells — sending data to the neighbour. Of course, these are not the only datypes offered by MPI but it fits our data structure well. To create higher-level datatypes for more readable



(a) Interior cells for the east neighbour of the subdomain (in red).

(b) Interior cells for the two northern neighbour of the current subdomain (in red and blue).

Figure 3.5: Examples of two MPI derived datatypes for interior cells using the *indexed* structure. Datatypes are constructed as a set of blocks, where each block is defined by an `offset` and a size (`nb`) array. Data is a contiguous 1D array in memory. Ghost, border and interior cells are shown as hatched, filled and empty cells respectively.

communications, we used the *struct* datatype (`MPI_Type_create_struct`) in Pangolin.

Now that we know what data to communicate, let us examine the communication protocol. As a reference, the different send modes of MPI are shown in Appendix B.2. From the performance tests (see Section. 3.4), non-blocking communications works best with `MPI_Isend` on our cluster, matched with a `MPI_Irecv`. As data being send to (or received from) is directly read from (or written to) the actual data in memory, data in the border cells cannot be over-written before the transfer have been completed. After the computation is performed on the interior cells, we check if all requests have been completed with `MPI_Waitall`.

Multiple tracers Advecting multiple tracers is done by simply repeating the advection of a single tracer for each tracer. To decrease communication costs, all tracer data is sent at once, instead of sending a message for each tracer. From the point of view of MPI, once a datatype has been created for a tracer, the datatype for all tracers is simply the union of these single-tracer datatypes. It is expected performance will scale linearly according to the number of tracers (see Section 3.4) as the latency penalty only needs to be paid once for all tracers.

3.3 Input/Output

Due to the increase of computational power, CTMs have been using finer and finer resolutions, thus managing larger and larger datasets. On top of that, CTMs need to read and write data periodically and efficiently in operational contexts. I/O can no longer be done sequentially and need to exploit parallelism. Here we examine the different alternatives in the context of parallel I/O. The first strategy is that each process writes its own data on a separate file. It has the advantage of being rather easy to implement. Yet it makes post-processing difficult, as the global structure (shape and position) of the process is lost after the simulation, especially for Pangolin. Another disadvantage is that restarting the simulation with a different number of cores requires an extra step to interpolate data to the proper number of files. A second approach is to reserve a process for I/O only, which will receive and send data to or from other processes. A third solution is that all processes write to the same file, using either custom functions or high-levels libraries.

Unfortunately, there is neither an unique solution, nor a perfect implementation for all problems. A solution would be to skip I/O entirely for periodic output and run the simulation again when the data is needed. This approach only works when the computation is less expensive than data storage. In the case of CTMs, it is not an option as periodic read must occur. Either way, performance is attained by benchmarking the different alternatives and tuning one of them to the problem at hand for the available hardware. We now explain the evolution of I/O in Pangolin, from a sequential model to a fully parallel version.

Sequential

In our first version, each process wrote in turn in the same file. Using MPI, the processes passed around a *token*, which allowed the current holder to write in the file. This can be seen as a distributed implementation of a lock. The algorithm to read/write for a subdomain is highlighted in Algorithm 6. The idea is to read/write all cells until we are in the desired subdomain. The output format is ASCII, a human-readable format. Unfortunately, it does not provide compression, a highly desirable feature for large datasets. Using binary files would be a solution to reduce the size of the dataset but would require to implement our own I/O strategies. We have chosen to use a parallel I/O library to parallelize our sequential algorithm. Three of them are presented below.

Algorithm 5 Sequential ASCII read/write

Require: The current process has the token
Ensure: Domain data is read from/written to file and token is passed on
  Skip all latitude lines up to $first\_line$
  for $i = first\_line, last\_line$ **do**
    for $j = 1, \text{nb\_cells}(i)$ **do**
      if (i, j) is in the subdomain **then**
        Read/write cell (i, j)
      end if
    end for
  end for
  Rewind file
  Pass token to the next process

Parallel

MPI-IO One solution to read or write in parallel a shared file is the MPI-IO library (a recent description is given in Liao and Thakur (2014)). It was defined in the MPI 2 standard and as such is a natural choice for a MPI application. However, it is a rather mid-level approach and is used in practice by higher-level libraries. It can also create some portability issues due to low-level specifications such as endianness.

netCDF While there is no standard file format for in climate or weather community, Network Common Data Form (netCDF) is the format which comes closest to that. It is a high-level library with its own portable format, providing parallelism in two ways. Native parallelism is available with the Parallel netCDF (Li et al. (2003)) or, starting with netCDF 4, using the parallelism of

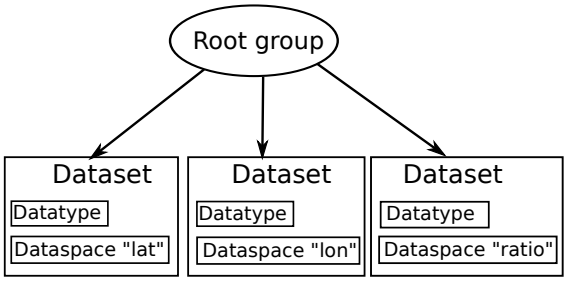


Figure 3.6: HDF5 data hierarchy for a simple file in Pangolin, storing the two coordinates of the cell center and the tracer ratio. Each data is stored in a container (*dataset*), which has data as an 1D array (*dataspace*) and the type of the array (*datatype*), here 64-bits floats (*H5T_IEEE_F64LE*).

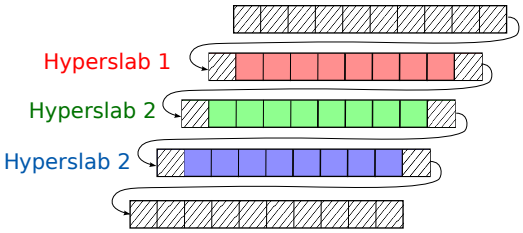


Figure 3.7: Constructing the interior (*i.e*, all non-ghost-cells) of subdomains as an union of hyperslabs. Each hyperslab is defined here by an offset and a number of blocks. Ghost cells are shown as hatched.

HDF5 shipped with it. Unfortunately, it is targeted at the regular latitude-longitude grid and does not support unstructured grids well. In Pangolin, we would like to define custom regions in our grid to read or write directly to/from it. It is possible in netCDF through the use of *slabs*, *i.e*, rectangular regions, but the lines inside a slab must be separated equally. This is not the case in Pangolin where data is stored as 1D arrays, so we must focus on a library with better unstructured grid support.

HDF5 Hierarchical Data Format (HDF) is another high-level library with its own format, adapted to large-volume and complex data, with efficient I/O. Its flexibility made it a natural choice for unstructured grids. For Pangolin, the latest version of the library, HDF5, was chosen. For more details about HDF5, the reader can refer to the introduction by Folk et al. (2011). To illustrate the flexibility of the HDF format, a simple example in the case of Pangolin is shown on Fig. 3.6.

HDF5 has the equivalent of the slabs from netCDF, except there are called *hyperslabs*. A hyperslab has the same limitation as the slabs in netCDF, *i.e*, the spacing between the blocks has to be regular. However, this limitation can be bypassed using an 'union' structure: we construct an hyperslab for each latitude line and the final hyperslab will be the union of all of them. This strategy is shown on Fig. 3.7, where an interior is constructed from the 1D array of cells. Once these hyperslabs are defined, the data can be read directly in parallel. For that, the input/output file must be opened with the adequate HDF5 flags (the *properties list*) and, with the help of some HDF5 wrappers, the proper section in the file is directly read into the proper memory location for each process.

3.4 Performances

Configuration

We now examine the parallel scalability of Pangolin for 2D advection. We focus on a strong-scaling test, where the total number of cells in the model is constant, while the number of subdomains increases. The speedup is a common metric to measure scalability for n cores. Here we take 3 cores as a reference and define the strong speedup as:

$$S(n) = \frac{T(3)}{T(n)}, \tag{3.1}$$

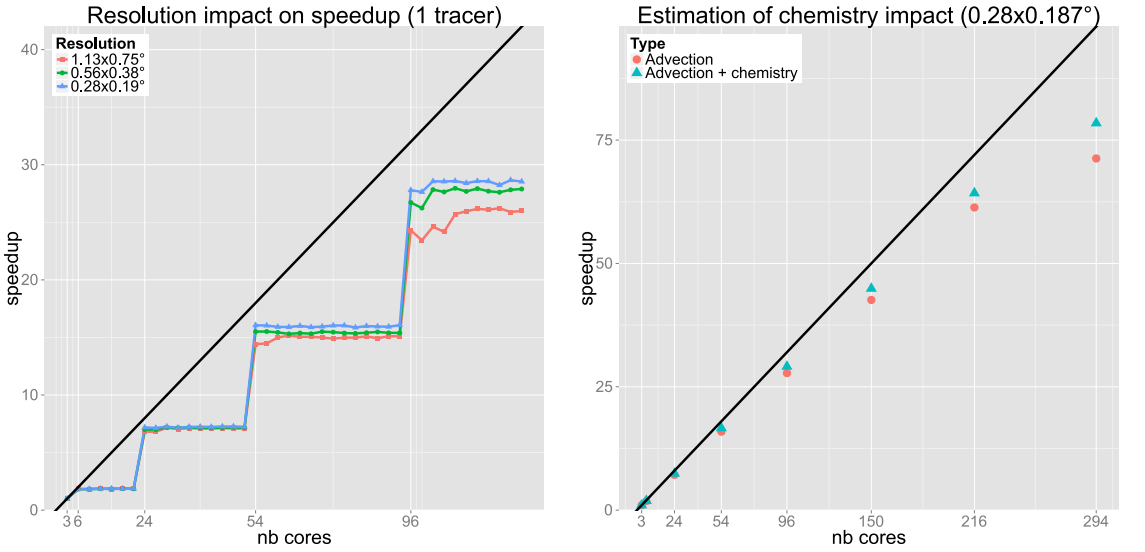
where $T(n)$ is the parallel elapsed time for n cores. In an ideal situation, $T(n)$ is reduced proportionally to the number of cores so that $S(n) = n/3$. In practice, we expect communication volume to increase such as the speedup departs from the ideal situation.

Here, only the elapsed time for advection is measured, *i.e*, the time spent in Algorithms 3 and 4. As a reference, a flowchart representing a full run of Pangolin is shown in the Appendix (Fig. 3.2). Measurements were done on "Neptune", the Bull cluster of the CERFACS. Its technical specifications are given on Fig. 3.1(a) and 3.1(b). Pangolin was compiled with the Intel compiler and run using Intel MPI.

Results

A first simulation was run up to 126 cores to study the impact of grid resolution on scalability. We used the Gaussian hills conditions described in Section A.2, where winds are constant, for a simulation of 12 days. Results are shown on Fig. 3.8(a). As expected, the increase in resolution improves the scalability as the computation workload grows. Furthermore, the speedup only increases at optimal configurations: namely 6, 24, 54 and 96 cores here. As expected, suboptimal cases uses more cores but the limiting factor is the size of the largest, *i.e*, slowest, subdomain, which does not change between optimal values fo the number of MPI processes. For a second test, we have ensured the speedup does not change when advecting multiple species (ten in our test). This confirms that communication costs are only paid once for all tracers and the cost of pure advection is truly a linear function of the number of tracers. Therefore, we can assume parallel performances will scale linearly with the number of tracers.

As a third test, the finest resolution from the first test ($0.28° \times 0.188°$) was used to examine the performance of 2D advection up to 294 cores. Results are presented on Fig. 3.8(b). At the time being, there is no full chemistry in Pangolin so the impact of chemistry was estimated using the average cost per cell. This cost was found using the ASIS chemical solver developed by D. Cariolle (personal communication, 2014). It solves locally a linear system associated with the integration of an Ordinary Differential Equation (ODE) required by the chemical interactions between the 90 species. An implementation was done by P. Moinat with the GMRES method, an iterative method to solve linear systems (Saad and Schultz (1986)).
As a first approximation, the cost of chemistry is constant for all cells so the estimation was found from the average cost multiplied by the number of cells. The figures only shows optimal situations, *i.e*, when the number of cores is of the form $6p^2$. For truly optimal configurations, we should require that p divides the number of latitudes in hemisphere. However, relaxing this constraint only unbalances slightly the load across the subdomains and allows for a larger choice of the number of



(a) Impact of grid resolution (up to 126 cores).

(b) Advection performances and an estimation of the chemistry impact on scalability.

Figure 3.8: Performance of 2D parallel advection using the speedup defined in Eq. (3.1). Resolutions used are: $1.125° \times 0.75°$, $0.56° \times 0.376°$, $0.28° \times 0.188°$. Both figures use non-divergent winds from Section A.2 over a full period with a CFL of 0.96.

cores. The plot is typical of parallel MPI applications, where the speedup departs from the linear speedup when the number of cores increases. As the chemistry is local to each cell, it does not require any communication volume and only benefits the parallelism, which is indeed confirmed by the plot. For instance, results at 294 cores give a speedup of around 72% of the theoretical speedup for the advection only and rises to 80% of the theoretical speedup with both advection the estimated chemistry.

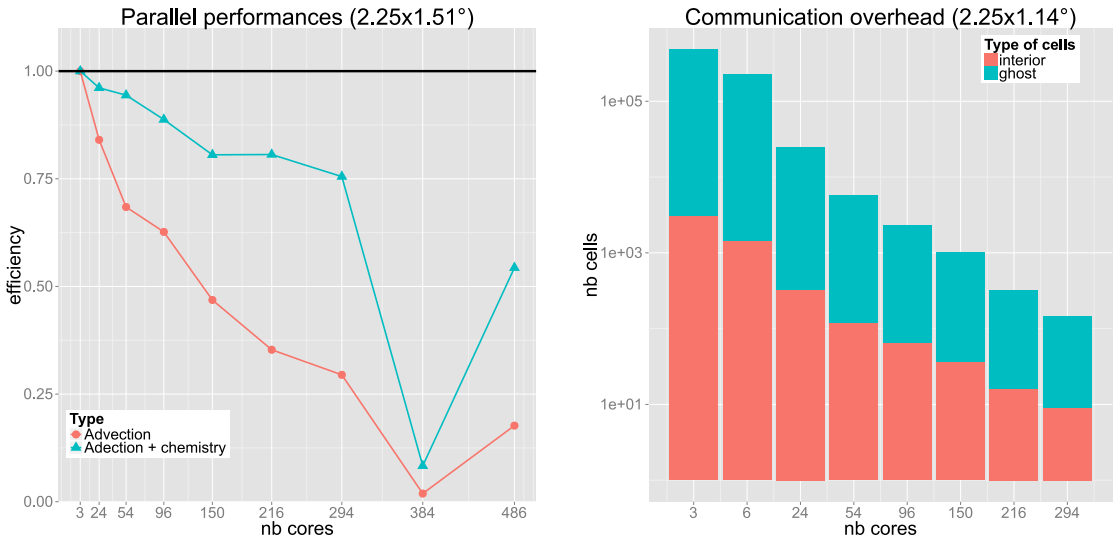
This leads us to examine the maximal number of cores that can be used for a given resolution. For that, we chose a rather coarse resolution ($2.25 \times 1.14°$) and increased the number of coarse until the speedup became too low. This time, we used the Airain supercomputer instead of the CERFACS cluster (see Table 3.3 for its main technical features). Results are shown on Fig. 3.9(a), which shows *efficiency* instead of speedup. Efficiency is simply defined as an adimensioned speedup, so in our case:

$$E(n) = \frac{T(3)}{nT(n)} \tag{3.2}$$

As such, the closest $E(n)$ is to 1, the better parallel performances are. To understand the results, the x-axis on Fig. 3.9(a) shows the number of interior cells for meridional fluxes over the total number of interior cells. For more details, Fig. 3.9(b) gives the ratio of ghost cells over interior cells. For advection only, we can consider efficiency is too low below 0.5, *i.e*, for 150 cores. This corresponds to subdomains of 8×8 cells, where roughly 44% of the cells are ghost cells. Such subdomains are obviously too small for practical purposes and only represent a waste of the memory of the

Table 3.3: Configuration of the Bull cluster Airain

– 549 nodes Bullx B510
– 9504 cores Intel SandyBridge
– each node has 16 cores at 2.7Ghz and 4GB shared by the core



(a) Efficiency for advection and with an estimation of the cost of chemistry.

(b) Proportion of ghost and interior cells for a subdomain and for meridional fluxes. The number of cores is not linear.

Figure 3.9: Impact of communication overhead on parallel performance (2D advection) using the Airain supercomputer. The simulation was run with a coarse resolution ($2.25 \times 1.14°$) until the performances broke down. Only perfect cases for our partitioning were considered.

node. This shows the performances of the parallel advection, which only 'breaks' down for such small subdomains. On top of that, we estimate the chemistry will more than double this maximal number of cores, as for 294 cores, the total efficiency is estimated at 0.75.

3.5 Future work

This chapter only dealt with 2D advection, on which our parallelizaton strategy is based on. In the next chapter, a linear scheme for ozone is presented, to test Pangolin in a "real-life" scenario. However, chemical schemes are more complex in practice and require to solve stiff linear systems. As such, the chemistry step is much more expensive than pure 2D advection. From a parallelism point of view, we expect the increase in the computational load to improve speedup. On the negative side, we expect load balance to be disturbed by heterogeneous chemistry, surface emissions, as well as convection, vertical diffusion and the day/night interface. Extension to the 3D case for advection should also be beneficial for parallel performances. The idea is to use the 2D domain decomposition

and extend it vertically so a subdomain is a set a vertical columns. It is not uncommon to have around 60 vertical levels so the computational workload for advection should be multiplied by 60.

Here we presented a static load-balancing strategy based on a domain decomposition strategy. To mitigate the future load unbalancing, a dynamic strategy can be a solution. A first approach would be to use a hybrid programming paradigm with MPI and OpenMP, where a varying number of threads would be added on top of the MPI processes to manage the load unbalance. Another solution would be to split each subdomain into several subdomains and to assign a thread to it. However, this would require a domain decomposition strategy, as complex as the one developed for MPI. A third possibility is to exploit suboptimal configurations in the number of cores. As parallel performance is similar to the closest lower optimal case, the number of cores for advection could be decreased and the now idle cores could be stealing work from these 'master' processes.

Chapter 4

Real-case simulation

4.1 Introduction

In this chapter, we examine the behaviour of Pangolin forced with realistic winds from meteorological analyses and with the introduction of a linearized chemistry scheme for ozone. Since our goal is to obtain in the long run a complete CTM based on Pangolin environment, this study is a natural extension of the previous simulations described in Chapter 2, in which the 2D advection was only tested with idealised tracer evolutions.

To validate the simulations, the results of Pangolin are compared to those of the MOCAGE CTM described in Section 4.2. MOCAGE includes a full 3D transport scheme so, since the advection scheme of Pangolin is only 2D, we focus on the advection on an isentropic surface located in the mid-stratosphere. Diabatic processes in the mid-stratosphere are dominated by radiation, which induces rather small net heating rates at those altitudes. It is therefore relevant to consider that on time scales of several days the air motions are mostly adiabatic. Here, we examine a simple chemistry scheme for ozone, where the ozone concentration does not depends on other species. The configuration used for the simulations is highlighted in Section 4.2. Results of Pangolin results are shown in Section 4.3, along with a comparison to the results of MOCAGE.

4.2 Model set up

We have chosen stratospheric ozone as it is a good candidate for studying the advection and its interference with chemistry. Simulations in the stratosphere are also relevant for climatic applications as ozone is a key component of the temperature balance at those altitudes. Stratospheric ozone distribution has also been extensively monitored for several decades using remote sensing instruments onboard satellites, providing useful datasets for model validation. In the stratosphere ozone is produced by photodissociation of molecular oxygen. This production is balanced by destruction by catalytic cycles involving NOx and HOx radicals. Ozone lifetime is in the range of several days at the equator in the mid-stratosphere, a time scale comparable to the advection around the polar vortices. We have therefore chosen to focus our analysis on an altitude close to 30 km for the month of september 2008, a period for which we had previously performed simulations with the MOCAGE model.

Meteorological data

A chemistry-transport model can be used in either an *online* or *offline* setup. In an online setup, the chemistry is directly integrated into GCMs. In this case, accessing the meteorological data (pressure, temperature) from the dynamics is not an issue. However, in Pangolin, we are in an offline setup: the CTM is external to fhe GCM and the two must be coupled. The most common strategy is to use desynchronised coupling. Noting Δt the timestep of the coupling, an iteration in the coupling would be:

1. chemical species are passed from the CTM to the GCM,
2. the GCM is integrated during Δt,
3. meteorological data are passed back to the CTM,
4. the CTM is integrated during Δt.

Δt is usually 3 to 6 hours. A shorter time-step would be expensive, while a longer time-step would create some bias between the data of the CTM and GCM, mostly due to the parametrization of unresolved processes (turbulence, convection).

In our study we use the data from the ECMWF *re-analyses* to force both MOCAGE and Pangolin. To obtain these re-analyses, observational data is assimilated a first time, then fed to the model. The output of the model is then assimilated again to match observational data at the next date. Here, Pangolin and MOCAGE are integrated using assimilated winds and temperature every three hours. The initial ozone distribution on the 1st of September comes from an analysis using the MOCAGE-VALENTINA suite (Emili et al. (2014)).

As the advection and chemistry time-steps are smaller than 3 hours, we added a temporal interpolation for the necessary data (temperature and winds) at the required steps. Input winds at t and $t + \Delta t$ are already corrected from the preprocessing step (Section 4.2) and the interpolation is linear so the interpolated winds remain non divergent.

Treatment of the vertical outputs of MOCAGE and Pangolin

In 3D models, several sets of vertical coordinates can be used. The first is to use simply pressure coordinates, which has the disadvantage of not taking the landscape (mountains for example) into account. A solution to that would be to use the latitude over the landscape but we would lose the pressure information. Another set of coordinates uses the *normalized pressure* defined by $\sigma = P/P_s$, where P_s is the pressure level. Unfortunately, these coordinates do not work well for higher altitudes. MOCAGE uses *hybrid* coordinates noted (σ, P), a common choice for atmospheric models as it solves the previous issues. For a latitude i and longitude j, the pressure levels at (i, j) are defined as:

$$P_i = A_i + B_i \times P_s(i, j),$$

where A_i and B_i are coefficients independent of the longitude and $P_s(i, j)$ is the surface pressure at (i, j). This set of coordinates is called hybrid as it follows the landscape at low altitudes (the A_i are close to 0), while the levels becomes similar to the isopressure levels (the B_i are close to 0). An illustration of theses coordinates is shown on Fig. 4.1.

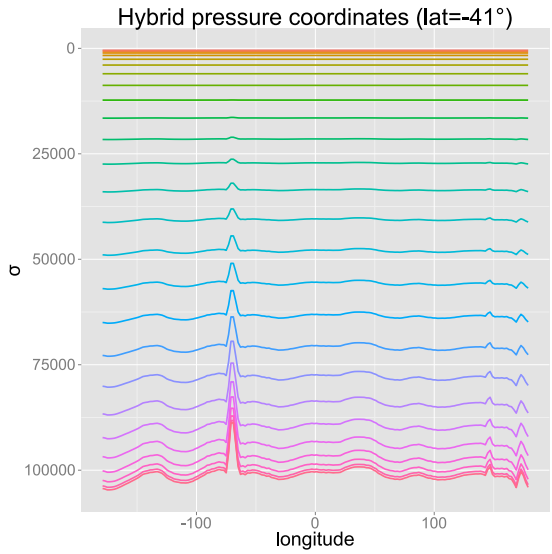


Figure 4.1: Hybrid coordinates of MOCAGE at latitude=-41°as a function of longitude. For readability purposes, not all levels are shown here (roughly one over two). The largest peak corresponds to the Andes mountains.

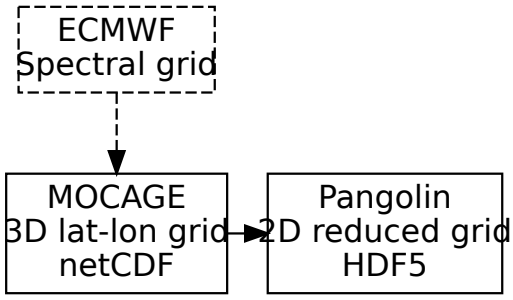


Figure 4.2: Workflow for the input data: MOCAGE uses data from the European Center, which is then interpolated onto the Pangolin grid.

In the stratosphere since the air motion can be considered adiabatic at a first approximation, it can be interesting to use isentropic coordinates. The *potential temperature* is defined as:

$$\theta = T\Big(\frac{P_0}{P}\Big)^{\gamma}, \tag{4.1}$$

where T is the temperature in Kelvin, $P_0 = 1000$ hPa the pressure of reference and $\gamma = 2/7$. If the atmospheric motions are considered as adiabatic, θ is a constant quantity and can be used as a vertical coordinate. We have thus chosen this coordinate to make the comparison between MOCAGE and Pangolin. Pangolin is integrated on a surface at $\theta_0 = 850$K with winds and temperature interpolated on that surface. The outputs from MOCAGE are directly interpolated on the same surface from its σ coordinate. The surface at $\theta_0 = 850$K corresponds to a pressure of roughly 10 hPa in the 30 km altitude range.

Spatial interpolation

The MOCAGE grid is a three-dimensional regular latitude-longitude grid. We want to interpolate the ozone concentration, temperature and winds onto the Pangolin grid, defined in the isentropic surface mentioned above.

First, we interpolate ozone concentration and temperature as there are both defined at the center of the Pangolin cells. For each latitude and longitude, we interpolate vertically the data from the

hybrid coordinates to $\theta = \theta_0$ using cubic splines. This gives us data on the isentropic surface but still on regular latitude-longitude grid. To interpolate it horizontally on the Pangolin grid, we use the SCRIP library (see Jones (1997)). Bi-dimensional interpolation on the sphere can be difficult due to the singular points in spherical coordinates. SCRIP manages these difficulties and offers an area-preserving mapping, described in more details in Jones (1999). This is especially interesting for us as the interpolation preserves the tracer fluxes needed by our finite-volume approach. SCRIP needs first two netCDF files defining the input and output grids. It then proceeds to create an intermediate netCDF file for the mapping between the grids. In the last step, it uses these mappings from the file to interpolate the data. We used the default parameters of SCRIP, with the exception of the northern threshold. Above this threshold, a coordinate transformation is used to perform the intersection search needed by the conservative remapping. Our tests showed this transformation introduced non physical structures so it was disabled.

Our first approach to interpolate winds was to keep using SCRIP. However, the library requires the data to be defined at the centers of the cells. While the input winds are defined at the center of the latitude-longitude grid, winds in Pangolin are defined at the interfaces of the cells. To accommodate the requirements of SCRIP, we defined a *dual grid* such as the winds in Pangolin is at the center of the cells in this dual grid. Zonal and meridional winds are set in a fashion similar to the D-grid of Arakawa (see Fig. 1.18) so a dual grid has to be defined for both zonal and meridional winds. Zonal winds adds a supplementary difficulty as there must be periodic, a feature not managed by SCRIP. To bypass this issue, zonal interpolation was done in two times. First, the latitude-longitude grid was shifted such as the westmost zonal values were interpolated correctly. The input grid was shifted back to its initial position to interpolate the other values. Finally, periodicity was enforced manually for the eastmost zonal values.

Unfortunately, this approach proved to generate interpolation artefacts in the simulation results when resolution was quite small (0.3° at the Equator). As winds do not need the area-preserving mapping from SCRIP and as the other interpolation schemes provided by SCRIP cannot accommodate a non-regular structure, we chose another, simpler interpolation technique. The idea is to use successive 1D interpolations with cubic splines. The algorithm is detailed in 6. We first construct an initial array of interpolated value at each latitudes, and then interpolate the final values from it.

Algorithm 6 Winds interpolation

Require: 2D winds array u on lat-lon grid
Ensure: 2D winds array U on Pangolin grid
  Skip all latitude lines up to $first\_line$
  for all ϕ_i in Pangolin **do**
    for all λ'_j in lat-lon **do**
      $u'(\lambda'_j)$ = cubic interpolation on u for all latitudes in lat-lon
    end for
    for all λ_j in Pangolin **do**
      $U(\phi_i, \lambda_j)$ = cubic interpolation on u' for all longitudes in lat-lon
    end for
  end for

Programming details

From a software point of view, we created a small external preprocessing tool which takes a 3D latitude-longitude grid as input in the netCDF format and interpolate it to a 2D isentropic Pangolin grid in HDF5. The workflow is illustrated on Fig. 4.2. For horizontal interpolation, the subroutines from SCRIP are called directly instead of running the executable for each timestep and for each data. Also, the mapping between the latitude-longitude and the Pangolin grid are only created once.

Description of MOCAGE

MOCAGE is the three-dimensional CTM developed and used by Météo-France, the French weather forecast agency. It is used in data assimilation experiments (Cathala et al. (2003)), where the output of models is combined statistically with observations, taking into account the incertitudes. It can also be used for chemical weather forecasting, where meteorological data is given to the CTM, which in turns forecasts the evolution of the chemical composition of the atmosphere. One application of this is air quality forecasts.

MOCAGE uses a semi-Lagrangian scheme based on Williamson and Rasch (1989) for advection. It includes the stratosphere and the troposphere and various chemistry schemes can be used, from simplified linearized schemes up to detailed schemes including gas-phase and heterogeneous stratospheric and tropospheric chemistry (Lefèvre and Brasseur (1994)). The input winds and temperature come either from ARPEGE, the operational global model of Météo-France, or from the ECMWF analyses or forecasts.

In our study MOCAGE and Pangolin are forced using the ECMWF operational analyses. MOCAGE uses is a bi-dimensional latitude-longitude grid with 60 vertical levels in hybrid coordinates. The time-step for advection is set to 1h, while the chemistry time-step is set to 15min. A complete description of the MOCAGE model is given by Josse et al. (2004).

Linear chemistry

The global evolution of the tracers is a CTM is the sum of many physical processes, where the most important in the stratosphere are advection and chemistry operators. This can be implemented by integrating in time both advection and chemistry directly. The different natures of the associated operators make that strategy especially difficult. Instead, a most common method is adopted where the temporal integration is carried out successively for each process over a series of time-step. The advection operator gives an intermediate tracer evaluation, which is then fed to the chemistry operator, resulting in the final tracer value at the end of the time-step. As this strategy introduces a splitting error, a scheme with alternate directions (in a way similar to the strategy presenting for the advection) can be used in practice.

Here, we focus on the chemistry scheme. Interactions between the different chemical species lead to the resolution of a set of coupled ODEs. Unfortunately, the chemical species in the atmosphere have very different lifespans, ranging from milliseconds to several weeks. The set of equations is then said to be *stiff* so the usual explicit methods do not work: the global time-step of the system will be set as the smallest time-step of the system, which is too constraining. Several methods can be used to bypass this issue. Quasi-Steady States Approximations is a method presented by Hesstvedt et al. (1978), which has the advantage of being positive and allowing larger time-step. Unfortunately, it does not guarantee mass preservation. Another approach by Verwer (1994) is called “two-steps

backwards". For a comparison between the different methods, the reader can refer to Sandu et al. (1996).

To simplify the problem and concentrate on comparison between MOCAGE and Pangolin, we have chosen instead to use a linear scheme for ozone. Such a scheme was developed by Cariolle and Teyssèdre (2007) and has proved to be well adapted for ozone simulation in the stratosphere. With this scheme, the ozone continuity equation writes:

$$\frac{\partial r_{O_3}}{dt} = P - L,$$

where r_{O_3} is the ozone mixing ratio, P and L are the production and loss rate respectively. The equation is expanded using a Taylor serie to:

$$\frac{\partial r_{O_3}}{dt} = A_1 + A_2(r_{O_3} - A_3) + A_4(T - A_5) + A_6(\sigma - A_7) + A_8 r_{O_3}, \tag{4.2}$$

where the A_i are monthly-averaged coefficients (the overlay denotes the average):

$A_1 = \overline{P - L}$ $\quad A_5 = \overline{T}$: temperature

$A_2 = \frac{\partial \overline{P-L}}{\partial r_{O_3}}$ $\quad A_6 = \frac{\partial \overline{P-L}}{\partial \sigma}$

$A_3 = r_{O_3}$ $\quad A_7 = \sigma$: ozone column

$A_4 = \frac{\partial \overline{P-L}}{\partial T}$ $\quad A_8$: heterogeneous chemistry term

More precisely, the coefficient are given by MOBIDIC, a bi-dimensional photochemical model developed by Cariolle and Brard (1985). The model was run for the year 2000 and its coefficients are used as if for the year of our simulation (2008). For our study, only the first five terms (A_1 to A_5) are taken into account in the Pangolin simulation. This means that the ozone concentration will be sensitive to temperature, but not to the ozone column above the chosen potential temperature surface. Equally, the potential impact of heterogeneous chemistry is not taken into account.

A semi-implicit scheme is then used to discretize Eq. (4.2) as it allows for a larger time-step and preserve the positivity of the ozone concentration. The equation becomes:

$$\frac{r_{O_3}^{n+1} - r_{O_3}^{n}}{\Delta t} = A_1 + A_2(r_{O_3} - A_3) + A_4(T - A_5),$$

leading to

$$r_{O_3}^{n+1} = \frac{r_{O_3}^{n} + \big(A_1 - A_2 A_3 + A_4(T^n - A_5)\big)\Delta t}{1 - A_2 \Delta t},$$

where T^n is the temperature at time-step n and Δt the corresponding time-step.

4.3 Results

We present here the results of a run of Pangolin over one month, both with and without chemistry. The simulation is run from September, 1st 2008 to October 1st 2008. MOCAGE uses a 3D regular latitude-longitude grid, with 60 vertical levels and a horizontal resolution of $2° \times 2°$. Several resolutions have been tested with Pangolin. The coarser grid has a resolution of $2.9° \times 1.96°$ at the equator, close to the one of MOCAGE. The same time-step of 30 min is used for both the chemistry and advection. The finer grids tested have resolutions of $1.5° \times 1.0°$ and $0.5° \times 0.33°$.

To ensure the horizontal interpolation of Section 4.2 are accurate, we compare the initial ozone concentrations interpolated on the Pangolin grid to its equivalent on the MOCAGE grid. To make the comparison relevant, the MOCAGE data is interpolated on the same isentropic coordinate as Pangolin. Results are shown on Fig. 4.3. It should be noted that the plot is done using the NCL software (see the NCL reference manual), which adds another interpolation when creating a contour plot on an irregular grid. The algorithm consists in creating a triangulation of the initial grid and creates the contour from the triangulated mesh (see NCL documentation). From the figure, it can be seen that the horizontal interpolation provides close results with the initial data. With the exception of the smallest structures, the various shapes of the ozone field are well-preserved. As expected, some structures are lost near the poles due to the smaller number of cells of Pangolin.

We first examine the results of resolution on Pangolin for advection only. Here, and in the following results, the winds are corrected and the shape-preserving limiter is enabled. Results after one month are shown on Fig.4.4 for the various resolutions. It can be seen that the increase in resolution allows smaller structures to appear, such as the vortex in the South Pacific Ocean, and in the South Pole region around the polar vortex. To ensure these structures had a physical reality and were not the byproduct of either interpolation or the advection scheme, we thoroughly checked the output of the simulation every three hours. The structures formed of filaments result from elongation of the tracer in region of strong wind shear. This occurs in particular in the transition regions between the equatorial easterlies and polar westerlies. This situation is what can be expected from quasi-2D turbulence where tracers present filamentary structures in presence of large scale vortices.

The same simulations with the chemistry enabled are shown on Fig. 4.5. The plots show that chemistry plays an important role in the redistribution of the ozone concentration. Despite that all the structures induced by advection are still visible, their amplitudes are 'smoothed out' by chemical production or destruction with a tendency to relax the concentrations towards the climatological values of the linearized scheme.

The output of MOCAGE at the end of the simulation is shown on Fig. 4.6. With a resolution similar to MOCAGE, Pangolin is able to match the larger structures after a month (Fig. 4.5(a)). With twice the resolution (Fig. 4.5(b)), medium-scale structure are correctly modelled by Pangolin, such as the vortex in the South Pacific Ocean mentioned before or the long filament south of Australia. With a resolution six times finer (Fig. 4.5(c)), some small-scale structures are reconstructed by Pangolin, like in western Russia, above Japan or north of Chile. However, much of the smaller structures do not appear on the MOCAGE simulation as the resolution is too coarse for such levels of details. Also, the difference in the North pole can be explained by the difference in the chemistry scheme: MOCAGE uses actually all the terms from Eq. (4.2), while Pangolin uses only the first five. Also, the radiative balance is not perfect over the poles due to the lack of solar heating that does not counterbalance the infrared cooling, so the motions are not completely adiabatic. This means that vertical advection can play a larger role and the comparison of the model outputs on isentropic surfaces can only be qualitative. Nevertheless, we can conclude from these tests that Pangolin gives quality simulations for 2D advection and can be an alternative to MOCAGE if used with sufficient resolution. This result validates our strategy since high resolutions can be easily implemented within Pangolin due to its good performances running on parallel computers.

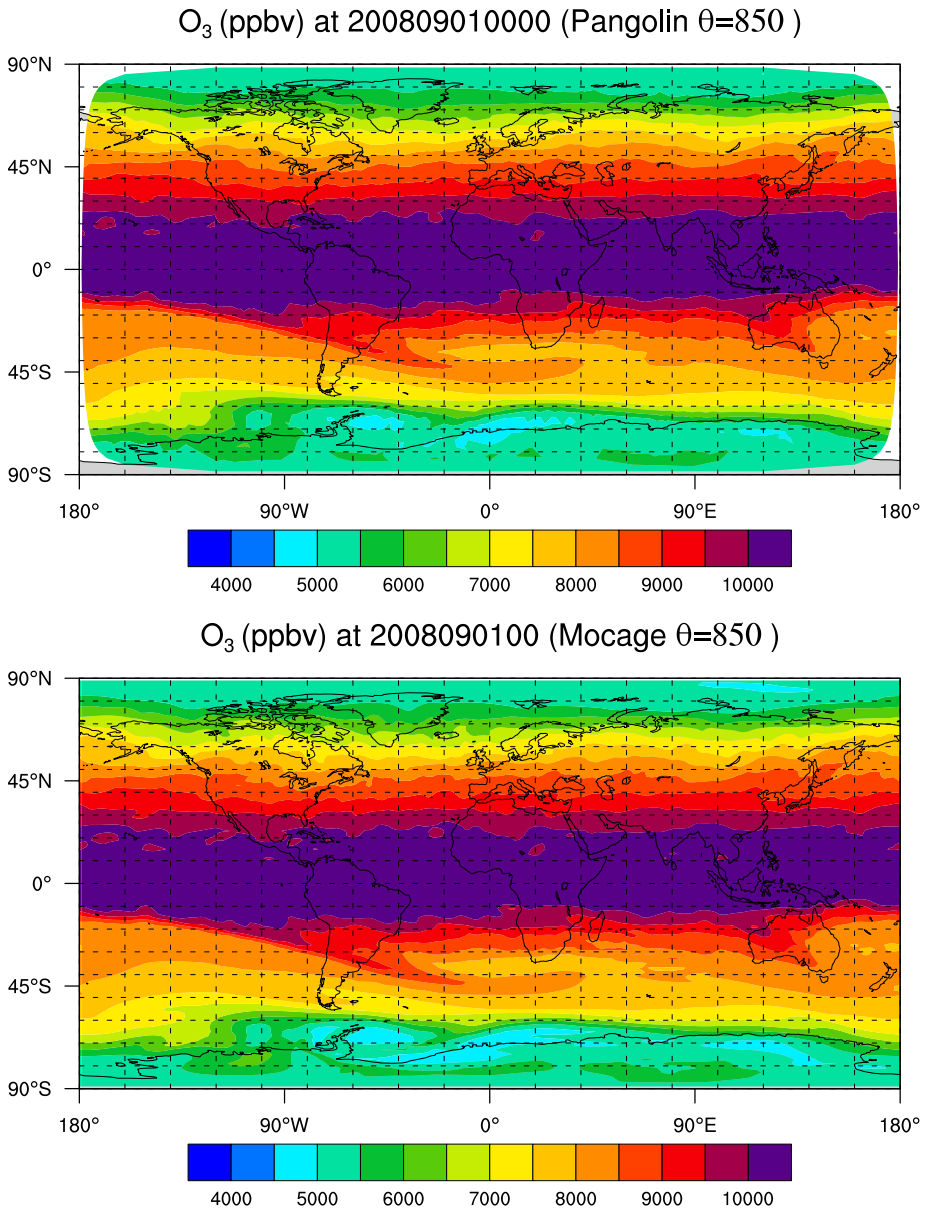
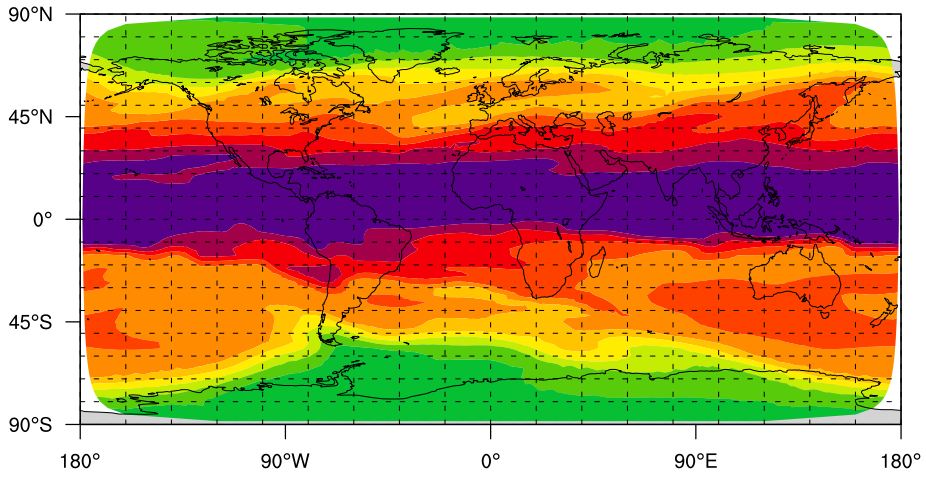
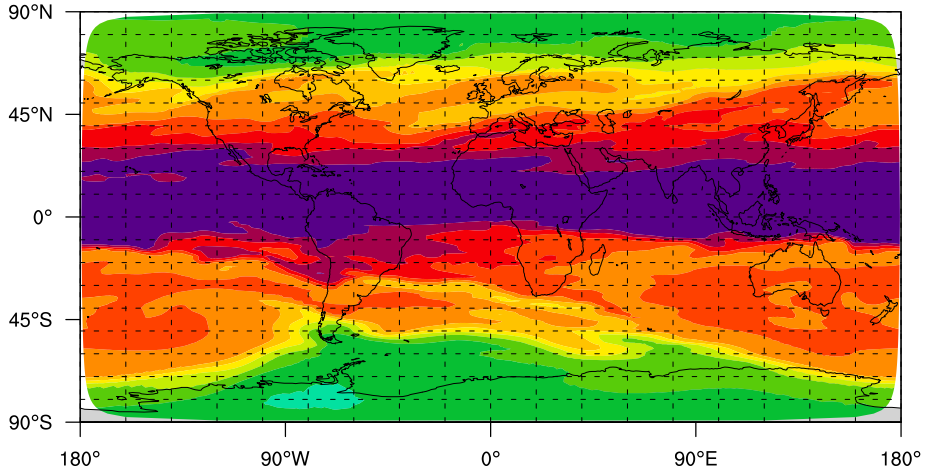


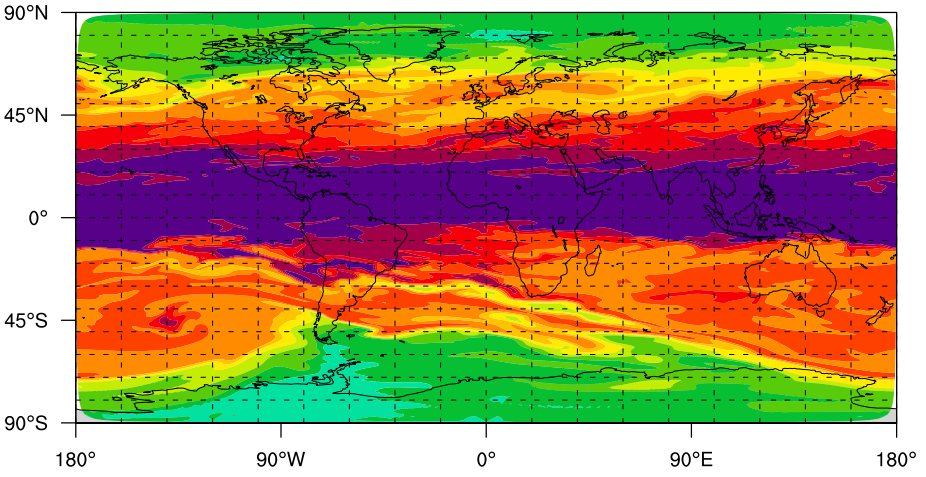
Figure 4.3: Comparison of the initial distributions for Pangolin and MOCAGE. Both are interpolated on the isentropic surface $\theta = 850$K. Ozone concentrations are given in particles per billion per volume (ppbv).



(a) $2.90 \times 1.97°$

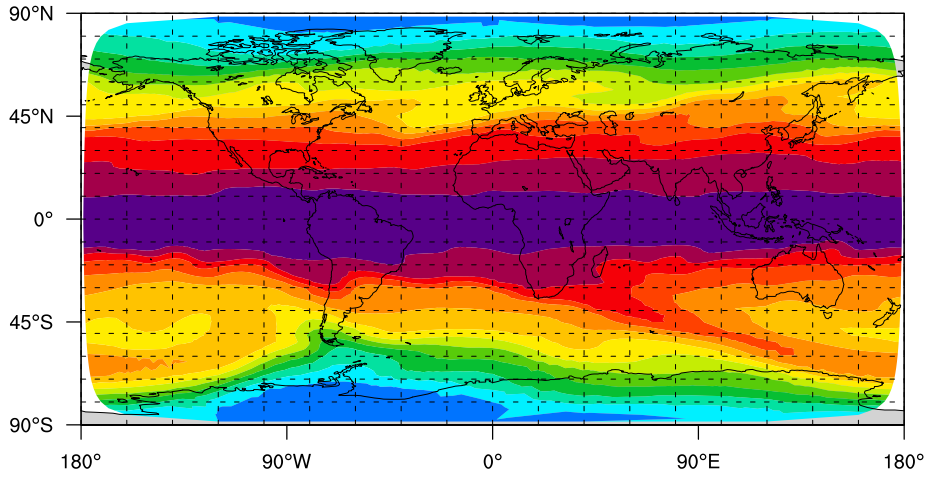


(b) $1.5 \times 1.00°$

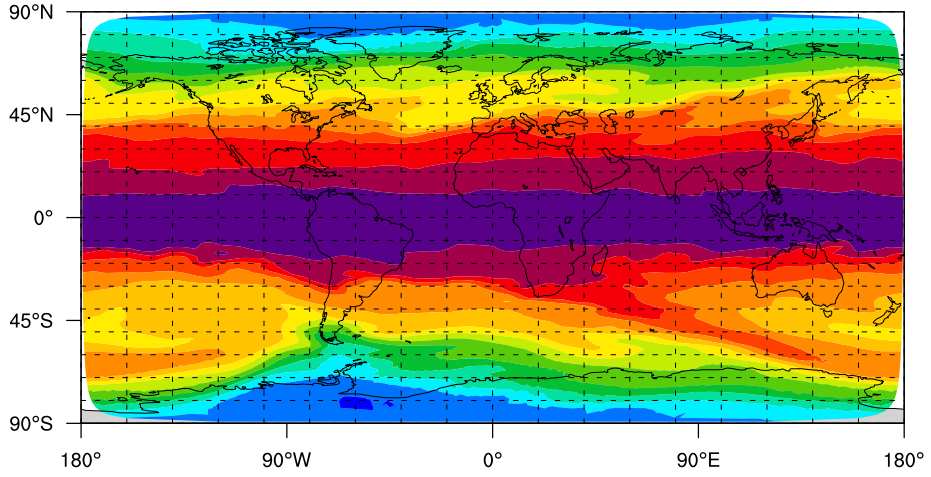


(c) $0.5 \times 0.33°$

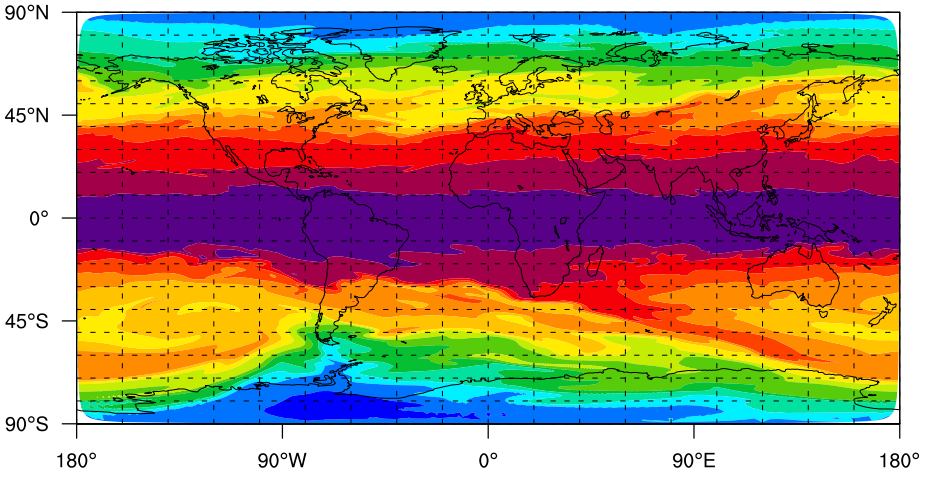
Figure 4.4: Impact of resolution on ozone distribution after a month with Pangolin. Only advection is enabled and the results are shown on the isentropic surface $\theta = 850$K. The legend is the same as on Fig. 4.3.



(a) $2.90 \times 1.97°$



(b) $1.5 \times 1.00°$



(c) $0.5 \times 0.33°$

Figure 4.5: Impact of increasing resolution on the ozone distribution after a month with Pangolin. Both advection and the linear ozone scheme are enabled and the results are shown on the isentropic surface $\theta = 850$K. The legend is the same as on Fig. 4.3.

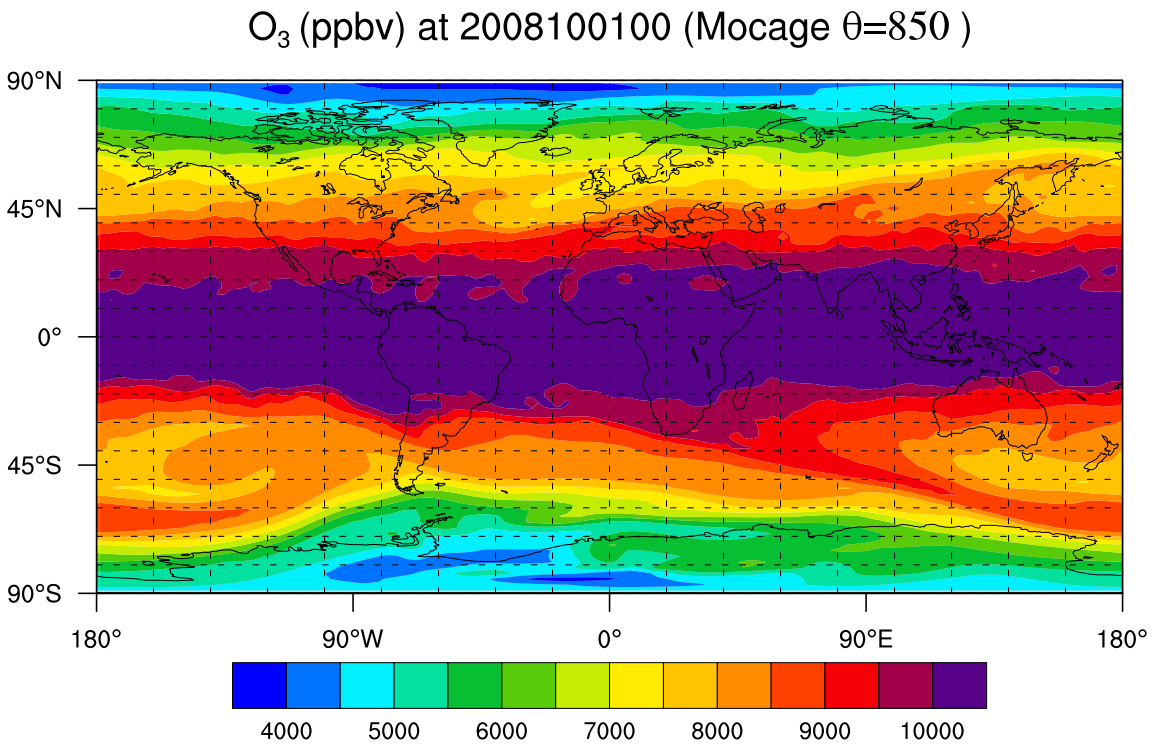


Figure 4.6: Ozone distribution after a month with MOCAGE with a resolution of $2° \times 2°$. The output is interpolated on isentropic coordinates $\theta = 850$K.

Conclusion

The goal of this thesis was to design and implement a framework for a global atmospheric chemistry-transport model, adapted to massively parallel emerging architectures. The characteristics of Pangolin have been chosen to obtain a scalable application for current and future parallel architectures. In particular, we focused on the message-passing paradigm applied to the advection to drive our parallelization strategy. We adopted an Eulerian approach for advection on the sphere associated with a specific domain decomposition that ease the parallelization of the code and appear competitive with the semi-Lagrangian approaches. In particular, the current CTM of Météo-France, MOCAGE, has a semi-Lagrangian advection scheme and the goal was to investigate an alternative approach to exploit more easily the parallel architecture of the large Météo-France's computers. In an Eulerian framework, a common issue of regular latitude-longitude grids is the convergence of meridians near the poles, which reduces the time-step as the resolution increases and therefore strongly penalizes the computational times. To bypass this 'pole issue', we proposed a new quasi-area-preserving grid where the number of cells decreases as we get closer to the poles. This grid has quasi-uniform cell surfaces and preserves a rectangular latitude-longitude structure.

As mass preservation is an important feature for the chemistry in a CTM, a finite-volume approach was chosen for advection on that grid. Global and local mass preservation is ensured by correcting the winds beforehand to ensure there are divergence free. Choosing the order of the scheme proved to be a compromise between accuracy and scability as higher-order schemes will increase the communication volume. As a result, a second-order van Leer scheme was chosen to favor scalability over accuracy. In practice, we expect to compensate the loss of accuracy with finer grid resolutions. The current bi-dimensional advection scheme and how it was adapted to our grid, along with general background information, was discussed in Chapter 1.

To validate the advection scheme adapted to the Pangolin grid, we examined key properties for an advection scheme, such as stability, accuracy and mass preservation (Chapter 2). Pangolin was tested on algebraic test cases and compared to other state-of-the-art transport schemes, using a recently published testing suite. The other schemes were chosen such as to be also second-order accurate and implemented on various grids. It was found that the van Leer scheme implemented in Pangolin is in practice to be first-order. This explained why Pangolin was most of the time less accurate than the models selected here. However, all the other schemes used a larger computational stencil and the loss in accuracy was compensated in all of the tests with a finer resolution. This is consistent with the initial design of the code where moderate accuracy was chosen in order to improve parallel performances.

We have then examined the parallel performances of Pangolin using the MPI library (Chapter 3). Due to our Eulerian approach, the parallelization of the advection scheme is reduced to a domain decomposition problem. The global grid is decomposed into connex subdomains of equal-area, which guarantees the load balancing for advection. Instead of using a professional tool, we developed our own partitioning algorithm to take advantage of the algebraic features of the Pangolin grid. This choice was made to decrease the memory footprint of Pangolin, in agreement with the possible future trends for computer design as discussed in Chapter 3. The partitioning however constrains the number of cores to achieve load-balancing. Even though the constraint can be relaxed to accommodate other configurations, parallel performances only improved for an optimal number of cores. We studied the parallel performances of the 2D advection scheme up to 294 cores. The scalability was found to be promising and it was estimated that addition of the chemistry will largely improve it. We also examined the maximal number of cores at a coarse resolutions in a strong-scaling study to find the limit of parallelism of 2D advection. As expected, it was found that the parallel performances decrease in extreme configurations where the subdomains are so small that our strategy for hiding communication cost is no longer relevant. With such extremely small subdomains, only a very small portion of the memory of the cores was used, making this configuration irrelevant for 'real-life' simulations.

After validating the parallel advection, a linear ozone chemistry scheme was added to examine the performances of Pangolin in 'real-life' configurations and compare it to MOCAGE simulations. Pangolin was run in an offline mode, where wind and temperature data is forced every three hours. While MOCAGE has a 3D advection scheme, advection in Pangolin is only bi-dimensional so it was integrated using an isentropic vertical coordinate to make the comparison relevant. It was found that generally, the large-scale structures were similar and converge to the results of MOCAGE when the resolution is increased.

Perspectives

Future versions of Pangolin will include an extension to 3D advection. The addition of the vertical advection will make the wind corrections more challenging. Also special care should be taken to obtain a non-divergent advection consistent with the model providing the wind velocities. Adding the vertical advection will use the cores more effectively and as such improve the parallel performances. At the moment, 2D advection is not 'expensive' enough to justify more than a few hundred of cores if the horizontal resolution at the Equator is limited to around $1^\circ \times 1^\circ$. If a complex chemistry scheme is added, the chemical calculations should be expensive enough to greatly benefit from the parallelization strategy adopted by Pangolin. Therefore, a larger number of cores could be used in practice.

However, the addition of chemistry will most likely introduce load unbalances due to several physical processes occurring not uniformly over the sphere (heterogeneous chemistry, surface emissions, convection, vertical diffusion or the day/night transitions). Several strategies are possible to mitigate this load unbalance, the most promising being the combination of OpenMP and MPI for a hybrid parallel programming.

Conclusion

Le but de cette thèse était de concevoir et d'implanter une infrastructure pour un modèle de chimie-transport atmosphérique global qui soit adapté aux architectures massivement parallèles. Les caractéristiques de Pangolin ont été choisies afin que le modèle passes à l'échelle sur les architectures parallèles actuelles et futures. Nous nous sommes en particulier penchés sur le paradigme de programmation à base d'échanges de messages en l'appliquant à l'advection, ce qui a guidé notre stratégie de parallélisation. Un approche Eulérienne a été retenue pour le schéma d'advection sur la sphère et combinée avec un nouvel algorithme de décomposition de domaines. Cela a permis de faciliter la parallélisation du code et semble pouvoir concurrencer les approches semi-Lagrangiennes. En particulier, le modèle de chimie-transport de Météo-France, MOCAGE, utilise un schéma semi-Lagrangien. L'objectif de la thèse était d'examiner une approche alternative à MOCAGE pour exploiter les supercalculateurs de Météo-France. Un problème classique de la grille régulière latitude-longitude provient de la convergence des méridiens aux pôles, qui réduit le pas de temps lorsque la résolution augmente et pénalise donc fortement le temps de calcul. Pour éviter ce problème aux pôles, nous avons proposé une nouvelle grille préservant approximativement les aires des cellules. Comme le nombre de cellules décroît près des pôles, les surfaces des cellules sont quasiment uniformes sur la sphère. La grille possède également une structure rectangulaire latitude-longitude.

La conservation de la masse est une propriété important pour les MCTs, c'est pourquoi une approche de type volume-finis a été utilisée sur notre grille. En corrigeant les vents en amont pour rendre leur divergence nulle, on obtient une conservation à la fois globale et locale de la masse. Le choix de l'ordre du schéma d'advection implique de faire un compromis entre la précision et la scalabilité car un ordre plus élevé augmentera le volume de communications. C'est pourquoi un schéma du second-ordre de type van Leer a été retenu, afin de privilégier la scalabilité. En pratique, la perte en précision sera compensée par des résolutions plus fines. Le schéma 2D ainsi que la manière dont il a été adapté à notre grille est présenté dans le chapitre 1, avec une présentation plus générale de la problématique du transport.

Afin de valider le schéma d'advection sur notre grille, nous avons examiné des propriétés essentielles pour un schéma d'advection, à savoir la stabilité, la précision et la préservation de la masse (chapitre 2). De plus, Pangolin a été testé avec des cas tests analytiques et comparé à d'autres schémas de transport à l'aide d'un banc d'essai récemment publié. Les autres schémas ont été choisis tels qu'ils soient également du second-ordre et qu'ils soient implémentés sur différentes grilles. Les tests ont montré que le schéma de type van Leer implémenté dans Pangolin était en pratique

d'ordre un. Cela explique pourquoi la plupart des tests ont montré que Pangolin était moins précis. Ce résultat doit être nuancé par le fait que les autres modèles utilisent un 'halo' plus large. La différence en précision a été aussi compensée avec une résolution plus fine, ce qui est cohérent avec les hypothèses de conception où la scalabilité est privilégiée par rapport à la précision.

Dans le chapitre 3, les performances en parallèle de Pangolin à l'aide de la librairie MPI ont été étudiées. L'approche Eulérienne retenue implique que la parallélisation de l'advection se ramène à un problème de décomposition de domaines. Il consiste à découper la grille en sous-domaines connexes de même taille afin de garantir l'équilibrage de la charge de calcul. Au lieu d'utiliser des outils déjà existants, nous avons développé un nouvel algorithme de partitionnement afin de prendre en compte les propriétés analytiques de la grille. Ce choix provient de la volonté de diminuer le coût en mémoire du modèle afin d'anticiper les tendances envisagées dans le chapitre 3. Cependant, le partitionnement impose une contrainte sur le nombre de cœurs pour que la charge de travail soit équilibrée. La contrainte peut être assouplie mais les performances parallèles ne seront améliorées que pour un nombre optimal de cœurs. Les performances en parallèles ont été testées pour l'advection avec 294 cœurs. Les résultats sont prometteurs et l'ajout de la chimie améliorera probablement fortement les performances. Nous avons également examiné le nombre maximal de cœurs pour une résolution grossière dans un test de *strong-scaling* afin de déterminer la limite du parallélisme pour l'advection 2D. Il a été confirmé que les performances parallèles diminuent fortement quand les sous-domaines sont extrêmement petits. De telles configurations rendent notre stratégie de 'recouvrement' des coûts de communications inefficace mais les sous-domaines sont suffisament petits pour que la mémoire des cœurs soient très peu utilisée. Ainsi, ces configurations ne correspondent pas à des situations réalistes.

Afin d'effectuer une comparaison avec MOCAGE, un schéma linéaire pour l'ozone a été ajouté pour examiner les performances de Pangolin dans des situations réalistes. Pangolin a été utilisé en mode offline, où les vents et la température sont forcés toutes les trois heures. Comme Pangolin ne dispose actuelle que d'une advection 2D, nous avons utilisé des coordonnées isentropes pour que la comparaison avec l'advection 3D de MOCAGE soit pertinente. Les résultats ont montré que les structures les plus importantes sont bien préservées et que Pangolin converge vers les résultats de MOCAGE lorsque la résolution diminue.

Perspectives

Une future version de Pangolin intègrera l'extension 3D pour l'advection. L'ajout de l'advection verticale rendra la correction des vents plus délicate. Il faudra aussi vérifier que l'advection non divergence soit cohérente avec le modèle d'où proviennent les vents. Avec l'advection verticale, les cœurs seront utilisés de manière plus efficace et cela améliorera les performances parallèles. Pour l'instant, l'advection 2D ne justifie pas d'utiliser plus d'une centaine de cœurs pour une résolution de $1 \times 1°$ à l'Équateur. Avec une chimie complexe, la charge de calcul devrait être suffisamment importante pour utiliser un nombre bien plus important de cœurs.

Cependant, l'ajout de la chimie s'accompagnera très probablement d'un déséquilibre de la charge de calcul, due à la non-uniformité de certains processus physiques (chimie hétérogène, émissions de surface, convection, diffusion verticale ou la transition jour/nuit). Plusieurs stratégies sont possibles pour y remédier mais la plus prometteuse semble être une combinaison d'OpenMP et MPI pour une programmation parallèle hybride.

Appendix A

Pangolin v1.0, a conservative 2-D transport model for large scale parallel calculation

This paper was accepted by the GMD journal.

Introduction

Global three-dimensional chemistry-transport models (hereafter referred to as CTMs) play an important role in monitoring and predicting the composition of the atmosphere (e.g., Chipperfield, 2006; Teyssèdre et al., 2007; Huijnen et al., 2010). Those models include large-scale transport, emissions and chemical transformations of trace species, and sub-scale grid processes like convection and deposition. In CTMs, advection by large-scale winds is a key process that must be handled by numerical algorithms. For these algorithms, mass conservation for the considered species, monotonicity and numerical accuracy are especially important for long simulations where accumulation of errors and bias must be avoided.

In this paper, we present a conservative advection model on the sphere which is intended to form the basic framework for a future CTM. The adopted scheme is based on a flux-form (Eulerian) tracer advection algorithm on a reduced latitude–longitude grid. A finite-volume approach was chosen as it provides an easy way to ensure mass preservation. Furthermore, parallelizing the model is then reduced to a classical domain decomposition problem.

The specificity of our model, named Pangolin[1], lies in the grid definition, where the number of cells on a latitude circle is progressively decreased towards the pole in order to obtain a grid which approximately preserves the cell areas at mid- and high latitudes. This avoids the so-called "pole problem" arising from the convergence of the meridians, which severely limits the size of the time steps for Eulerian models. Several approaches have been adopted in previous studies to cope with this issue. Finite-volume schemes on latitude–longitude grids often aggregate latitudinal fluxes near the poles (e.g., Hourdin and Armengaud (1999)) or successively double the grid size at high latitudes (e.g., Belikov et al. (2011)).

Alternatively, quasi-uniform spherical grids have been developed, such as a cubed-sphere[2], com-

[1] PArallel implementatioN of a larGe scale multi-dimensiOnaL chemIstry-traNsport scheme

[2] Each face uses Cartesian coordinates and is projected gnomonically or conformally on the sphere.

posite mesh – Yin-Yang – or icosahedral grids[3]. A review of the different grids can be found in Staniforth and Thuburn (2012) and Williamson (2007). However, those approaches lose the latitudinal regularity arising from the rotation of the Earth. Furthermore, they require specific treatments at the singularities of the adopted polygons, which may also induce resolution clustering near these points. On the plus side, they allow for the implementation of more accurate algorithms than the ones on reduced latitude–longitude grids. This last point is especially important for weather and climate models that solve the nonlinear momentum equation, but is less stringent for the two-dimensional (2-D) linear transport of trace species on the sphere.

To construct the reduced grid, one difficulty is to define a structure which avoids treating the poles as special cases, as that can impact the precision and properties of the advection algorithm. Thus we have chosen to adopt a semi-structured approach. The grid is not regular as the number of cells varies with the latitude, but the coordinates of cell interfaces can be computed algebraically. We thus avoid storing a list of neighbors as an irregular unstructured grid would normally require, hence decreasing memory costs. This was done to anticipate future parallel architectures, which may have less memory capacity per core than current systems.

Our goal is to use an adequate algorithm exploiting the grid features to achieve efficiency and scalability on massively parallel architectures. Fine control over parallelization was obtained using the Message Passing Interface (MPI) library. In that context, the advection scheme must be chosen as to balance its accuracy vs. the volume of the required parallel communications. Furthermore, grid properties were carefully studied to improve the parallel version. In particular, a custom domain decomposition consistent with our grid was designed.

The present paper is organized as follows. Section A.1 lists the basic equations and numerical methods used to solve the advection of the chemical species. In Sect. A.2, results from standard test cases for advection of tracer on the sphere are reported. Those cases were chosen from the test case suite proposed by Lauritzen et al. (2012). Section A.3 gives details on the model implementation on parallel architectures and the results of parallel scalability experiments. In Sect. A.4, we summarize the results obtained and discuss the possible extension of our method.

A.1 Numerical scheme

Finite-volume formulation

Our model is based on a finite-volume method to integrate the tracer advection equation. This is performed on a bi-dimensional discrete grid on the sphere, which is described in more detail in Sect. A.1. In each grid cell, the tracer concentration changes according to the divergence of the fluxes at the cell boundaries. This comes from the flux form of the continuity and tracer conservation equations:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0, \tag{A.1}$$

$$\frac{\partial \rho q}{\partial t} + \nabla \cdot (\rho q \vec{V}) = 0, \tag{A.2}$$

where ρ is the air density, q the tracer mixing ratio and $\vec{V}$ the winds vector field. Equations (A.1) and (A.2) are first integrated over a cell area $\mathcal{A}$. With $\partial\mathcal{A}$ noted as the cell boundary and $\vec{n}$ as the

[3]An icosahedra is subdivided until the desired resolution and projected on the sphere.

local normal pointing outward, the divergence theorem yields

$$\frac{\partial m}{\partial t} = \frac{\partial}{\partial t}\int_{\mathcal{A}} \rho = \int_{\partial\mathcal{A}} (\rho\vec{V}\cdot\vec{n}\,\mathrm{d}S), \tag{A.3}$$

$$\frac{\partial m_r}{\partial t} = \frac{\partial}{\partial t}\int_{\mathcal{A}} \rho q = \int_{\partial\mathcal{A}} (\rho q\vec{V}\cdot\vec{n}\,\mathrm{d}S), \tag{A.4}$$

where m is the total air mass in a cell and m_r is the total tracer mass in a cell. The right-hand side of Eqs. (A.3) and (A.4) can be seen as the integral of all the fluxes across the cell boundaries. This formulation gives a conservative scheme when the same fluxes are used for upstream and downstream adjacent cells.

The above equations are then integrated in time during a time step:

$$m^{n+1} = m^n - \int_{t_n}^{t_{n+1}} \Big(\int_{\partial\mathcal{A}} \rho\vec{V}\cdot\vec{n}\,\mathrm{d}S\Big)\mathrm{d}t, \tag{A.5}$$

$$m_r^{n+1} = m_r^n - \int_{t_n}^{t_{n+1}} \Big(\int_{\partial\mathcal{A}} \rho q\vec{V}\cdot\vec{n}\,\mathrm{d}S\Big)\mathrm{d}t. \tag{A.6}$$

Equation (A.5) can be omitted for non-divergent flows since the divergence of the mass fluxes is null and the mass inside the cells is constant: $[m]^{n+1} = [m]^n$. In this paper, we only consider non-divergent flows. As such, winds are corrected to be divergence-free in a preprocessing step, as explained in Sect. A.1. Handling divergent flows would require minor adjustments to the scheme (removing the correction of winds and adding Eq. (A.5) to the scheme), but this configuration was not considered as a typical use case of CTMs, where large-scale 3-D winds can be considered divergence-free.

There are many options to evaluate the tracer mass fluxes at the cell boundary. These fluxes are approximated as the air mass fluxes multiplied by the mean tracer ratio $\widehat{q}$ crossing the interface. The simplest approach to evaluate $\widehat{q}$ was introduced by Godunov (Godunov et al., 1961), who considered $\widehat{q}$ as constant within each upstream cell. The resulting scheme is conservative and monotonicity preserving but very diffusive. Improvements to the Godunov scheme were introduced by van Leer in van Leer (1977), where q is now approximated by a non-constant polynomial function. Depending on the polynomial degree used, i.e., the order moments of the distribution of q inside the cell, van Leer obtained several schemes (up to six) that varied in complexity. A review of the different possible options can be found in Rood (1987) and Hourdin and Armengaud (1999). In general, accuracy is found to increase when higher-order moments are used, but the price to pay lies in larger computational and memory costs. Using higher moments requires a larger number of grid points to compute the derivatives, which increases communication volumes when domain decomposition techniques are used for parallel clusters (see Sect. A.3).

For our model, we have adopted a first-order reconstruction, the van Leer scheme (noted as van Leer I in the original paper). The distribution of q in the cells is approximated by a linear function in latitudinal and meridional directions. The slope of the linear function is computed as a finite difference using the values of q within the nearest cells. In that configuration, the scheme is second-order accurate in space. To extend the algorithm to multiple dimensions, a time-splitting

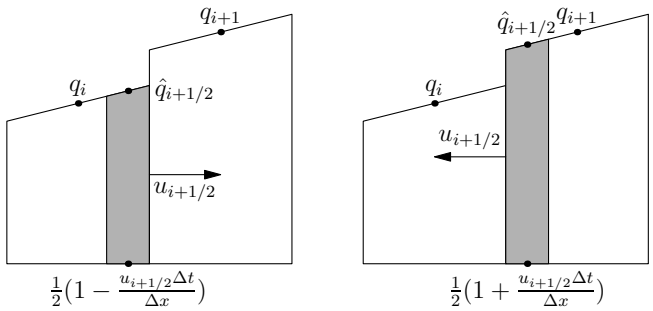


Figure 1.1: Van Leer scheme for positive (left) and negative (right) winds. The distribution of the tracer is shown as a linear distribution (broken line). The grey area is the quantity of tracer passing through the interface during a time step.

scheme is used. Equations (A.5) and (A.6) are first integrated in the zonal direction:

$$\begin{aligned} \widetilde{m}_i^{n+1} &= m_i^n + U_{i-1/2} - U_{i+1/2}, \\ (\widetilde{m_r})_i^{n+1} &= (m_r)_i^n + F_{i-1/2} - F_{i+1/2}, \end{aligned} \tag{A.7}$$

where U and F are the air and tracer fluxes, respectively, across the borders orthogonal to the chosen direction during a time step. With these notation, tracer fluxes are approximated as $F_{i+1/2} \approx \widehat{q}_{i+1/2} u_{i+1/2}$. Figure 1.1 illustrates the reconstruction of $\widehat{q}_{i+1/2}$. The linear distribution represents the tracer distribution in the 1-D case for cells i and $i+1$. Finding $\widehat{q}_{i+1/2}$ depends on the wind direction: for outward winds ($u_{i+1/2} > 0$), the grey area in the left diagram will move from cell i to cell $i+1$. Then the mean tracer ratio corresponding to this flux is computed from the distribution in cell i. The same can be applied for inward fluxes, resulting in

$$\widehat{q}_{i+1/2} = \begin{cases} q_i + \left(1 - \frac{u_{i+1/2}\Delta t}{\Delta x}\right)(\delta q)_i & \text{if} \quad u_{i+1/2} > 0, \\ q_{i+1} - \left(1 + \frac{u_{i+1/2}\Delta t}{\Delta x}\right)(\delta q)_{i+1} & \text{otherwise,} \end{cases}$$

where δq is the slope of the linear reconstruction and $u_{i+1/2}$ the wind at the interface. Δt and Δx are the time step and cell spacing, respectively.

This first advection step gives us the intermediate mass value $\widetilde{m}$ and tracer value $q' = \widetilde{m}/\widetilde{m_r}$. These new values are then used to integrate Eq. (A.6) in the meridional direction. As the grid is unstructured, mass and tracer fluxes in the north–south direction have to be evaluated for all the neighbors of each cell. This is detailed in Sect. A.1.

It should be noted that other time-splitting schemes are available. Another approach is to use a zonal–meridional advection during a time step and meridional–zonal for the next. It is also possible to compute both zonal–meridional and meridional–zonal advection and then use their mean as the final value. For the complete description of each method, see Machenhauer et al. (2009). Either way, the final bidimensional algorithm is second-order accurate. In Pangolin, all three time-splitting schemes have been tested using the numerical order of convergence test (see Sect. A.2). It was found the choice of the time-splitting algorithm has little impact on accuracy.

To ensure monotonicity of the solution and to prevent numerical oscillations, van Leer introduced

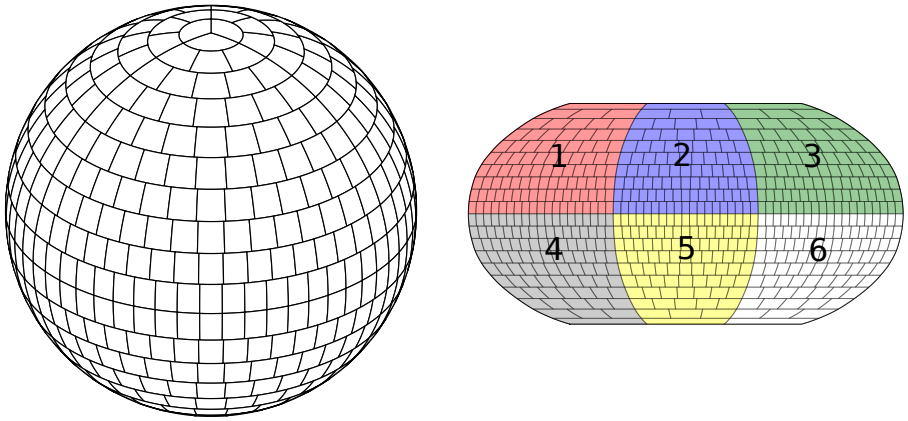


Figure 1.2: Grid used in Pangolin with 20 latitudes: orthographic projection (left) and Robinson projection, with the six identical zones highlighted (right).

a slope limiter. The idea is to limit the slope value within a given cell such as the tracer value in that cell does not exceed the mean value of the adjacent cells. This is more restrictive than limiting the fluxes to ensure that the tracer values remain between the maximum–minimum values of the adjacent cells but is easier to implement. The slope is given by

$$(\delta q)_i = \min\left(\frac{1}{2}|q_{i+1} - q_{i-1}|, 2|q_{i+1} - q_i|, 2|q_i - q_{i-1}|\right) \times \text{sign}(q_{i+1} - q_i) \tag{A.8}$$

if q_i lies in between q_{i-1} and q_{i+1}, and $(\delta q)_i = 0$ otherwise. As discussed by Hourdin and Armengaud (1999), the slope limiter efficiently damps the numerical oscillations but introduces more diffusion of the numerical solutions. For the test cases reported in this study, the slope limiter appears to have little impact on the accuracy of the numerical solutions.

Grid

The grid used in Pangolin is completely defined by the number of cells at the North Pole and the number of latitudes n_{lat} on a hemisphere. The Southern Hemisphere is simply constructed in the same way as the Northern Hemisphere. To find the number of cells at the North pole, we can write the equality between cell areas at the pole and at the Equator. If we consider squared cells at the Equator, like the ones on a regular latitude–longitude grid, and small latitudinal spacing, the number of cells is approximately π. Thus we set the number of cells at the poles to 3.

At a given latitude, all cells have the same area. We can then compute the number of cells for all latitudes. For maximum flexibility, latitudinal and longitudinal spacings are no longer assumed identical. Let us consider the area of cell (i, j), noted $\mathcal{A}_i$ as it does not depends on j. Let us note ϕ_i as the colatitude and λ_{ij} as the position of the south and east cell borders. As we assume that the cell spacings are constant, we can write $\phi_i = i\Delta\phi_i$ and $\lambda_{ij} = j\Delta\lambda_i$. The area is defined on

$\Omega_{ij} = [\lambda_j, \lambda_{j+1}] \times [\phi_{i-1}, \phi_i]$, so in spherical coordinates we have

$$\begin{aligned}\mathcal{A}_i &= \iint_{\Omega_{ij}} r^2 \sin\phi \mathrm{d}\lambda \mathrm{d}\phi \\ &= r^2 \Delta\lambda_i \big(\cos(\phi_{i-1}) - \cos(\phi_{i-1} + \Delta\phi_i)\big).\end{aligned} \tag{A.9}$$

Areas are preserved so $\mathcal{A}_i = \mathcal{A}_1$:

$$\Delta\lambda_i = \Delta\lambda_1 \frac{1 - \cos(\Delta\phi_1)}{2\sin\left(\frac{\Delta\phi_i}{2}\right)\sin\left(\left(i - \frac{1}{2}\right)\Delta\phi_i\right)}.$$

Noting n_i the number of cells at colatitude i, we get

$$\frac{n_i}{n_1} = \left\lfloor \frac{\Delta\lambda_1}{\Delta\lambda_i} \right\rfloor = \left\lfloor \frac{2\sin\left(\frac{\Delta\phi_i}{2}\right)\sin\left(\left(i - \frac{1}{2}\right)\Delta\phi_i\right)}{1 - \cos(\Delta\phi_1)} \right\rfloor.$$

Now let us assume $\forall i$, $\Delta\phi_i$ and $(i - \frac{1}{2})\Delta\phi_i$ are small enough:

$$\frac{n_i}{n_1} \approx \left\lfloor \frac{2\frac{\Delta\phi_i}{2}\left(i - \frac{1}{2}\right)\Delta\phi_i}{\frac{\Delta\phi_1^2}{2}} \right\rfloor = 2i - 1.$$

Finally, we can define the number of cells for the whole grid, with $2n_{\mathrm{lat}}$ latitudes, as

$$n_i = \begin{cases} 3(2i-1) & \text{if } 1 \le i \le n_{\mathrm{lat}}, \\ n_{2n_{\mathrm{lat}}-i+1} & \text{otherwise.} \end{cases} \tag{A.10}$$

It follows that the total number of cells on the grid is $6n_{\mathrm{lat}}^2$. As an illustration, the grid is shown in Fig. 1.2.

The previous formula is a sound approximation for area preservation near the poles and when latitudinal spacing is constant. In practice, we consider the approximation as reasonable up to 75°: the relative error is then less than 1 %. At lower latitudes, the error increases, with a maximum of 56 % at the Equator. So the grid used in Pangolin gives higher resolutions at the Equator than at the poles. One way around this issue is to truncate the number of cells at a given threshold. As a comparison, Fig. 1.3 shows the number of cells for Pangolin with the "exact" and truncated version. By "exact", we mean the number of cells comes from the area-preservation formulae without any approximations for $\Delta\phi_i$. Furthermore, to truly preserve the cell areas, we should use a variable latitudinal spacing. However, the distortion due to a constant latitudinal spacing was found to be acceptable and much less pronounced compared with a regular latitude–longitude grid.

The formula given in Eq. (A.10) allows us to easily determine the coordinates of the cell neighbors in each zone. The grid used in Pangolin has four axes of symmetry – $\lambda = 0$, $\lambda = 120$, $\lambda = 240°$ and the Equator – and so the grid can be split into six identical *zones*, numbered with regard to west to east and north to south as shown in Fig. 1.2. The following formulae to compute the position of the cell neighbors are given for the first zone – i.e., $[90°, 0] \times [0, 120°]$. The formulae on zones 2 and 3 are obtained by adding the proper offset. In the Southern Hemisphere (zone 4 to 6), the north and south formulae are simply inverted. The zonal neighbors for cell (i, j) are simply $(i, j-1)$ and

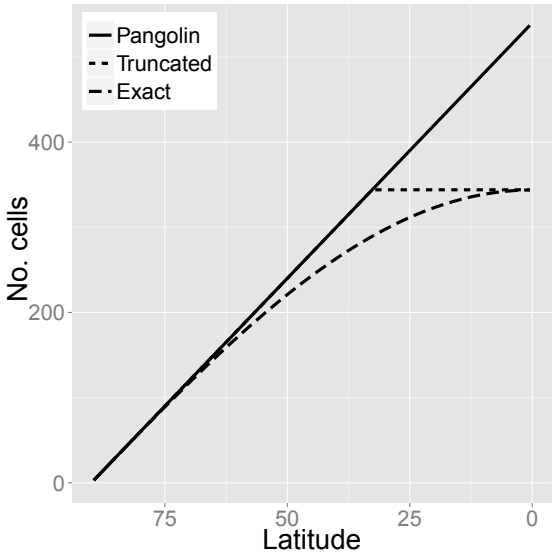


Figure 1.3: Number of cells for the grid used by Pangolin on one hemisphere with 90 latitudes (solid line). The truncated and "exact" version are shown as dotted and dashed lines, respectively.

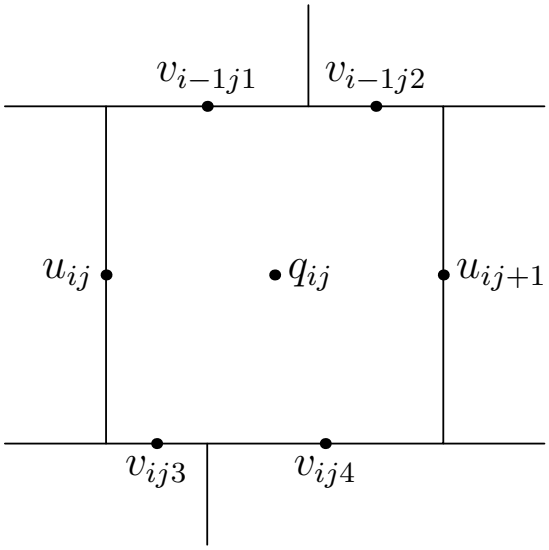


Figure 1.4: Discretization for zonal and meridional winds (u and v, respectively) and tracer mixing ratio q.

$(i, j+1)$. For the first zone, its north and south neighbors are respectively $\{(i-1, j_1), \ldots, (i-1, j_2)\}$ and $\{(i+1, j_3), \ldots, (i+1, j_4)\}$, with

$$j_1 = \left\lfloor \frac{n_{i-1}}{n_i}(j-1) + 1 \right\rfloor, \quad j_2 = \left\lceil \frac{n_{i-1}}{n_i} j \right\rceil,$$
$$j_3 = \left\lfloor \frac{n_{i+1}}{n_i}(j-1) + 1 \right\rfloor, \quad j_4 = \left\lceil \frac{n_{i+1}}{n_i} j \right\rceil. \tag{A.11}$$

From that formulation, it follows that the number of meridional neighbors is not constant, even though most of cells have two north and two south adjacent cells. Special cases include the middle of each sector (one north and three south neighbors) and its extremities (one north and two south). These figures apply for the Northern Hemisphere and must be inverted in the Southern Hemisphere.

As a consequence, computing the position of the neighbors is quite efficient, involving mostly integer operations and roundings. These computations are thus performed on the fly to reduce the storage requirements. In a more general way, the algebraic properties of the grid are exploited as much as possible. Our parallelization strategy relies heavily on the properties of the grid, as shown in Sect. A.3.

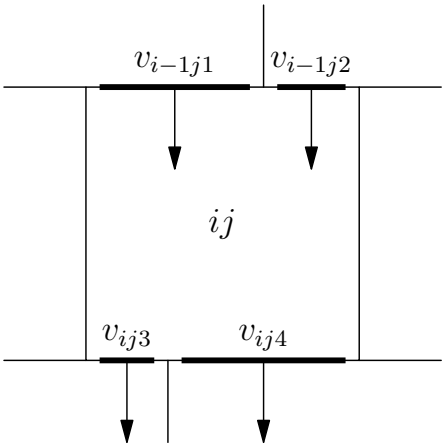


Figure 1.5: Meridional interfaces (bold lines) and fluxes (arrows) for cell (i, j).

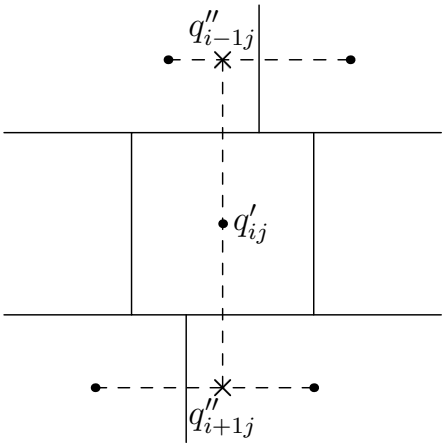


Figure 1.6: Zonal interpolation to compute the meridional gradient of q'_{ij}.

Adapting the scheme to the grid

For winds and tracer discretization, we adapt the Arakawa C grid (Arakawa and Lamb, 1977) to our scheme, as shown in Fig. 1.4. To avoid interpolating the winds components during advection, winds are taken at the middle of the interfaces. Tracer concentrations are defined at the centers of the cells.

Due to the structure of the grid (Fig. 1.2), air and tracer fluxes need to be computed for all the neighbors of the cells as illustrated in Fig. 1.5. For each flux, the frontier between the current cell and its neighbor is computed algebraically using the cell neighbors formulae (Eq. A.11). While there is no special treatment for zonal advection, meridional advection requires an interpolation to compute the meridional gradient. A linear interpolation in the zonal direction is computed to evaluate the value of the mixing ratio on the meridian passing through the center of the cell (see Fig. 1.6). These values are used to compute the meridional gradient by finite difference.

It is critical for mass preservation that winds have a null divergence, so we correct interpolated winds (both zonal and meridional) to achieve that. The first step in our correction deals with meridional winds. If we consider all the cells on a latitude circle, the total mass variation in this "band" only comes from the north and south meridional fluxes. Now, if we consider the latitude circle containing all south meridional fluxes, it constitutes a closed contour. Therefore, the wind divergence is null and the sum of all meridional winds must be zero. Meridional winds are thus corrected by removing the mean value from the interpolated values. For a future 3-D case, a preprocessing step involving vertical winds will be needed to ensure non-divergent circulation. Depending on the system of vertical coordinates used, the "mass-winds inconsistency" issue (see, for example, Jöckel et al. (2001)) will have to be addressed.

Then we correct zonal winds to ensure that the sum of all fluxes is locally null in each cell. As meridional winds are already corrected, only east and west zonal winds of each cell must be modified. We take zonal winds at longitude 0 as a reference and browse each cell sequentially from west to east to progressively correct each of the zonal winds and ensure mass preservation.

Finally, we need to take care that fluxes in a given direction do not completely empty the cells

Table A.1: For a given resolution at the Equator, we compare the total number of cells of each model n_{model} vs. the total number of cells of Pangolin n_{pangolin}.

Model	$n_{\text{model}}/n_{\text{pangolin}}$	n_{model}
FARSIGHT	2	$6 \cdot (90/\Delta\lambda)^2$
CLAW	0.68	$2 \cdot (90/\Delta\lambda)^2$
SLFV-ML	2.17	NA
CAM-FV	2.7	$\lfloor 360/\Delta\lambda \rfloor \cdot \lfloor 180/\Delta\lambda \rfloor$
UCISOM	2.7	$\lfloor 360/\Delta\lambda \rfloor \cdot \lfloor 180/\Delta\lambda \rfloor$
Pangolin	1	$6 \cdot \lceil 0.5 \cdot \lceil 120/\Delta\lambda \rceil + 1 \rceil^2$

during an advection step. For each cell, a local advective Courant number restricts the time step in order to avoid this situation. As advection is performed sequentially in two different directions, we define two unidimensional local Courant numbers. Then the global Courant number C is simply defined as the most restrictive condition on all cells:

$$C = \max_{ij} \left(\frac{u_{ij}\Delta t}{\Delta\phi_{ij}}, \frac{\Delta t \sum_{k\in V_{ij}} v_k \Delta\lambda_k}{\Delta\phi_{ij} \sum_{k\in V_{ij}} \Delta\lambda_k} \right),$$

where V_{ij} is the set of meridional neighbors for the cell (i, j) and $\Delta\lambda_k$ is the interface size between the cell and its neighbor. For the tests in this paper, we use $C_{\max} = 0.96$ as a Courant–Friedrichs–Lewy (CFL) condition.

A.2 Testing suite

A standard 2-D testing suite to check the accuracy and properties of a transport model was proposed in Lauritzen et al. (2012). A comparison with state-of-the-art schemes was subsequently published in Lauritzen et al. (2013), which offers a convenient benchmark to compare transport models on the sphere. From it, we have extracted a subset of the models and cases which we felt were relevant to Pangolin. In Lauritzen et al. (2013), the different grids were compared with a constant resolution at the Equator. In the present paper, we retain simulations performed with a constant total number of cells. The number of cells in each model was computed using the resolution at the Equator given in the appendix of Lauritzen et al. (2013). As a summary, Table A.1 contains the formulae used and gives an idea of the size of each grid in comparison with Pangolin.

Models features

The models were chosen as their spatial order is similar to Pangolin. They are implemented on both regular and non-regular grids and provide a basis for a comparison between semi-Lagrangian, finite-volume and wave propagation methods. A summary is given in Table A.2. Other features are described below:

- FARSIGHT is a grid-point semi-Lagrangian model, running on parallel architectures.
- CLAW uses a wave propagation technique with a first-order method (donor cell upwind) in each direction.

- SLFV-ML (Slope Limited Finite Volume scheme with Method of Lines), a flux-form finite volume with simplified swept area and linear reconstruction.

- CAM-FV (Community Atmosphere Model Finite-Volume) is a finite-volume model on a regular latitude–longitude grid. It uses the piecewise parabolic method (PPM), with the addition of a slope and curvature limiters.

- USICOM (UC Irvine Second-Order Moments scheme) is also a flux-form finite volume. It uses an improved version of Prather's second-order moments scheme on an Eulerian regular latitude–longitude grid.

All of these models are mass preserving, so the comparison with Pangolin is relevant. CAM-FV and UCISOM are of particular interest as they also use a directional splitting algorithm. Furthermore, all models have a shape-preserving algorithm but only FARSIGHT and Pangolin do not expand the initial tracer concentration range.

Test cases

In this paper, we only consider the case of non-divergent and time-dependent winds. Zonal and meridional winds (u and v, respectively) are given by

$$u(\lambda,\theta,t) = \frac{10R}{T}\sin^2(\lambda')\sin(2\theta)\cos\left(\frac{\pi t}{T}\right) + \frac{2\pi R}{T}\cos(\theta),$$
$$v(\lambda,\theta,t) = \frac{10R}{T}\sin(2\lambda')\cos(\theta)\cos\left(\frac{\pi t}{T}\right),$$

where θ is now the latitude and λ the longitude, both in radians. R is the Earth radius, T is the period, set at 12 days here, and $\lambda' = \lambda - 2\pi t/T$. With these winds, the tracer concentration first moves eastwards and is then deformed into filaments up to $t = T/2$. After that, the flux is inverted and the tracer continues to move to the east until it comes back to its initial distribution at $t = T$.

These winds provide an easy way to compute the errors as the solution after a full period can be simply compared with the initial concentration. We will use the same normalized errors as in Lauritzen et al. (2012):

$$\ell_2(q) = \sqrt{\frac{\mathcal{I}((q-q_0)^2)}{\mathcal{I}(q_0^2)}} \quad \text{and} \quad \ell_\infty(q) = \frac{\max\limits_{\forall\lambda,\theta}|q-q_0|}{\max\limits_{\forall\lambda,\theta}|q_0|},$$

where $q = q(\lambda,\theta,t)$ is the tracer concentration and q_0 the initial concentration. Also, $\mathcal{I}$ is defined as the global integral:

$$\mathcal{I}(q) = \frac{1}{4\pi}\int_0^{2\pi}\int_{-\pi/2}^{\pi/2} q(\lambda,\theta,t)\cos\theta \mathrm{d}\lambda\mathrm{d}\theta.$$

For our model, the tracer mixing ratio is approximated as linear functions in a cell. Thus the mean value corresponds to the value at the middle of the cell, so the integral is approximated by $\mathcal{I}(q) = \sum_{i,j}\widehat{q}_{ij}\mathcal{A}_{ij}$, where $\widehat{q}_{ij}$ is the tracer mean value in cell (i,j) and $\mathcal{A}_{ij}$ its area given by Eq. (A.9).

Table A.2: Summary of the models used as a comparison: name, implementation grid, the total number of cells vs. Pangolin, and the time scheme.

Model	Grid	Formal accuracy	Time scheme	Reference
FARSIGHT	Gnomonic cubed sphere	2	Third order	White and Dongarra (2011)
CLAW	Two-patch sphere grid	2	Euler forward	LeVeque (2002)
SLFV-ML	Icosahedral–hexagonal	2	Runge–Kutta 3rd-order TVD	Miura (2007)
CAM-FV	Regular lat–long	2	Euler forward	Collins and Rasch (2004)
UCISOM	Regular lat–long	2	Euler forward	Prather (1986)
Pangolin	Pangolin	2	Euler forward	This paper

Two initial conditions are used here: a sum of two Gaussian hills and a sum of two cosine bells. We note $(\lambda_1, \theta_1) = (5\pi/6, 0)$ and $(\lambda_2, \theta_2) = (7\pi/6, 0)$ as the coordinates of the two "centers" used below. Gaussian hills are defined as

$$q(\lambda, \theta) = h_1(\lambda, \theta) + h_2(\lambda, \theta).$$

Noting $h_{\max} = 0.95$ and $b = 5$, for $i = 1, 2$ we have

$$h_i(\lambda, \theta) = h_{\max} e^{-b((X-X_i)^2+(Y-Y_i)^2+(Z-Z_i)^2)},$$

where X, Y, Z are the Cartesian coordinates of (λ, θ) and X_i, Y_i, Z_i are the Cartesian coordinates of (λ_i, θ_i) for $i = 1, 2$. Cosine bells are defined by

$$q(\lambda, \theta) = \begin{cases} b + c \times h_1(\lambda, \theta) & \text{if} \quad r_1 < r, \\ b + c \times h_2(\lambda, \theta) & \text{if} \quad r_2 < r, \\ b \quad \text{otherwise}, \end{cases}$$

with the background value $b = 0.1$ and amplitude $c = 0.9$. Noting $h_{\max} = 1$ and $r = R/2$, we also have

$$\forall i = 1, 2, \quad h_i(\lambda, \theta) = \frac{h_{\max}}{2} \left(1 + \cos\left(\pi \frac{r_i}{r}\right)\right),$$

where the r_i are the great-circle distances to (λ_i, θ_i) on the sphere:

$$r_i(\lambda, \theta) = R \arccos(\sin\theta_i \sin\theta + \cos\theta_i \cos(\lambda - \lambda_i)).$$

Pangolin results for the Gaussian hills and cosine bells test cases are shown in Fig. 1.7 at $t = 0$, half the period and after a full period. The shape of the tracer distribution is well preserved but numerical diffusion contributes to a decrease in the tracer maxima, as it appears at $t = T/2$ and $t = T$. To compute the numerical order of convergence in Sect. A.2, results at $t = 0$ and $t = T$ will be used, while the preservation of filaments in Sect. A.2 is computed using the results at $t = 0$ and $t = T/2$.

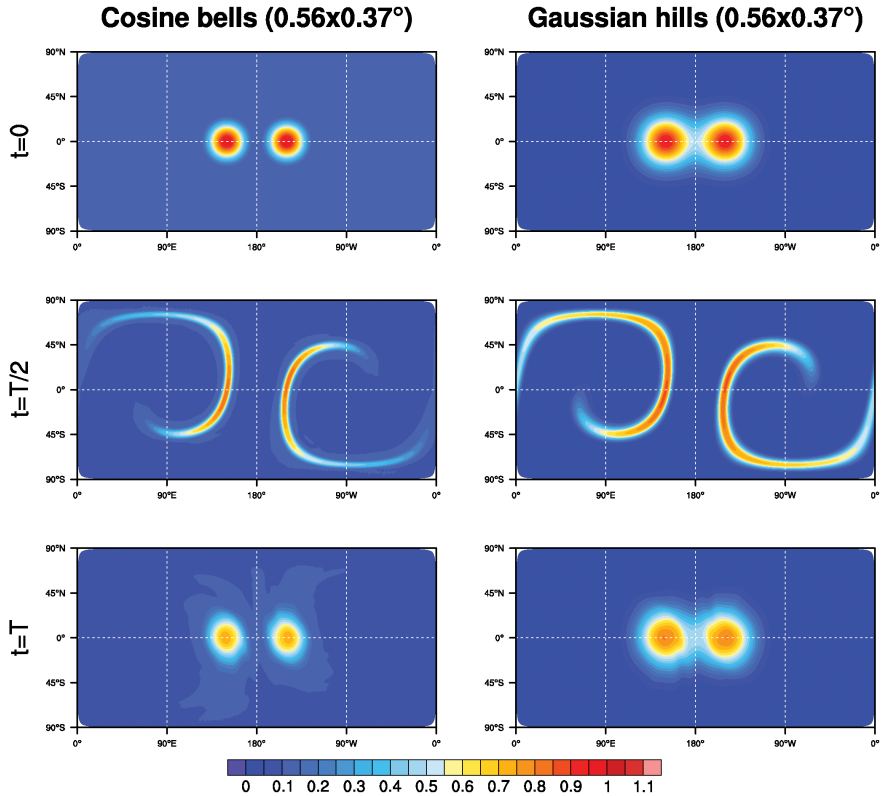


Figure 1.7: Cosine bells and Gaussian hills results for $t = 0$, $t = T/2$, $t = T$ (top to bottom) with Pangolin. The initial distribution is first deformed into filaments and then advected back to its initial position. The Equator resolution is $0.56° \times 0.37°$. For these plots, Pangolin data are interpolated to a regular latitude–longitude grid.

Numerical order of convergence

The aim of this test is to check the rate at which numerical error decreases when resolution increases. Ideally, this rate should be close to the theoretical order of convergence. The Gaussian hills test case is used here, as it provides an infinitely smooth function. Results are plotted in Fig. 1.8, where ℓ_2 and ℓ_∞ are plotted with a varying number of cells. When available, the impact of shape-preserving filters is also represented on the plot, with the exception of UCISOM. For this choice of models, it does not reduce errors in a significant way as can be expected from lower-order models.

For errors at low and medium resolutions, Pangolin is quite close to the other models, with the exception of UCISOM. However, the errors with a large number of cells are lower for models other than Pangolin. One possible explanation is the loss of accuracy due to the interpolation when computing the meridional gradient. In general, the order of convergence of Pangolin is lower. To quantify that, we use numerical optimal order of convergences corresponding to the errors ℓ_2 and ℓ_∞. They result from a least-squares linear regression on the errors plotted vs. the resolution at

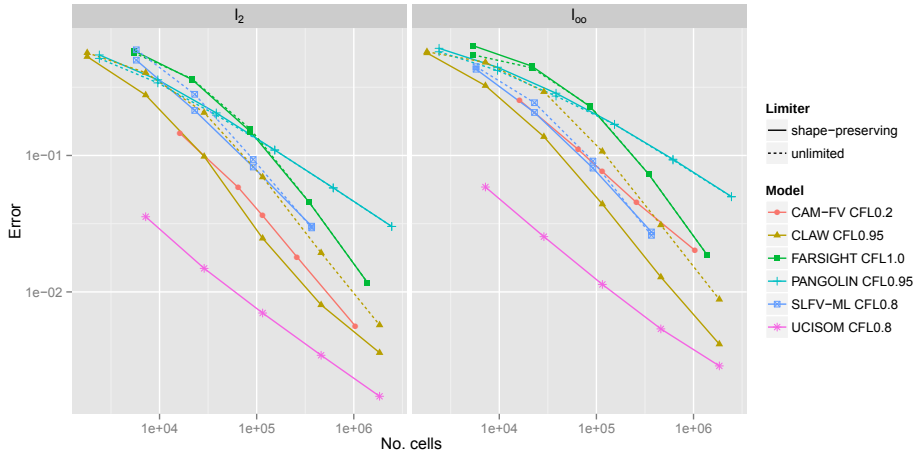


Figure 1.8: Numerical order of convergence for both error measures. These are computed using Gaussian hills after a full rotation. When available, the models are shown with and without the shape-preserving limiters.

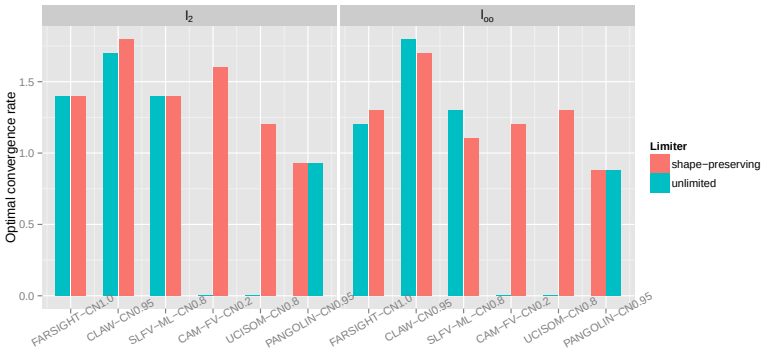


Figure 1.9: Optimal order of convergence computed by a least-squares regression on data from Fig. 1.8. Some models did not offer data without shape-preserving limiters (CAM-FV, UCISOM).

the Equator:

$$\log(\ell_i) = -\kappa_i \log(\Delta\lambda) + b_i \quad \text{with} \quad i = 2, \infty.$$

For Pangolin, the regression is applied after the optimal convergence has been reached. This corresponds to longitudinal resolutions at the Equator in the range $[0.75°, 0.1875°]$. The final numerical convergence rates are shown in Fig. 1.9.

This selection of models and parameters shows that theoretical order is not achieved for all models. For most of them, using a different Courant number does not improve the convergence speedup, with the exception of FARSIGHT and CAM-FV. Using a Courant number of 10.4 (1.2) greatly improves the result of FARSIGHT (CAM-FV) with a Courant number of 1.4 (0.2).

Furthermore, we can see the numerical order of convergence of Pangolin is lower than other models. This is not surprising when comparing with similar finite-volume schemes such as CAM-

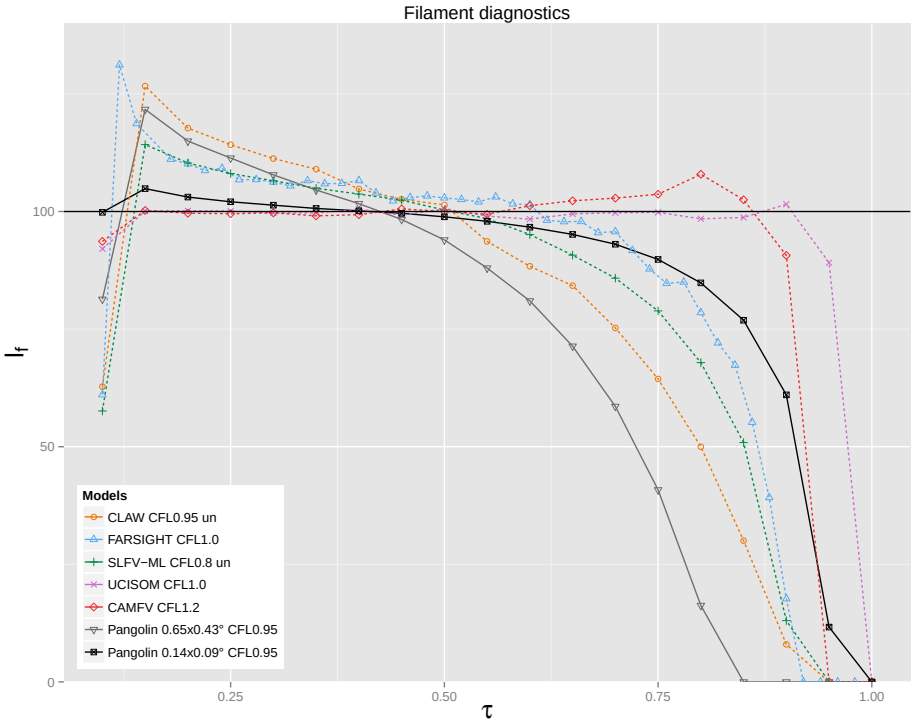


Figure 1.10: Filament diagnostics between for Pangolin (solid line) and other models (dashed line). By default, the shape-preserving version is used. Only CLAW and SLFV-ML use an "unlimited" version.

FV and UCISOM, which use higher-order schemes in one or more directions. We studied this issue further using solid-body rotation test cases (described in Williamson et al. (1992)) and found that the accuracy was limited by the linear interpolation done for the meridional gradient. When the axis of the solid-body rotation matches the polar axis, accuracy is close to second order, whereas the level of accuracy decreases when the axis is in the equatorial plane. In practice, Pangolin needs finer resolutions, as shown in the following test cases, to match the accuracy of other models.

Preservation of filaments

Realistic distributions will most likely be deformed into filaments when the tracer material is stretched and gradients are increased. For some applications, it is important to check how well these filaments are preserved. In the cosine bells test case, the initial concentration is deformed into thin filaments up to $t = T/2$, before being advected to the initial position. Diagnostics are thus computed at $T/2$.

Let us consider the area where the tracer is greater than a given threshold τ. For non-divergent flows and for all thresholds, if this area is not preserved at all times, it suggests filaments are

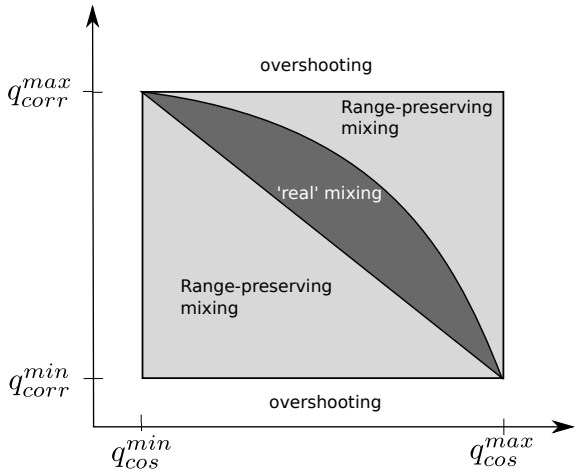


Figure 1.11: Different types of mixing when plotting the correlated concentration against the cosine bells distribution.

degraded. This leads to the definition of the following diagnostic:

$$\ell_{\mathrm{f}} = \begin{cases} 100 \times \dfrac{A(\tau,t)}{A(\tau,0)} & \text{if } A(\tau,0) \neq 0, \\ 0 \quad \text{otherwise,} \end{cases}$$

where $A(\tau,t)$ is the surface area for which the tracer ratio at time t is greater or equal to τ. For Pangolin and other Eulerian schemes, $A(\tau,t)$ is defined as the sum of the cell areas where $q(t) \geq \tau$. Thus the closest ℓ_{f} is to 100, the better the filaments are preserved.

Results are shown in Fig. 1.10. All models except Pangolin are compared with different numbers of cells but with the same resolution of 0.75° at the Equator. Pangolin is shown with two different numbers of cells. The first case (shown as grey solid line) has a resolution of approximately 0.75° at the Equator. The second one (shown as black solid line) is a higher-resolution version (0.1° at the Equator) which approaches the results of more accurate models.

The behavior of the models on non-latitude–longitude grids (all models except UCISOM and CAM-FV) is typical of diffusive schemes. They tend to diffuse the base of the distribution and reduce the maxima, thus leading to an increase in ℓ_{f} for small τ and a decrease in ℓ_{f} for large τ. Another piece of information we can extract from this is the alteration of gradients. Here CAM-FV can be seen to have $\ell_{\mathrm{f}} > 100$ for large τ, which results from gradient steepening. On the other hand, schemes on non-regular grids have a smooth and decreasing profile, showing that the scheme diffusion is also smooth and continuous.

Furthermore, the shape from the diagnostic for Pangolin is quite close to models using non-regular latitude–longitude grids (CLAW, FARSIGHT, SLFV-ML). The behavior is typical of diffusive schemes, where ℓ_{f} is increased for lower threshold values and decreased for higher τ. Pangolin is less accurate than these models, but similar accuracy can be achieved using a finer resolution. Nevertheless, the models using a regular latitude–longitude grid, namely UCISOM and CAM-FV, are the most accurate for this test.

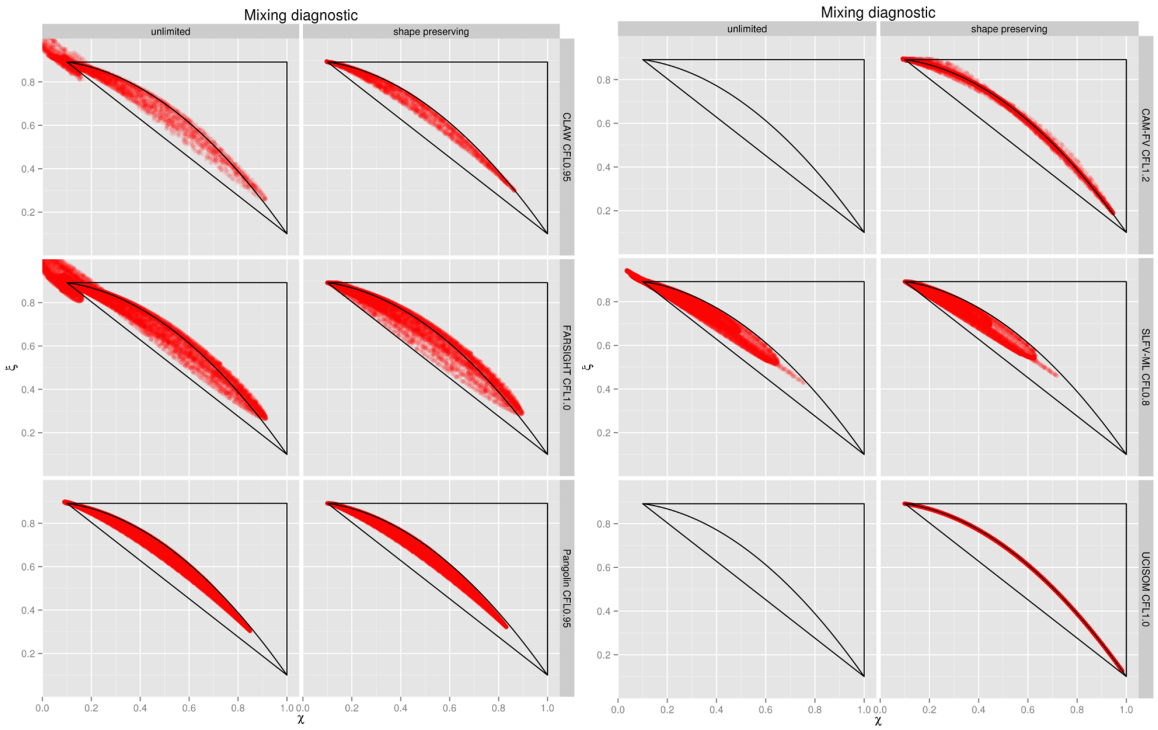


Figure 1.12: Mixing plots for both unlimited and shape-preserving versions. The resolution is set at 0.75° at the Equator, except for Pangolin, which has 0.376°.

Preservation of preexisting relations

When advecting several correlated tracers, numerical transport schemes should preserve the relations between them. However, errors due to numerical diffusion can modify these relations and introduce a "mixing". This is by no means an unphysical feature as real-life tracers can undergo some mixing too, either by chemical reactions or diffusion. This test aims at assessing the amount of unphysical mixing. To that end, we follow Lauritzen et al. (2012) and use a first tracer with the cosine bells initial condition $q_{\cos}(t=0)$ and a second tracer correlated to the first:

$$q_{\text{corr}}(t=0) = -0.8q^2_{\cos}(t=0) + 0.9. \tag{A.12}$$

After a half-period of advection using the non-divergent winds for each case, we plot $q_{\text{corr}}(t = T/2)$ against $q_{\cos}(t = T/2)$ as a scatterplot. Depending on the position of the points, we can then check the mixing level. An illustration of the different zones is given in Fig. 1.11. The shaded convex area in the figure corresponds to "real" mixing as it contains all the lines between two points on the curve corresponding to Eq. (A.12). The light-grey area is not a physical mixing but is still in the initial range. Everything outside the box is overshooting, which may result in unphysical concentrations such as negative values.

Results are shown in Fig. 1.12. All unlimited schemes present some overshooting in the upper

left corner of the figure, which is then removed with a shape-preserving filter. Also, all schemes, with the exception of UCISOM, present real mixing. For this selection of models, only FARSIGHT and CAM-FV have range-preserving unmixing. For FARSIGHT, this can be removed with a larger Courant number. In that case, its accuracy is on the same level as UCISOM. Concerning accuracy, the closer the points lie to the curved defined in Eq. (A.12), the more the scheme preserves the initial relation. In that respect, UCISOM is the most accurate model of this selection, followed by Pangolin and CAM-FV.

Comparing parallel performances

Pangolin was designed with scalability and parallel performances in mind, thus leading to the choice a smaller stencil than the models presented here. From the results presented above, it is clear Pangolin can match the accuracy of other models using finer resolution. Unfortunately, comparing the parallel performances in terms of running time on multi-core architectures is difficult. Some models provide technical details about the performances – see White and Dongarra (2011) for FARSIGHT, Dennis et al. (2011) for CAM-FV or Erath and Nair (2014) and Guba et al. (2014) for some other state-of-the art schemes – but there are not enough data for a thorough comparison. We thus provide as much detail as possible on the parallel performances of Pangolin alone. In particular, Sect. A.3 highlights the smallest size of subdomains needed for a reasonable efficiency for 2-D parallel advection.

A.3 Parallelization

For large-scale numerical simulations, using only sequential computations is no longer affordable. To use a parallel approach, we need to split and balance the computational effort among a set of computing units. For partial different equation-based simulations where the computational cost is evenly shared among the cells, a natural and widely used approach is to partition the computational domain into connex subdomains of similar sizes. Each subdomain is handled by a different computing unit that leads a well-balanced parallel calculation.

The original objectives in designing this model were twofold. First, we intended to have a discretization with cells of equal areas so that the CFL condition is not penalized by the smallest cells. Second, we targeted a semi-structured grid to avoid managing complex data structures, as well as an extra-tool to generate it. This leads us to define the grid detailed in Sect. A.1, where computing the neighbors of grid cells is fully algebraic. In a parallel framework, this grid has an additional asset as it enables a custom algebraic partitioning. Otherwise, mesh splitting often requires sophisticated mesh partitioning tools such as those developed by Karypis and Kumar (1995) and Pellegrini (2012).

Partitioning

In order to perform the partitioning, we first exploit the grid symmetries. The grid is composed of six identical zones as described in Sect. A.1. We then focus on partitioning one of these zones, which contains n_{lat}^2 cells, where n_{lat} is the number of latitudes in the hemisphere. A perfect work balance occurs when there are p^2 subdomains with p dividing n_{lat}. In this case, each subdomain contains exactly $(n_{\text{lat}}/p)^2$ cells. The most natural way to gather cells to form the subdomains is to use the same algorithm as the one to build the grid. The p^2 subdomains are set on p bands,

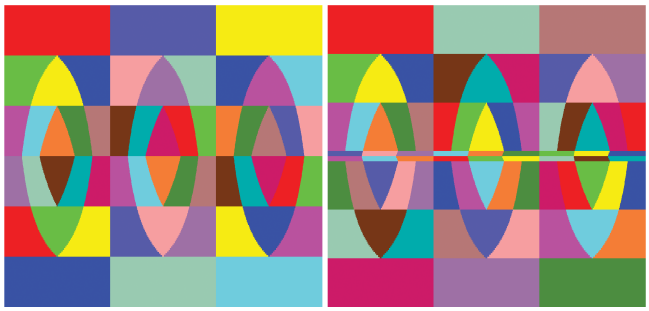


Figure 1.13: Algebraic mesh partitioning for an optimal case (54 domains, left) and sub-optimal case (74 partitions, left). Each color corresponds to a subdomain (the same color can be used for different subdomains). The grid contains 48 600 cells and is shown in latitude–longitude coordinates.

where the kth band contains $2k-1$ subdomains ($k>0$). Applying the same decomposition to the remaining five zones leads to the structure shown in Fig. 1.13.

For the sake of flexibility, this optimal situation can be slightly degenerated to accommodate any number of subdomains. For these sub-optimal situations, we consider the closest lower square p'^2 on a zone, with p' dividing n_{lat}. These p'^2 subdomains are set according to the previous strategy. The remaining $p^2-p'^2$ cores are associated with on a special band, with less cells and thus without preserving the partition size.

These results are given for one zone only. For the complete grid, we find the optimal number of subdomains is $6p^2$ with p dividing n_{lat}. Otherwise, the model can manage any number of subdomains on one-third of the grid (between longitude 0 and 120). The total number of subdomains in these optimal cases is then a multiple of 3. It is worth noting that White and Dongarra (2011) need a condition similar to our perfect case for their parallel version: the number of subdomains must be of the form $6p^2$, with p dividing the number of cells on a cube edge.

This algebraic partitioning uses the regular topology of the grid to create subdomains with a regular shape. This feature ensures regular data access and allows for possible optimizations by anticipation strategies such as pre-fetching to improve faster data access by loading data into the cache before it is actually needed. To highlight this regularity, and for the sake of comparison, we display in Fig. 1.14 the partition computed by the mesh partitioner Scotch (Pellegrini, 2012) for the same grid as in Fig. 1.13. One can observe that this general-purpose tool does not succeed in preserving the regular shape of the subdomains. In addition, our partitioning reduces the number of neighbors for the subdomains and consequently the number of messages exchanged. The total number of neighbors of our partitioning is divided by at least 2 when compared with Scotch, even in sub-optimal cases.

Parallel implementation

Our parallel implementation first targets distributed memory architectures. Therefore, we consider a message passing parallel implementation on top of the MPI library where each subdomain is assigned to a different computing unit. To update the tracer ratio for all the cells in a subdomain, most of the information required to compute the fluxes is already available in the subdomain. However, some communications need to be performed to exchange information along the interfaces

Figure 1.14: Mesh partitioning computed by the general purpose mesh partitioner Scotch for 54 domains (optimal case for Pangolin, left) and 72 domains (sub-optimal case for Pangolin, right). Each color corresponds to a subdomain (the same color can be used for different partitions). The grid contains 48 600 cells and is shown in latitude–longitude coordinates.

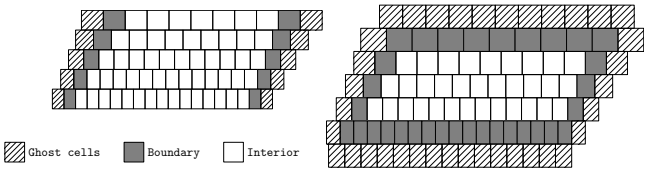


Figure 1.15: Ghost, boundary and interior cells for zonal (left) and meridional (right) advection.

```
1: subroutine meridional_advection
2:     Starts ratio exchange for ghost cells
3:     Compute zonal gradient on the interior
4:     Wait for end of communications
5:     Compute zonal gradient on the boundary
6:
7:     Starts zonal gradient exchange for ghost cells
8:     Compute meridional gradient on the interior
9:     Wait for end of communications
10:    Compute meridional gradient on the boundary
11:
12:    Starts meridional gradient exchange for ghost cells
13:    Compute meridional fluxes on the interior
14:    Wait for end of communications
15:    Compute meridional fluxes on the boundary
16:
17:    Update all ratios
18: end subroutine
```

```
1: subroutine zonal_advection
2:     Starts ratio exchange for ghost cells
3:     Compute zonal gradient on the interior
4:     Wait for end of communications
5:     Compute zonal gradient on the boundary
6:
7:     Starts zonal gradient exchange for ghost cells
8:     Compute zonal fluxes on the interior
9:     Wait for end of communications
10:    Compute zonal fluxes on the boundary
11:
12:    Update all ratios
13: end subroutine
```

Figure 1.16: Main steps of the algorithm for zonal (left) and meridional (right) advection.

generated by the partitioning. For example, zonal fluxes need to exchange concentration and gradient data with the east and west neighboring subdomains. In that respect, we introduce ghost cells which store data received from the neighbors via message exchanges. It should be noted that, due to the shape of the subdomains, meridional advection requires communications with the north, south, east and west neighbors. This is illustrated in Fig. 1.15, where the ghost cells are shown as hatched cells.

Table A.3: Configuration for one of the 158 Sandy Bridge nodes.

– 2 Intel Sandy Bridge 8-core CPUs (peak performance of 330 GFLOPS s^{-1})
– 32 GB of memory
– 31 KB L1 cache, 256 KB L2 cache for each core
– 20 MB L3 cache, shared by the 8 cores of each CPU

In order to improve the parallel performance of the code and hide the communication time as much as possible, non-blocking communications are used. This avoids waiting for the completion of communications and allows for computations to be optimized. In practice, we first post the communication requests, and then perform the calculation on the interior cells, which do not need data outside the current subdomain. Then we finalize data reception and eventually compute the new quantities on the boundary cells using the values received in the ghost cells. This approach is a rather classical implementation to overlap computation and communication in large-scale simulations. The specificity of Pangolin comes rather from the algebraic features of the decomposition, so the neighbors of a subdomain and the cells inside it are computed on the fly. The model was also designed to have only one layer of ghost-cells as to decrease the communication volume.

Each advection step can be decomposed into several tasks: gradient computing, fluxes computing and mass update. The first two tasks require some communication and the non-blocking approach is used for each one of them. It should be noted that meridional advection requires an extra communication as it needs a zonal gradient to perform an interpolation (see Sect. A.1). The different tasks are shown in detail in the algorithms shown in Fig. 1.16. The combination of boundary and interior cells for all these tasks is shown in Fig. 1.15.

The boundary zone is much larger for meridional advection as data need to be exchanged with the zone's four neighbors. More precisely, computation of the zonal gradient requires communication with the east and west neighbors of the subdomain. Furthermore, computing the meridional gradient and fluxes requires access to the north and south neighbors. Due to the semi-structured layout of the grid, this results in the "stair-like" structure for boundary cells in the meridional advection.

Most of the computation is performed during meridional advection. First, due to the extra step of computing the zonal gradient, meridional advection requires three message exchanges (vs. two for zonal advection). Also, the number of boundary cells is larger, thus increasing the communication volume. Finally, computation in a cell is more expensive as the number of neighbors is four on average (vs. two in the zonal case).

Performances

In this section, we investigate the parallel scalability of our implementation of the numerical scheme. Tests were done on the Bull cluster at CERFACS, whose features are shown in Table A.3. We consider a strong speedup study where the size of the global grid is fixed and the number of computing cores is increased. Ideally, the parallel elapsed time should be reduced proportionally to the number of cores selected for the parallel simulation. For our experimental study, we consider

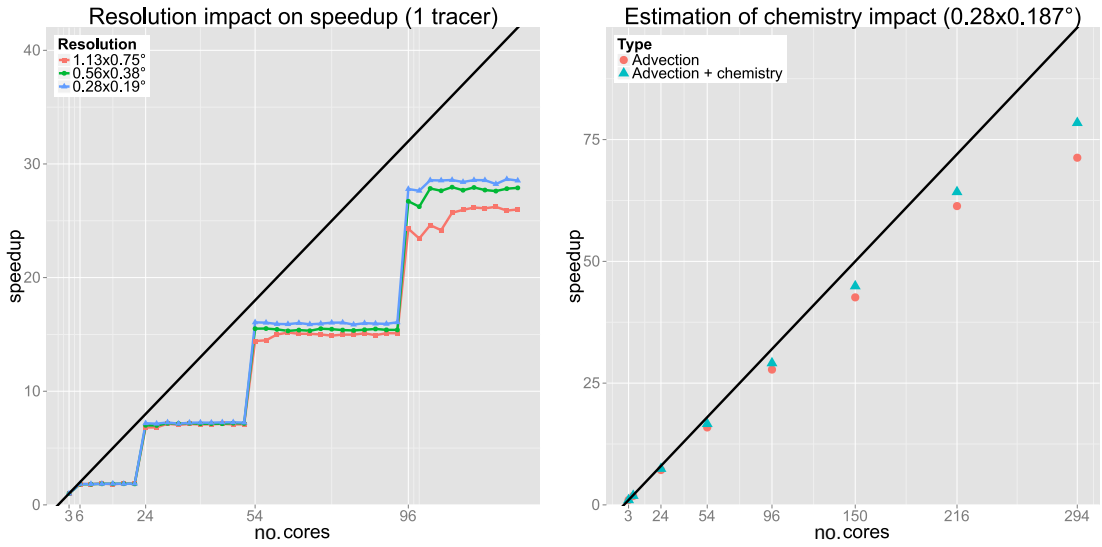


Figure 1.17: Speedup up to 126 cores (left) and 294 cores (right). The left plot shows the impact of grid resolution for smaller configurations. The right plot shows both the advection performances and an estimation of the chemistry impact on scalability. Resolutions used are $1.125° \times 0.75°$, $0.56° \times 0.376°$ and $0.28° \times 0.188°$. Both figures use non-divergent winds from Sect. A.2 over a full period with a CFL of 0.96.

the elapsed time on three cores as the reference time so that the speedup is defined by

$$S(p) = \frac{T(3)}{T(p)},$$

where $T(p)$ is the parallel elapsed time observed when performing the calculation on p cores.

In strong speedup studies, the number of cells per subdomain decreases when the number of subdomains increases. Even though we attempt to overlap the communication and the calculation, communication volume tends to grow when the number of subdomains increases. A direct consequence is that the observed speedups gradually depart from the ideal speedup when the number of cores increases, as can be observed in Fig. 1.17 (left-hand side). On the right plot, we report experiments for the 2-D advection scheme using various grid sizes ranging from $1.13° \times 0.75°$ to $0.28° \times 0.19°$ when the number of cores varies from 3 to 128. As expected, the larger the grid, the better the parallel performance, since we can better overlap calculation and computation. The speedup curves exhibit steps, with significant gaps when an optimal number of cores (6, 24, 54, 96) is used. In between, using more cores does not translate into an improvement in performance as the workload of the largest subdomain is not reduced.

The final version of Pangolin will be combined with chemistry modeling. As the chemistry computation is fully local, we can estimate the performances of the chemistry-advection model. Figure 1.17 (right-hand side) shows the estimated speedups of the complete chemistry advection simulation on the finest grid (i.e., $0.28° \times 0.19°$). We assumed the chemistry cost was constant across all cells and the chemical time step was similar to the advection time step. These assump-

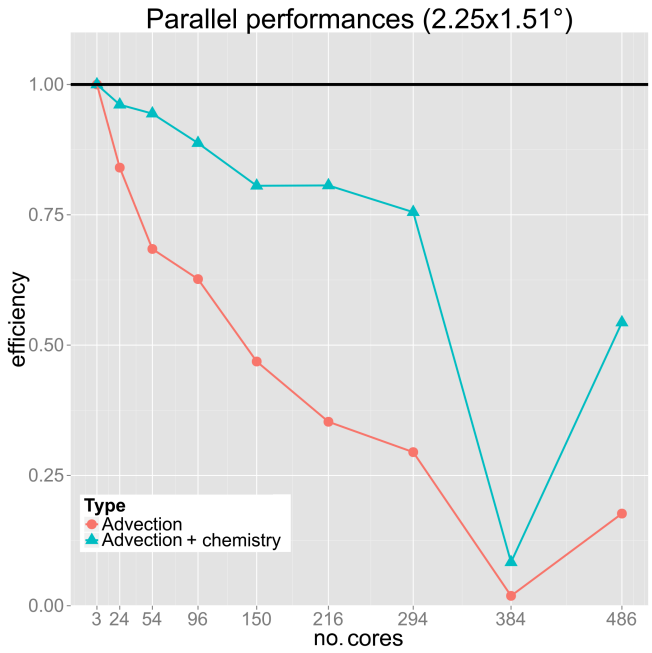


Figure 1.18: Efficiency up to 486 cores. A resolution of $2.25 \times 1.51°$ was used to examine the limit of the parallelization performances for 2-D advection. For this test, the Airain cluster was used, which has 9504 nodes where each node has an Intel Xeon processor (16 cores, 2.7 GHz and 4 GB of shared memory).

tions are not valid in practice and only give an upper bound on the speedup. Nevertheless they give some insight into the final performances. The chemical time step was obtained from a new solver developed by D. Cariolle (personal communication, 2014) called ASIS with 90 species. As a reference, we use its implementation by P. Moinat, using the GMRES method (personal communication, 2014). As a result, adding the chemistry greatly increases the computational load in a subdomain and thus improves the scalability. On the other hand, communication volumes only increase linearly as a function of the number of tracers, as expected.

Comparing performances between parallel models is not an easy task. A meaningful comparison would require all of the models to be compiled and run on the same cluster as hardware and software performances are paramount in such studies. Such tests are not within the scope of the current paper. However, we examine the limits of our parallelization strategy, an additional strong scaling test was run with a rather coarse resolution ($2.25 \times 1.75°$): the number of cores was increased until the subdomains became extremely small. At that point, the computational load inside the subdomains is not enough to cover the communication costs. These results are shown in Fig. 1.18, where the efficiency is plotted against the number of cores. Here, the efficiency is defined as

$$E(p) = \frac{T(3)}{pT(p)},$$

so ideal performances should be close to 1. We can consider the parallel performances "break

Table A.4: Estimation of the number of cores needed for an efficiency of 0.75 for 2-D advection at several resolutions (lat × long).

Resolution	No. cores
1.5×1.0	150
0.75×0.5	486
0.15×0.1	6144

Table A.5: Some Pangolin configurations, with the number of latitudes on a hemisphere n_{lat}, total number of cells n_{pangolin} and resolution at the Equator in degrees.

n_{lat}	n_{pangolin}	$\Delta\phi \times \Delta\lambda$
20	2400	4.5×3.08
90	48 600	1.0×0.67
320	614 400	0.28×0.188

down" at 294 cores. The size of the subdomains is then 5×5, a non-realistic configuration where the subdomains are so small that the communication cost can no longer be hidden. From this test, we have estimated the size of subdomains needed for an efficiency of 0.75 was 18×18. In practice, this allowed us to estimate the number of cores needed for the same efficiency at different resolutions. Results are shown in Table A.4.

A.4 Conclusions

In this paper, we have presented a parallel scalable algorithm for 2-D-advection on the sphere. We focused on enabling the model to be as parallel as possible. Pangolin uses a reduced latitude–longitude grid, which overcomes the pole issue, and a finite-volume formulation that ensures local conservation of the tracer mass. Grid features were carefully exploited to minimize memory requirements on the one hand, and provide maximal efficiency on parallel architectures on the other. The accuracy of the scheme was also chosen as to minimize the impact on message passing. It was found that the approximations made for computing the meridional gradients near the poles limits the accuracy of the model. Therefore, to reach the accuracy of other second-order models, resolution must be increased. This can be easily achieved without large computation penalty due to the good scalability of Pangolin.

We expect further improvement in terms of parallelism when chemistry is added. An ongoing work addresses real-case atmospheric situations using a linear scheme (Cariolle and Teyssèdre (2007)), which is used to model the evolution of stratospheric ozone on an isentropic surface. In future versions, vertical advection will be added, requiring a more advanced correction of the winds for mass preservation. A complex chemistry will also be added using the ASIS solver and the RACMOBUS scheme (Dufour et al. (2005)). This chemistry will most likely perturb the load balancing. One mitigation strategy would be to use multi-threading in the subdomains. To conclude, Pangolin is a practical model that is aimed at taking advantage of present and future parallel architectures for large-scale atmospheric transport.

Code availability

The code is copyright of the CERFACS laboratory. The documentation is available as a user manual and as code documentation at `http://cerfacs.fr/~praga/pangolin/index.html`. To request access to either the source code or documentation, please email A. Praga (alexis.praga@gmail.com)

or D. Cariolle (cariolle@cerfacs.fr). The data and scripts for the plots of this paper are also available as a supplement.

Acknowledgements

Financial support from the DGAC (Direction Générale de l'Aviation Civile) through the project IMPACT is gratefully acknowledged. A. Praga was supported by a PhD grant from Météo-France.

Edited by: A. Colette

Appendix **B**

Supplement

B.1 Proof: number of cells at the poles

Here, we demonstrate why the number of cells at the poles for the Pangolin grid can be approximated by 3. The latitude spacing is supposed to be "small" enough and the cells are supposed to be square at the Equator. We first consider the surface of the band near the Equator, defined by $\pi/2 - \Delta\phi \leq \phi \leq \pi/2$:

$$S_{eq} = \int_0^{2\pi} \int_{\pi/2}^{\pi/2-\Delta\phi} r^2 \sin\phi \mathrm{d}\lambda \mathrm{d}\phi = 2\pi r^2 \sin\Delta\phi$$

The surface of the band near the North Pole ($0 \leq \phi \leq \Delta\phi$) is:

$$S_{pole} = \int_0^{2\pi} \int_0^{\Delta\phi} r^2 \sin\phi \mathrm{d}\lambda \mathrm{d}\phi = 2\pi r^2 (1 - \cos\Delta\phi)$$

The number of cells at the Equator is $2\pi/\Delta\lambda = 2\pi/\Delta\phi$ so the area of the cells at the Equator is:

$$\mathcal{A}_{eq} = \frac{S_{eq}}{\frac{2\pi}{\Delta\phi}} = r^2 \Delta\phi \sin\Delta\phi$$

The grid preserves the areas so the number of cells at the pole is:

$$n_1 = \left\lfloor \frac{S_{pole}}{\mathcal{A}_{eq}} \right\rfloor = \left\lfloor \frac{2\pi(1 - \cos\Delta\phi)}{\Delta\phi \sin\Delta\phi} \right\rfloor$$

Using small angles approximations, it gives:

$$n_1 = \left\lfloor \frac{2\pi(\frac{\Delta\phi^2}{2})}{\Delta\phi^2} \right\rfloor = \lfloor \pi \rfloor = 3$$

B.2 MPI send modes

The following is taken from http://www.mcs.anl.gov/research/projects/mpi/sendmode.html:

`MPI_Send`
: will not return until you can use the send buffer. It may or may not block (it is allowed to buffer, either on the sender or receiver side, or to wait for the matching receive). May buffer; returns immediately and you can use the send buffer. A late add-on to the MPI specification. Should be used only when absolutely necessary.

`MPI_Ssend`
: will not return until matching receive posted.

`MPI_Rsend`
: may be used **only** if matching receive already posted. User responsible for writing a correct program.

`MPI_Isend`
: Nonblocking send. But not necessarily asynchronous. You can **not** reuse the send buffer until either a successful, wait/test or you **know** that the message has been received (see `MPI_Request_free`). Note also that while the I refers to immediate, there is no performance requirement on `MPI_Isend`. An immediate send must return to the user without requiring a matching receive at the destination. An implementation is free to send the data to the destination before returning, as long as the send call does not block waiting for a matching receive. Different strategies of when to send the data offer different performance advantages and disadvantages that will depend on the application.

`MPI_Ibsend`
: buffered nonblocking.

`MPI_Issend`
: Synchronous nonblocking. Note that a Wait/Test will complete only when the matching receive is posted.

`MPI_Irsend`
: As with MPI_Rsend, but nonblocking.

Appendix C

Design

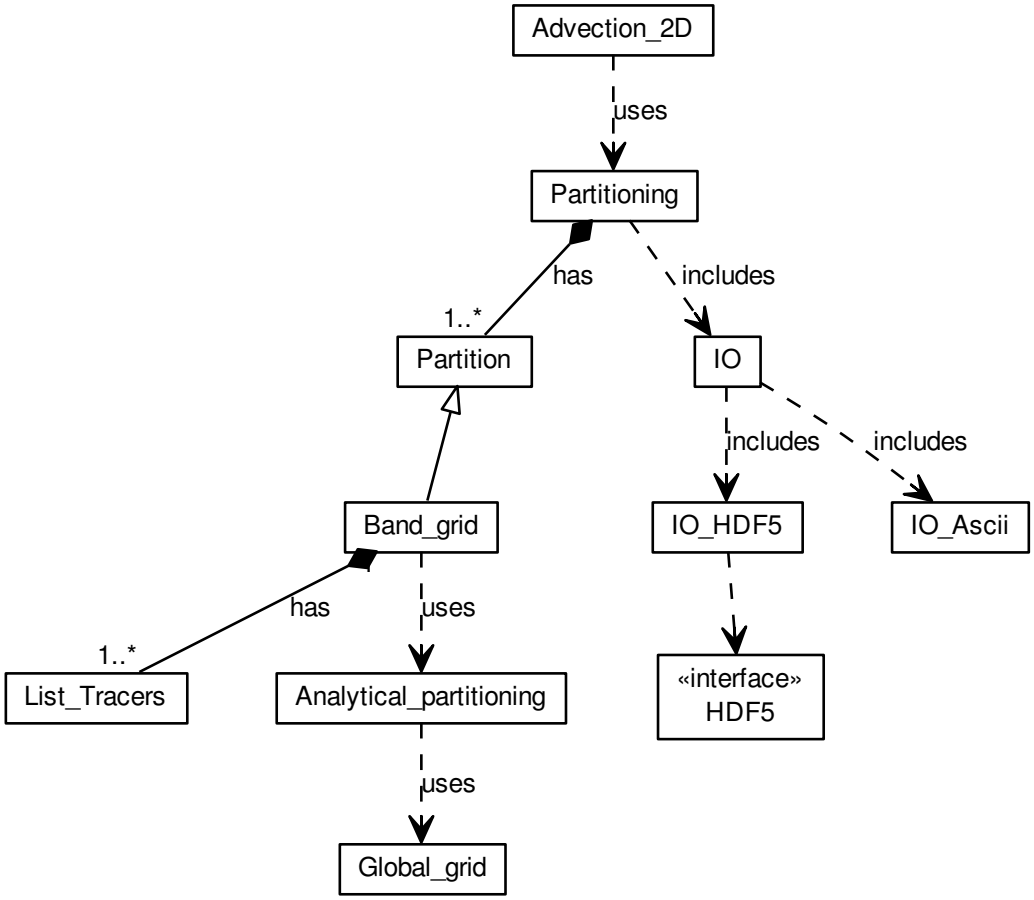


Figure 3.1: Classes diagram for Pangolin, with the main classes.

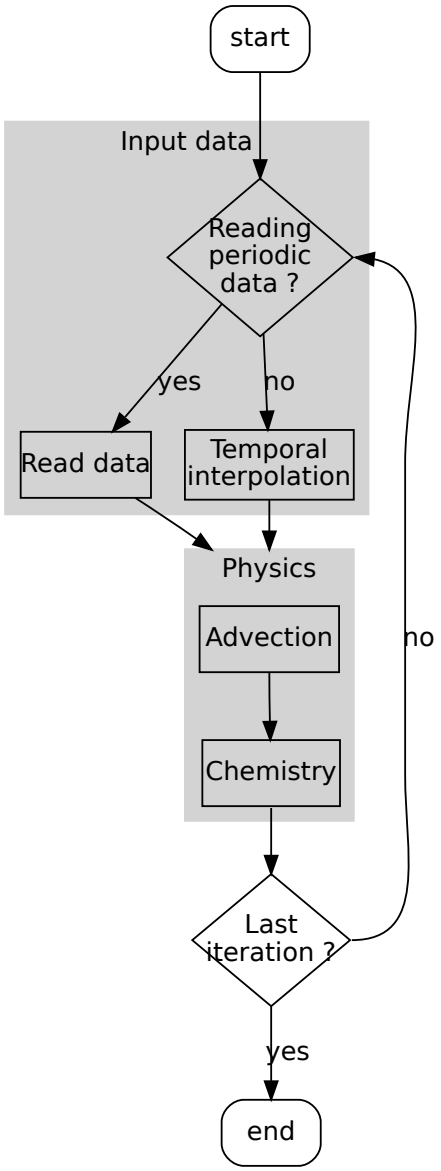


Figure 3.2: Flowchart representing a full run in Pangolin. Input winds are supposed to be divergence free: there were corrected either during their generation (analytical winds) or during interpolation (real winds). As temporal interpolation is linear, interpolated winds do not need to be corrected either

Bibliography

Akio Arakawa and Vivian R Lamb. Computational design of the basic dynamical processes of the UCLA general circulation model. *Methods in computational physics*, 17:173–265, 1977.

D. Belikov, S. Maksyutov, T. Miyasaka, T. Saeki, R. Zhuravlev, and B. Kiryushov. Mass-conserving tracer transport modelling on a reduced latitude-longitude grid with NIES-TM. *Geoscientific Model Development*, 4(1):207–222, March 2011. ISSN 1991-9603. doi: 10.5194/gmd-4-207-2011.

G. L. Browning. A comparison of three numerical methods for solving differential equations on the sphere. *Monthly Weather Review*, 1989. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1989)117<1058:ACOTNM>2.0.CO;2.

Radmila Bubnová, Gwenaëlle Hello, Pierre Bénard, and Jean-François Geleyn. Integration of the Fully Elastic Equations Cast in the Hydrostatic Pressure Terrain-Following Coordinate in the Framework of the ARPEGE/Aladin NWP System. *Monthly Weather Review*, 123(2):515–535, February 1995. ISSN 0027-0644. doi: 10.1175/1520-0493(1995)123⟨0515:IOTFEE⟩2.0.CO;2. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1995)123<0515:IOTFEE>2.0.CO;2.

R. Buizza, D. S. Richardson, and T. N. Palmer. The new 80-km High-Resolution ECMWF EPS. *Citeseer*, 2008(90):2–9, 2008. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+new+80-km+High-Resolution+ECMWF+EPS#0.

D. Cariolle and D. Brard. The distribution of ozone and active stratospheric species: Results of a two-dimensional atmospheric model. In *Atmospheric Ozone*, 1985. URL http://link.springer.com/chapter/10.1007/978-94-009-5313-0_16.

D. Cariolle and H. Teyssèdre. A revised linear ozone photochemistry parameterization for use in transport and general circulation models: multi-annual simulations. *Atmospheric Chemistry and Physics*, 7(2006):2183–2196, 2007. ISSN 16807316. doi: doi:10.5194/acp-7-2183-2007.

W. W. Carlson, J. M. Draper, D. E. Culler, and Kathy Yelick. *Introduction to UPC and language specification*, volume 20715. Center for Computing Sciences, Institute for Defense Analyses, 1999. URL http://www3.uji.es/~aliaga/UPC/upc_intro.pdf.

M. L. Cathala, J. Pailleux, and V. H. Peuch. Improving global chemical simulations of the upper troposphere–lower stratosphere with sequential assimilation of MOZAIC data. *Tellus B*, 2003. URL http://onlinelibrary.wiley.com/doi/10.1034/j.1600-0889.2003.00002.x/full.

DeHui Chen, JiShan Xue, XueSheng Yang, HongLiang Zhang, XueShun Shen, JiangLin Hu, Yu Wang, LiRen Ji, and JiaBin Chen. New generation of multi-scale NWP system (GRAPES): general scientific design. *Chinese Science Bulletin*, 53(22):3433–3445, November 2008. ISSN 1001-6538. doi: 10.1007/s11434-008-0494-z. URL http://link.springer.com/10.1007/s11434-008-0494-z.

M. P. Chipperfield. New version of the TOMCAT/SLIMCAT off-line chemical transport model: Intercomparison of stratospheric tracer experiments. *Quarterly Journal of the Royal Meteorological Society*, 132(617):1179–1203, April 2006. ISSN 00359009. doi: 10.1256/qj.05.51.

Martin J. Chorley and David W. Walker. Performance analysis of a hybrid MPI/OpenMP application on multi-core clusters. *Journal of Computational Science*, 1(3):168–174, August 2010. ISSN 18777503. doi: 10.1016/j.jocs.2010.05.001. URL http://linkinghub.elsevier.com/retrieve/pii/S1877750310000396.

Phillip Colella and Paul R. Woodward. The Piecewise Parabolic Method (PPM). *Journal of Computational Physics*, 201:174–201, 1984.

WD Collins and Philip J. Rasch. Description of the NCAR community atmosphere model (CAM 3.0). *NCAR Technical Note*, 2004.

Jean Côté, Sylvie Gravel, and André Méthot. The operational CMC-MRB global environmental multiscale (GEM) model. Part I: Design considerations and formulation. *Monthly Weather Review*, 126(6):1373–1395, June 1998. ISSN 0027-0644. doi: 10.1175/1520-0493(1998)126⟨1373:TOCMGE⟩2.0.CO;2. URL http://journals.ametsoc.org/doi/full/10.1175/1520-0493(1998)126<1373:TOCMGE>2.0.CO;2http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1998)126<1373:TOCMGE>2.0.CO;2.

T. Davies, M. J. P. Cullen, A. J. Malcolm, M. H. Mawson, A. Staniforth, A. A. White, and N. Wood. A new dynamical core for the Met Office's global and regional modelling of the atmosphere. *Quarterly Journal of the Royal Meteorological Society*, 131(608):1759–1782, April 2005. ISSN 00359009. doi: 10.1256/qj.04.101. URL http://doi.wiley.com/10.1256/qj.04.101.

Robert Delmas, Gérard Mégie, and Vincent-Henri Peuch. *Physique et chimie de l'atmosphère*. Belin, 2005.

J. M. Dennis, J Edwards, KJ J. Evans, O. Guba, P. H. Lauritzen, A. A. Mirin, A. St-Cyr, M. A. Taylor, and P. H. Worley. CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model. *International Journal of High Performance Computing Applications*, 26:74–89, 2011. ISSN 1094-3420. doi: 10.1177/1094342011428142. URL http://hpc.sagepub.com/content/early/2011/11/12/1094342011428142.abstract.

M. F. Dixon and Kenneth Tan. Distributed solution of high-order compact difference schemes for multidimensional convection-diffusion equations. *Computational Science and Its Applications – ICCSA*, pages 226–235, 2003. URL http://link.springer.com/chapter/10.1007/3-540-44842-X_24.

A. Dufour, M. Amodei, G. Ancellet, and V.-H. Peuch. Observed and modelled "chemical weather" during ESCOMPTE. *Atmospheric Research*, 74(1-4):161–189, March 2005. ISSN 01698095. doi: 10.1016/j.atmosres.2004.04.013. URL http://linkinghub.elsevier.com/retrieve/pii/S0169809504001498.

J. K. Dukowicz and J. R. Baumgardner. Incremental remapping as a transport/advection algorithm. *Journal of Computational Physics*, 335:318–335, 2000. doi: 10.1006/jcph.2000.6465. URL http://www.sciencedirect.com/science/article/pii/S0021999100964659.

ECMWF. The ECMWF Model Grids, 2004. URL https://badc.nerc.ac.uk/data/ecmwf-op/grids.html#n80_reduced.

E. Emili, B. Barret, S. Massart, E. Le Flochmoen, A. Piacentini, L. El Amraoui, O. Pannekoucke, and D. Cariolle. Combined assimilation of IASI and MLS observations to constrain tropospheric and stratospheric ozone in a global chemical transport model. *Atmospheric Chemistry and Physics*, 14(1):177–198, January 2014. ISSN 1680-7324. doi: 10.5194/acp-14-177-2014. URL http://www.atmos-chem-phys.net/14/177/2014/.

Christoph Erath and Ramachandran D. Nair. A conservative multi-tracer transport scheme for spectral-element spherical grids. *Journal of Computational Physics*, 271:244–260, August 2014. ISSN 00219991. doi: 10.1016/j.jcp.2014.04.008. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999114002629.

Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. An overview of the HDF5 technology suite and its applications. *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases - AD '11*, pages 36–47, 2011. doi: 10.1145/1966895.1966900. URL http://portal.acm.org/citation.cfm?doid=1966895.1966900.

I. T. Foster and P. H. Worley. Parallel algorithms for the spectral transform method. *SIAM Journal on Scientific Computing*, 18(3):806–837, 1997. URL http://epubs.siam.org/doi/abs/10.1137/S1064827594266891.

Francis X. Giraldo. Lagrange–Galerkin Methods on Spherical Geodesic Grids: The Shallow Water Equations. *Journal of Computational Physics*, 160(1):336–368, May 2000. ISSN 00219991. doi: 10.1006/jcph.2000.6469. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999100964696.

Francis X. Giraldo and T. E. Rosmond. A scalable spectral element Eulerian atmospheric model (SEE-AM) for NWP: dynamical core tests. Technical report, DTIC Document, 2003. URL http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA515908.

S. K. Godunov, A. V. Zabrodin, and G. P. Prokopov. A computational scheme for two-dimensional nonstationary problems of gas dynamics and calculation of the flow from a shock wave approaching a stationary state. *Zhurnal Vychislitel'noi ...*, 1(6):1020—-1050, 1961.

William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface (2nd edition)*. MIT Press, 1999. URL http://books.google.com/books?hl=en&lr=&id=xpBZ0RyRb-oC&oi=fnd&pg=PA1&dq=Using+MPI:+portable+parallel+programming+with+the+message-passing+interface&ots=ubato1OM8_&sig=AFwehbZi4M49OEnjob3WxIjWuEY.

Oksana Guba, Mark Taylor, and Amik St-Cyr. Optimization-based limiters for the spectral element method. *Journal of Computational Physics*, 267:176–195, June 2014. ISSN 00219991. doi: 10.1016/j.jcp.2014.02.029. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999114001491.

Xiaohu Guo, Michael Lange, Gerard Gorman, Lawrence Mitchell, and Michèle Weiland. Developing a scalable hybrid MPI/OpenMP unstructured finite element model. *Computers & Fluids*, September 2014. ISSN 00457930. doi: 10.1016/j.compfluid.2014.09.007. URL http://linkinghub.elsevier.com/retrieve/pii/S0045793014003442.

James J. Hack and R Jakob. *Description of a global shallow water model based on the spectral transform method*. National Center for Atmospheric Research, 1992. URL http://nldr.library.ucar.edu/repository/assets/technotes/TECH-NOTE-000-000-000-154.pdf.

Lucas M. Harris, Peter Hjort Lauritzen, and Rashmi Mittal. A flux-form version of the conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed sphere grid. *Journal of Computational Physics*, 230(4):1215–1237, February 2011. ISSN 00219991. doi: 10.1016/j.jcp.2010.11.001. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999110006005.

E. Hesstvedt, Ö. Hov, and I.S.A. Isaksen. Quasi-steady-state approximations in air pollution modeling: Comparison of two numerical schemes for oxidant prediction. *International Journal of . . .*, 1978. URL http://onlinelibrary.wiley.com/doi/10.1002/kin.550100907/abstract.

M. Hortal and A. J. Simmons. Use of reduced Gaussian grids in spectral models. *Monthly Weather Review*, 1991. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1991)119%3C1057:UORGGI%3E2.0.CO%3B2.

Frédéric Hourdin. Représentation du transport direct et inverse dans les modèles globaux de climat et étude des couplages entre composition et dynamique atmosphérique sur Titan. *Mémoire présenté pour obtenir une Habilitation à diriger des Recherches*, 2005.

Frédéric Hourdin and A. Armengaud. The use of finite-volume methods for atmospheric advection of trace species. Part I: Test of various formulations in a general circulation model. *Monthly weather review*, 127(1959):822–837, 1999.

V. Huijnen, J. Williams, M. van Weele, T. van Noije, M. Krol, F. Dentener, A. Segers, S. Houweling, W. Peters, J. de Laat, F. Boersma, P. Bergamaschi, P. van Velthoven, P. Le Sager, H. Eskes, F. Alkemade, R. Scheele, P. Nédélec, and H.-W. Pätz. The global chemistry transport model TM5: description and evaluation of the tropospheric chemistry version 3.0. *Geoscientific Model Development*, 3(2):445–473, October 2010. ISSN 1991-9603. doi: 10.5194/gmd-3-445-2010.

IGAC. IGAC Science Plan and Implementation Strategy. Technical report, IGAC, 2006.

G. S. Jiang and C. Wu. A high-order WENO finite difference scheme for the equations of ideal magnetohydrodynamics. *Journal of Computational Physics*, 1999. URL http://www.sciencedirect.com/science/article/pii/S0021999199962071.

Haoqiang Jin, Dennis Jespersen, Piyush Mehrotra, Rupak Biswas, Lei Huang, and Barbara Chapman. High performance computing using MPI and OpenMP on multi-core parallel systems. *Parallel Computing*, 37(9):562–575, September 2011. ISSN 01678191. doi: 10.1016/j.parco.2011.02.002. URL http://linkinghub.elsevier.com/retrieve/pii/S0167819111000159.

JMA. Outline of the operational numerical weather prediction of the Japan Meteorological Agency. Technical report, JMA, 2013. URL http://www.jma.go.jp/jma/jma-eng/jma-center/nwp/outline2013-nwp/pdf/outline2013_03.pdf.

P. Jöckel, R. von Kuhlmann, M. G. Lawrence, B. Steil, Carl A. M. Brenninkmeijer, Paul J. Crutzen, Philip J. Rasch, and Brian Eaton. On a fundamental problem in implementing flux-form advection schemes for tracer transport in 3-dimensional general circulation and chemistry transport models. *Quarterly Journal of the Royal Meteorological Society*, 127(573):1035–1052, April 2001. ISSN 00359009. doi: 10.1002/qj.49712757318. URL http://doi.wiley.com/10.1002/qj.49712757318.

P. W. Jones. A user's guide for SCRIP: A spherical coordinate remapping and interpolation package. *Los Alamos National Laboratory*, 1997. URL http://oceans11.lanl.gov/trac/SCRIP/export/21/trunk/SCRIP/doc/SCRIPusers.pdf.

P. W. Jones. First-and second-order conservative remapping schemes for grids in spherical coordinates. *Monthly Weather Review*, 127(3):2204–2210, 1999. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1999)127<2204:FASOCR>2.0.CO;2.

B. Josse, P. Simon, and V.-H. Peuch. Radon global simulations with the multiscale chemistry and transport model MOCAGE. *Tellus B*, 56(4):339–356, September 2004. ISSN 0280-6509. doi: 10.1111/j.1600-0889.2004.00112.x. URL http://www.tellusb.net/index.php/tellusb/article/view/16448.

Akira Kageyama and Tetsuya Sato. The "Yin-Yang Grid": An Overset Grid in Spherical Geometry. *Geochemistry, Geophysics, Geosystems*, 5:1–15, 2004.

George Karypis and Vipin Kumar. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. *Citeseer*, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.376.

Dave Kennison. Non-uniform grids that NCL can contour, 2014. URL https://www.ncl.ucar.edu/Document/Graphics/contour_grids.shtml.

Peter Kogge. Exascale computing study: Technology challenges in achieving exascale systems. *DARPA IPTO*, 2008. URL http://www.cp.eng.chula.ac.th/~piak/teaching/cbs-it-seminar/2012/TR-2008-13.pdf.

Peter Kogge and John Shalf. Exascale Computing Trends : Adjusting to the "New Normal". *IEEE CS*, 2004.

Yoshio Kurihara. Numerical integration of the primitive equations on a spherical grid. *Monthly Weather Review*, 93(7), 1965.

J. P. René Laprise and A. Plante. A class of semi-lagrangian integrated-mass (SLM) numerical transport algorithms. *Monthly Weather Review*, 1995. URL http://journals.ametsoc.org/doi/pdf/10.1175/1520-0493(1995)123%3C0553%3AACOSLI%3E2.0.CO%3B2.

Peter Hjort Lauritzen, William C. Skamarock, M. J. Prather, and M. a. Taylor. A standard test case suite for two-dimensional linear transport on the sphere. *Geoscientific Model Development Discussions*, 5(1):189–228, January 2012. ISSN 1991-962X. doi: 10.5194/gmdd-5-189-2012. URL http://www.geosci-model-dev.net/5/887/2012/gmd-5-887-2012.html.

Peter Hjort Lauritzen, P. a. Ullrich, Christiane Jablonowski, P. a. Bosler, D. Calhoun, a. J. Conley, T. Enomoto, L. Dong, S. Dubey, O. Guba, a. B. Hansen, E. Kaas, J. Kent, J.-F. Lamarque, M. J. Prather, D. Reinert, V. V. Shashkin, William C. Skamarock, B. Sø rensen, M. a. Taylor, and M. a. Tolstykh. A standard test case suite for two-dimensional linear transport on the sphere: results from a collection of state-of-the-art schemes. *Geoscientific Model Development*, 6(3):105–145, September 2013. ISSN 1991-9603. doi: 10.5194/gmdd-6-4983-2013. URL http://www.geosci-model-dev-discuss.net/6/4983/2013/.

Franck Lefèvre and G. P. Brasseur. Chemistry of the 1991–1992 stratospheric winter: Three-dimensional model simulations. *Journal of Geophysical Research: Atmospheres (1984–2012)*, 99: 8183–8195, 1994. URL http://onlinelibrary.wiley.com/doi/10.1029/93JD03476/full.

Sanjiva K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, November 1992. ISSN 00219991. doi: 10.1016/0021-9991(92)90324-R. URL http://linkinghub.elsevier.com/retrieve/pii/002199919290324R.

B. P. Leonard, A. P. Lock, and M. K. MacVean. Conservative explicit unrestricted-time-step multidimensional constancy-preserving advection schemes. *Monthly Weather Review*, 1996. URL http://www.rsmas.miami.edu/personal/miskandarani/Courses/MPO662/Leonard/mwrNov96.pdf.

Randall J. LeVeque. *Finite volume methods for hyperbolic problems*. Cambridge University Press, 2002.

Jianwei Li, W Liao, and A Choudhary. Parallel netCDF: A high-performance scientific I/O interface. In *Supercomputing, 2003 ACM/IEEE Conference*, 2003. ISBN 1581136951. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1592942.

Wei-keng Liao and Rajeev Thakur. MPI-IO. Technical report, Northwestern University, 2014. URL http://www.mcs.anl.gov/papers/P5162-0714.pdf.

Shian-Jiann Lin and Richard B. Rood. Multidimensional flux-form semi-Lagrangian transport schemes. *Monthly Weather Review*, 124:2046—-2070, 1996.

Bennert Machenhauer, Eigil Kaas, and Peter Hjort Lauritzen. Finite volume methods in meteorology. *Handbook of Numerical Analysis*, M(08):3–120, 2009. doi: 10.1016/S1570-8659(08)00201-9.

D. Majewski, D. Liermann, and P. Prohl. The operational global icosahedral-hexagonal gridpoint model GME: Description and high-resolution tests. *Monthly Weather Review*, (1968):319–338, 2002. URL http://journals.ametsoc.org/doi/full/10.1175/1520-0493(2002)130%3C0319:TOGIHG%3E2.0.CO%3B2.

Hans Meueur, Erich Strohmaier, Jack Dongarra, and Horst Simon. Top 500 Report, 2014. URL http://www.top500.org/list/2014/06/.

Hiroaki Miura. An Upwind-Biased Conservative Advection Scheme for Spherical Hexagonal–Pentagonal Grids. *Monthly Weather Review*, 135(12):4038–4044, December 2007. ISSN 0027-0644. doi: 10.1175/2007MWR2101.1.

S. Moorthi, R. W. Higgins, and J. R. Bates. A global multilevel atmospheric model using a vector semi-Lagrangian finite-difference scheme. Part I: Adiabatic formulation. *Monthly weather review*,

1993. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1993)121<0244:AGMAMU>2.0.CO;2http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1995)123<1523:AGMAMU>2.0.CO;2.

R. D. Nair and B. Machenhauer. The mass-conservative cell-integrated semi-Lagrangian advection scheme on the sphere. *Monthly Weather Review*, pages 649–667, 2002. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(2002)130<0649:TMCCIS>2.0.CO;2.

Ramachandran D. Nair and Christiane Jablonowski. Moving Vortices on the Sphere: A Test Case for Horizontal Advection Problems. *Monthly Weather Review*, 136(2):699–711, February 2008. ISSN 0027-0644. doi: 10.1175/2007MWR2105.1. URL http://journals.ametsoc.org/doi/abs/10.1175/2007MWR2105.1.

NVIDIA. NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110. Technical report, NVIDIA, 2012. URL http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf.

François Pellegrini. PT-Scotch and LibPTScotch 0.6 User's Guide. Technical report, INRIA Bordeaux Sud-Ouest, 2012.

X. Peng, Feng Xiao, and Keiko Takahashi. Conservative constraint for a quasi-uniform overset grid on the sphere. *Quarterly Journal of the Royal Meteorological Society*, pages 979–996, 2006. doi: 10.1256/qj.05.18. URL http://onlinelibrary.wiley.com/doi/10.1256/qj.05.18/abstract.

N. A. Phillips. A map projection system suitable for large-scale numerical weather prediction. *J. Meteor. Soc. Japan*, 1957. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:A+map+projection+system+suitable+for+large-scale+numerical+weather+prediction#0.

N. A. Phillips. Numerical integration of the hydrostatic system of equations with a modified version of the Eliassen finite-difference grid. In *Proc. International Symposium on Numerical Weather Prediction*, number September, 1962. URL http://www.ccpo.odu.edu/~klinck/Reprints/PDF/phillipsMWR59.pdfhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Numerical+integration+of+the+hy-+drostatic+system+of+equations+with+a+modified+version+of+the+Eliassen+Finite-difference+grid#0.

A. Praga, D. Cariolle, and L. Giraud. Pangolin v1.0, a conservative 2-D advection model towards large-scale parallel calculation. *Geoscientific Model Development*, 8:205–220, 2015. ISSN 1991-9603. doi: 10.5194/gmd-8-205-2015. URL http://www.geosci-model-dev.net/8/205/2015/.

Michael J. Prather. Numerical advection by conservation of second-order moments. *Journal of Geophysical Research*, 91(D6):6671—-6681, 1986.

A. Priestley. A quasi-conservative version of the semi-Lagrangian advection scheme. *Monthly Weather Review*, 1993. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1993)121<0621:AQCVOT>2.0.CO;2.

R. J. Purser and L. M. Leslie. An efficient interpolation procedure for high-order three-dimensional semi-Lagrangian models. *Monthly Weather Review*, 1991. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1991)119<2492:AEIPFH>2.0.CO;2.

William M. Putman and Shian-Jiann Lin. Finite-volume transport on various cubed-sphere grids. *Journal of Computational Physics*, 227(1):55–78, November 2007. ISSN 00219991. doi: 10.1016/j.jcp.2007.07.022. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999107003105.

Abdessamad Qaddouri and Vivian Lee. The canadian global environmental multiscale model on the yin-yang grid system. *Quarterly Journal of the Royal Meteorological Society*, 137(660):1913–1926, 2011.

M. Rančić, R. J. Purser, and F. Mesinger. A global shallow-water model using an expanded spherical cube: Gnomonic versus conformal coordinates. *Quarterly Journal of the Royal Meteorological Society*, 122(532):959–982, 1996.

Miodrag Rančić. Semi-lagrangian piecewise biparabolic scheme for two-dimensional horizontal advection of a passive scalar. *Monthly weather review*, 120(7):1394–1406, 1992.

Philip J. Rasch. Conservative shape-preserving two-dimensional transport on a spherical reduced grid. *Monthly weather review*, 1994. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1994)122%3C1337:CSPTDT%3E2.0.CO%3B2.

H. Ritchie and C. Temperton. Implementation of the semi-Lagrangian method in a high-resolution version of the ECMWF forecast model. *Monthly Weather ...*, 1995. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1995)123<0489:IOTSLM>2.0.CO;2.

R. B. Rood. Numerical advection algorithms and their role in atmospheric transport and chemistry models. *Reviews of geophysics*, 1987. URL http://onlinelibrary.wiley.com/doi/10.1029/RG025i001p00071/full.

Youcef Saad and Martin H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7 (3):856–869, July 1986. ISSN 0196-5204. doi: 10.1137/0907058. URL http://epubs.siam.org/doi/abs/10.1137/0907058.

Robert Sadourny. Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids. *Monthly Weather Review*, 100(2), 1972. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1972)100%3C0136:CFAOTP%3E2.3.CO;2.

Robert Sadourny, Akio Arakawa, and Y Mintz. Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere. *Monthly Weather Review*, 96(443):2–7, 1968. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1968)096<0351:IOTNBV>2.0.CO;2.

A. Sandu, J. G. Verwer, M. Van Loon, and G. R. Carmichael. Benchmarking Sti ODE Solvers for Atmospheric Chemistry Problems I : Implicit versus Explicit 1 Introduction. *Environmental Research*, 52246(85):1–23, 1996.

Piotr K. Smolarkiewicz and Janusz A. Pudykiewicz. A Class of Semi-Lagrangian Approximations for Fluids. *Journal of the Atmospheric Sciences*, 49(22):2082–2096, November 1992. ISSN 0022-4928. doi: 10.1175/1520-0469(1992)049⟨2082:ACOSLA⟩2.0.CO;2. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0469(1992)049<2082:ACOSLA>2.0.CO;2.

Andrew Staniforth and Jean Côté. Semi-Lagrangian integration schemes for atmospheric models-a review. *Monthly Weather Review*, 1991. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1991)119<2206:SLISFA>2.0.CO;2.

Andrew Staniforth and John Thuburn. Horizontal grids for global weather and climate prediction models: a review. *Quarterly Journal of the Royal Meteorological Society*, 138(662):1–26, January 2012. ISSN 00359009. doi: 10.1002/qj.958.

P. N. Swarztrauber. Multiprocessor FFTS. *Parallel computing*, 5(1-2):197–210, July 1987. ISSN 01678191. doi: 10.1016/0167-8191(87)90018-4. URL http://linkinghub.elsevier.com/retrieve/pii/0167819187900184http://www.sciencedirect.com/science/article/pii/0167819187900184.

Richard Swinbank and R James Purser. Fibonacci grids: A novel approach to global modelling. *Quarterly Journal of the Royal Meteorological Society*, 132(619):1769–1793, July 2006. ISSN 00359009. doi: 10.1256/qj.05.227. URL http://doi.wiley.com/10.1256/qj.05.227http://onlinelibrary.wiley.com/doi/10.1256/qj.05.227/abstract.

H. Teyssèdre, M. Michou, H.L. Clark, B. Josse, F. Karcher, D. Olivié, V.H. Peuch, D. Saint-Martin, Daniel Cariolle, J.L. Attié, P. Nédélec, P. Ricaud, V. Thouret, R.J. van der A, and A. Volz-Thomas. A new tropospheric and stratospheric Chemistry and Transport Model MOCAGE-Climat for multi-year studies: evaluation of the present-day climatology and sensitivity. *Atm. Chem. Phy*, 7(22):5815–5860, 2007.

Rajeev Thakur, Pavan Balaji, and Darius Buntinas. MPI at Exascale. *Procceedings of SciDAC*, pages 1–14, 2010. URL http://aegjcef.unixer.de/publications/img/mpi_exascale.pdf.

M. A. Tolstykh. Semi-Lagrangian high-resolution model of the atmosphere for numerical weather prediction. *Russian Meteorology and Hydrology*, 2001. URL http://www.wmo.int/pages/prog/www/Documents/PublicWeb/www/gdpfs/GDPFS-NWP_Annualreports11/2011_RussianFed_EN.dochttp://elibrary.ru/item.asp?id=13383817.

Hirofumi Tomita, Motohiko Tsugawa, Masaki Satoh, and Koji Goto. Shallow Water Model on a Modified Icosahedral Geodesic Grid by Using Spring Dynamics. *Journal of Computational Physics*, 174(2):579–613, December 2001. ISSN 00219991. doi: 10.1006/jcph.2001.6897. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999101968974.

UCAR. NCL Reference. URL https://www.ncl.ucar.edu/Document/Manuals/Ref_Manual/.

Bram van Leer. Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. *Journal of computational physics*, 23(3):276—-299, 1977. URL http://www.sciencedirect.com/science/article/pii/002199917790095X.

Bram van Leer. Towards the ultimate conservation difference scheme: V. A second-order sequel to godunov's method. 136:101–136, 1979.

J. G . Verwer. Gauss-Seidel iteration for stiff ODEs from chemical kinetics. *SIAM Journal on Scientific Computing*, 15(5):1243–1250, September 1994. ISSN 1064-8275. doi: 10.1137/0915076. URL http://epubs.siam.org/doi/abs/10.1137/0915076.

J. B. White and J. J. Dongarra. High-performance high-resolution semi-Lagrangian tracer transport on a sphere. *Journal of Computational Physics*, 230(17):6778–6799, July 2011. ISSN 00219991. doi: 10.1016/j.jcp.2011.05.008.

David L. Williamson. The evolution of dynamical cores for global atmospheric models. *Meteorological Society Of Japan*, 85:241–269, 2007.

David L. Williamson and Philip J. Rasch. Two-dimensional semi-Lagrangian transport with shape-preserving interpolation. *Monthly Weather Review*, 1989. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1989)117%3C0102:TDSLTW%3E2.0.CO;2.

David L. Williamson and Philip J. Rasch. Water vapor transport in the NCAR CCM2. *Tellus A*, 1994. URL http://onlinelibrary.wiley.com/doi/10.1034/j.1600-0870.1994.00004.x/abstract.

David L. Williamson, John B. Drake, James J. Hack, Rüdiger Jakob, and Paul N Swarztrauber. A standard test set for numerical approximations to the shallow water equations in spherical geometry. *Journal of Computational Physics*, 102(1):211–224, 1992.

F. Xiao. Conservative and oscillation-less atmospheric transport schemes based on rational functions. *Journal of Geophysical Research*, 107(D22):4609, 2002. ISSN 0148-0227. doi: 10.1029/2001JD001532. URL http://doi.wiley.com/10.1029/2001JD001532.

Mohamed Zerroukat, Nigel Wood, and Andrew Staniforth. SLICE: A Semi-Lagrangian Inherently Conserving and Efficient scheme for transport problems. *Quarterly Journal of the Royal Meteorological Society*, 128(586):2801–2820, October 2002. ISSN 1477870X. doi: 10.1256/qj.02.69. URL http://doi.wiley.com/10.1256/qj.02.69.

Mohamed Zerroukat, Nigel Wood, and Andrew Staniforth. A monotonic and positive–definite filter for a Semi-Lagrangian Inherently Conserving and Efficient (SLICE) scheme. *Quarterly Journal of the Royal Meteorological Society*, 131(611):2923–2936, October 2005. ISSN 00359009. doi: 10.1256/qj.04.97. URL http://doi.wiley.com/10.1256/qj.04.97.

Glossary

API Application Programming Interface. 55, 56

CFL Courant-Friedrichs-Lewy. 20, 24, 41

CMC Canadian Meteorological Center. 24, 34

Conformal projection this projection preserves the angles locally (i.e at the intersection of the lines. In particular, parallel cross meridians at right angles. In practice, dimensions are deformed when far from the center (they do not preserve areas) and as such, are used for global instead of regional maps. Most conformal maps have singularities. 32, 33

CPU Central Processing Unit. It carries out instructions stored in a program. For that, it executes arithmetical, logical and I/O operations. 53

CTM Chemistry Transport Model. 7, 15, 21, 26, 32, 53, 72

DCISL Departure Cell-Integrated Semi-Lagrangian. 25

DWD Deutscher Wetterdienst. German weather agency.. 34

ECMWF European Centre for Medium-Range Weather Forecasts. It is an European centre providing medium-range weather and seasonal forecasts. 22, 24, 75

Endianess Describes how the bytes of a word are ordered in the memory. In big-endian configurations, the most significant byte has the smallest address. .In little-endian, the least significant byte has the largest address. 64

FFT Fast Fourier Transform. 22

flops floating-point operations per seconds. 53

GCM Global Climate Model. 15, 27, 72

GMD Geoscientific Model Development. 48, 87

Gnomonic projection a projection which represents the shortest route between two points (greats circles) as straight lines, which makes it easy to find the shortest path between two points. However, distance and shape are distorted, except near the center of the projection. 32

GPU Graphics Processing Unit. 53

HDF Hierarchical Data Format. 65

JMA Japanese Meteorological Agency. 24, 35

Lambert equal-area projection As an azimuthal projection, it maps a portion of the globe to a tangent plane. It also preserves the areas. When used for a global map, shapes are distorted near the boundaries. 36

MCT Modèle de Chimie-Transport. 11

MPI Message Passing Interface. 55, 62

NCAR National Center for Atmospheric Research. It is an federal organisation, located in the US, which focus on climate, weather, air chemistry and other related areas.. 32

netCDF Network Common Data Form. 64

NWP Numerical Weather Prediction model. 15, 22

ODE Ordinary Differential Equation. 66, 75

OpenMP Open Multi-Processing. 56

Overhead happens when additional computer resources are needed to achieve a particular goal. For example, MPI overhead can happen for blocking operations, ,i.e a certain process loses time by waiting a request. 55, 56

PGAS Partitioned Global Adress Space. 54

PPM Piecewise Parabolic Method. 27, 29

Process can be defined as an adress space and the current state of a program (program counter, call stack, register values). Different processes do not share resources between them, contrary to threads. 55

PVM Parallel Virtual Machine. 55

Robinson projection a projection, which is neither area-preserving nor conformal but aims to represents the whole world in a ”good-looking” way. In particular, it aims to reduce the global distortion. Used by the National Geographic Society between 1988 and 1998. 39

SOM Second-Order Moments. 28

Stencil for numerical schemes, it indicates the number of adjacent cells needed to compute the given quantity. 21

Stereographic projection this conformal projection maps the whole sphere onto a plane, except the projection point. For a projection center at the North pole, It computes the intersection of the line between the pole and the considered point with the plane $z = 0$. 33

Stratosphere second atmosphere layer. It is located between the troposphere and the mesosphere. At mid-latitudes, it is situated between 18-13km and 50km. 15

Thread smallest set of instructions that can be managed independently by the operating system (program counter and call stack). It is a lightweight process. If several threads exist in the same process, they share resources (e.g memory). 53, 55

Troposphere first atmosphere layer from the surface of the Earth to the tropopause, which separates it from the stratosphere. It contains more than 80% of the total mass of the atmosphere. At mid-latitudes, its depth is around 17km. 15

UPC Unified Parallel C. 54